

globus rls client
5.2

Generated by Doxygen 1.8.4

Sat Aug 17 2013 14:01:22

Contents

1	Main Page	1
2	Module Index	2
2.1	Modules	2
3	Data Structure Index	2
3.1	Data Structures	2
4	Module Documentation	3
4.1	Status Codes	3
4.1.1	Detailed Description	3
4.1.2	Macro Definition Documentation	4
4.2	Miscellaneous	7
4.2.1	Detailed Description	8
4.2.2	Macro Definition Documentation	8
4.2.3	Enumeration Type Documentation	8
4.2.4	Function Documentation	9
4.3	Query Results	13
4.3.1	Detailed Description	13
4.3.2	Function Documentation	13
4.4	Activation	15
4.4.1	Detailed Description	15
4.4.2	Macro Definition Documentation	15
4.4.3	Variable Documentation	15
4.5	Connection Management	16
4.5.1	Detailed Description	16
4.5.2	Macro Definition Documentation	16
4.5.3	Function Documentation	16
4.6	LRC Operations	19
4.6.1	Detailed Description	20
4.6.2	Macro Definition Documentation	20
4.6.3	Function Documentation	20
4.7	RLI Operations	33
4.7.1	Detailed Description	33
4.7.2	Function Documentation	33
5	Data Structure Documentation	38
5.1	globus_rls_attribute_object_t Struct Reference	38

5.1.1	Detailed Description	38
5.1.2	Field Documentation	38
5.2	globus_rls_attribute_t Struct Reference	38
5.2.1	Detailed Description	38
5.2.2	Field Documentation	38
5.3	globus_rls_handle_t Struct Reference	39
5.3.1	Detailed Description	39
5.3.2	Field Documentation	39
5.4	globus_rls_rli_info_t Struct Reference	40
5.4.1	Detailed Description	40
5.4.2	Field Documentation	40
5.5	globus_rls_sender_info_t Struct Reference	40
5.5.1	Detailed Description	40
5.5.2	Field Documentation	40
5.6	globus_rls_stats_t Struct Reference	41
5.6.1	Detailed Description	41
5.6.2	Field Documentation	41
5.7	globus_rls_string2_bulk_t Struct Reference	41
5.7.1	Detailed Description	41
5.8	globus_rls_string2_t Struct Reference	41
5.8.1	Detailed Description	41
5.8.2	Field Documentation	41

Index 42

1 Main Page

The Globus Replica Location Service (RLS) C API provides functions to view and update data in a RLS catalog. There are 2 types of RLS servers, Local Replica Catalog (LRC) servers, which maintain Logical to Physical File Name mappings (LFN to PFN), and Replica Location Index (RLI) servers, which maintain LFN to LRC mappings. Note an RLS server can act as both an LRC and RLI server.

Functions are divided into the following groups:

- **Activation** (p. 15)
- **Connection Management** (p. 16)
- **Operations on an LRC server** (p. 19)
- **Operations on an RLI server** (p. 33)
- **Miscellaneous Types and Functions** (p. 7)
- **Query Results** (p. 13)

- **Status Codes** (p. 3)

Applications using this API should include **globus_rls_client.h**, and be linked with the library **globus_rls_client - FLAVOR**.

2 Module Index

2.1 Modules

Here is a list of all modules:

Status Codes	3
Miscellaneous	7
Query Results	13
Activation	15
Connection Management	16
LRC Operations	19
RLI Operations	33

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

globus_rls_attribute_object_t Globus_rls_client_lrc_attr_search() returns a list of these structures which include the object name (LFN or PFN) and attribute value found by the query	38
globus_rls_attribute_t Object (LFN or PFN) attribute type	38
globus_rls_handle_t RLS Client Handle	39
globus_rls_rli_info_t Information about RLI server, returned by globus_rls_client_lrc_rli_info() (p. 32) and globus_rls_client_lrc_rli_list() (p. 32)	40
globus_rls_sender_info_t Information about server sending updates to an rli, returned by globus_rls_client_rli_sender_list() (p. 35)	40
globus_rls_stats_t Various configuration options and statistics about an RLS server returned in the following structures by globus_rls_client_stats() (p. 10)	41

globus_rls_string2_bulk_t	
String pair result with return code, returned by bulk query operations	41
globus_rls_string2_t	
String pair result	41

4 Module Documentation

4.1 Status Codes

Macros

- #define **GLOBUS_RLS_SUCCESS** 0
- #define **GLOBUS_RLS_GLOBUSERR** 1
- #define **GLOBUS_RLS_INVHANDLE** 2
- #define **GLOBUS_RLS_BADURL** 3
- #define **GLOBUS_RLS_NOMEMORY** 4
- #define **GLOBUS_RLS_OVERFLOW** 5
- #define **GLOBUS_RLS_BADARG** 6
- #define **GLOBUS_RLS_PERM** 7
- #define **GLOBUS_RLS_BADMETHOD** 8
- #define **GLOBUS_RLS_INVSERVER** 9
- #define **GLOBUS_RLS_MAPPING_NEXIST** 10
- #define **GLOBUS_RLS_LFN_EXIST** 11
- #define **GLOBUS_RLS_LFN_NEXIST** 12
- #define **GLOBUS_RLS_PFN_EXIST** 13
- #define **GLOBUS_RLS_PFN_NEXIST** 14
- #define **GLOBUS_RLS_LRC_EXIST** 15
- #define **GLOBUS_RLS_LRC_NEXIST** 16
- #define **GLOBUS_RLS_DBERROR** 17
- #define **GLOBUS_RLS_RLI_EXIST** 18
- #define **GLOBUS_RLS_RLI_NEXIST** 19
- #define **GLOBUS_RLS_MAPPING_EXIST** 20
- #define **GLOBUS_RLS_INV_ATTR_TYPE** 21
- #define **GLOBUS_RLS_ATTR_EXIST** 22
- #define **GLOBUS_RLS_ATTR_NEXIST** 23
- #define **GLOBUS_RLS_INV_OBJ_TYPE** 24
- #define **GLOBUS_RLS_INV_ATTR_OP** 25
- #define **GLOBUS_RLS_UNSUPPORTED** 26
- #define **GLOBUS_RLS_TIMEOUT** 27
- #define **GLOBUS_RLS_TOO_MANY_CONNECTIONS** 28
- #define **GLOBUS_RLS_ATTR_VALUE_NEXIST** 29
- #define **GLOBUS_RLS_ATTR_INUSE** 30

4.1.1 Detailed Description

All of the functions in the API that return status return it in a `globus_result_t` structure. Prior to version 2.0.0 an integer status was returned. The `globus_result_t` structure includes an integer "type" which is set to one of the status codes defined below (the same values that were returned by earlier versions of the API). The function **globus_rls_client_error_info()** (p. 11) may be used to extract the status code and/or error message from a `globus_result_t`. **GLOBUS_SUCCESS** is returned when the operation was successful.

4.1.2 Macro Definition Documentation

4.1.2.1 #define GLOBUS_RLS_SUCCESS 0

Operation succeeded.

4.1.2.2 #define GLOBUS_RLS_GLOBUSERR 1

An error was returned by the Globus I/O module.

4.1.2.3 #define GLOBUS_RLS_INVHANDLE 2

The **globus_rls_handle_t** (p. 39) handle is invalid.

4.1.2.4 #define GLOBUS_RLS_BADURL 3

The URL could not be parsed.

4.1.2.5 #define GLOBUS_RLS_NOMEMORY 4

Out of memory.

4.1.2.6 #define GLOBUS_RLS_OVERFLOW 5

A result was too large to fit in buffer.

4.1.2.7 #define GLOBUS_RLS_BADARG 6

Bad argument (eg NULL where string pointer expected).

4.1.2.8 #define GLOBUS_RLS_PERM 7

Client does not have permission for requested action.

4.1.2.9 #define GLOBUS_RLS_BADMETHOD 8

RPC error, invalid method name sent to server.

4.1.2.10 #define GLOBUS_RLS_INVSERVER 9

LRC request made to RLI server or vice versa.

4.1.2.11 #define GLOBUS_RLS_MAPPING_NEXIST 10

LFN,PFN (LRC) or LFN,LRC (RLI) mapping doesn't exist.

4.1.2.12 #define GLOBUS_RLS_LFN_EXIST 11

LFN already exists in LRC or RLI database.

4.1.2.13 #define GLOBUS_RLS_LFN_NEXIST 12

LFN doesn't exist in LRC or RLI database.

4.1.2.14 #define GLOBUS_RLS_PFN_EXIST 13

PFN already exists in LRC database.

4.1.2.15 **#define GLOBUS_RLS_PFN_NEXIST 14**

PFN doesn't exist in LRC database.

4.1.2.16 **#define GLOBUS_RLS_LRC_EXIST 15**

LRC already exists in LRC or RLI database.

4.1.2.17 **#define GLOBUS_RLS_LRC_NEXIST 16**

LRC doesn't exist in RLI database.

4.1.2.18 **#define GLOBUS_RLS_DBERROR 17**

Database error.

4.1.2.19 **#define GLOBUS_RLS_RLI_EXIST 18**

RLI already exists in LRC database.

4.1.2.20 **#define GLOBUS_RLS_RLI_NEXIST 19**

RLI doesn't exist in LRC.

4.1.2.21 **#define GLOBUS_RLS_MAPPING_EXIST 20**

LFN,PFN (LRC) or LFN,LRC (RLI) mapping already exists.

4.1.2.22 **#define GLOBUS_RLS_INV_ATTR_TYPE 21**

Invalid attribute type, see `globus_uls_attr_type_t`.

4.1.2.23 **#define GLOBUS_RLS_ATTR_EXIST 22**

Attribute already exists.

4.1.2.24 **#define GLOBUS_RLS_ATTR_NEXIST 23**

Attribute doesn't exist.

4.1.2.25 **#define GLOBUS_RLS_INV_OBJ_TYPE 24**

Invalid object type, see `globus_uls_obj_type_t`.

4.1.2.26 **#define GLOBUS_RLS_INV_ATTR_OP 25**

Invalid attribute search operator, see `globus_uls_attr_op_t`.

4.1.2.27 **#define GLOBUS_RLS_UNSUPPORTED 26**

Operation is unsupported.

4.1.2.28 **#define GLOBUS_RLS_TIMEOUT 27**

IO timeout.

4.1.2.29 **#define GLOBUS_RLS_TOO_MANY_CONNECTIONS 28**

Too many connections.

4.1.2.30 #define GLOBUS_RLS_ATTR_VALUE_NEXIST 29

Attribute with specified value not found.

4.1.2.31 #define GLOBUS_RLS_ATTR_INUSE 30

Attribute in use by some object, can't be deleted.

4.2 Miscellaneous

Data Structures

- struct **globus_rls_attribute_t**
- struct **globus_rls_stats_t**

Macros

- #define **RLS_LRCSERVER** 0x1
- #define **RLS_RLISERVER** 0x2
- #define **RLS_RCVLFNLIST** 0x4
- #define **RLS_RCVBLOOMFILTER** 0x8
- #define **RLS_SNDLFNLIST** 0x10
- #define **RLS_SNDBLOOMFILTER** 0x20
- #define **RLS_INITIALIZED** 0x40

Enumerations

- enum **globus_rls_pattern_t** {
 rls_pattern_unix,
 rls_pattern_sql }
- enum **globus_rls_attr_type_t** {
 globus_rls_attr_type_date,
 globus_rls_attr_typeflt,
 globus_rls_attr_type_int,
 globus_rls_attr_type_str }
- enum **globus_rls_obj_type_t** {
 globus_rls_obj_lrc_lfn,
 globus_rls_obj_lrc_pfn,
 globus_rls_obj_rli_lfn,
 globus_rls_obj_rli_lrc }
- enum **globus_rls_attr_op_t** {
 globus_rls_attr_op_all,
 globus_rls_attr_op_eq,
 globus_rls_attr_op_ne,
 globus_rls_attr_op_gt,
 globus_rls_attr_op_ge,
 globus_rls_attr_op_lt,
 globus_rls_attr_op_le,
 globus_rls_attr_op_btw,
 globus_rls_attr_op_like }
- enum **globus_rls_admin_cmd_t** {
 globus_rls_admin_cmd_ping,
 globus_rls_admin_cmd_quit,
 globus_rls_admin_cmd_ssu }

Functions

- globus_result_t **globus_rls_client_admin** (globus_rls_handle_t *h, globus_rls_admin_cmd_t cmd)
- globus_result_t **globus_rls_client_get_configuration** (globus_rls_handle_t *h, char *option, globus_list_t **conf_list)
- globus_result_t **globus_rls_client_set_configuration** (globus_rls_handle_t *h, char *option, char *value)

- globus_result_t **globus_rls_client_stats** (globus_rls_handle_t *h, globus_rls_stats_t *rlsstats)
- char * **globus_rls_client_attr2s** (globus_rls_attribute_t *attr, char *buf, int buflen)
- globus_result_t **globus_rls_client_s2attr** (globus_rls_attr_type_t type, char *sval, globus_rls_attribute_t *attr)
- globus_result_t **globus_rls_client_error_info** (globus_result_t r, int *rc, char *buf, int buflen, globus_bool_t preserve)
- int **globus_list_len** (globus_list_t *len)
- char * **globus_rls_errmsg** (int rc, char *specificmsg, char *buf, int buflen)

4.2.1 Detailed Description

Miscellaneous functions and types.

4.2.2 Macro Definition Documentation

4.2.2.1 #define RLS_LRCSERVER 0x1

Server is LRC server.

4.2.2.2 #define RLS_RLISERVER 0x2

Server is RLI server.

4.2.2.3 #define RLS_RCVLFNLIST 0x4

RLI accepts LFN list updates.

4.2.2.4 #define RLS_RCVBLOOMFILTER 0x8

RLI accepts Bloom filter updates.

4.2.2.5 #define RLS_SNDLFNLIST 0x10

LRC sends LFN list updates.

4.2.2.6 #define RLS_SNDBLOOMFILTER 0x20

LRC sends Bloom filter updates.

4.2.2.7 #define RLS_INITIALIZED 0x40

RLC is fully initialized.

4.2.3 Enumeration Type Documentation

4.2.3.1 enum globus_rls_pattern_t

Wildcard character style.

Enumerator

rls_pattern_unix Unix file globbing chars (*, ?)

rls_pattern_sql SQL "like" wildcards (% , _)

4.2.3.2 enum globus_rls_attr_type_t

Attribute Value Types.

Enumerator

globus_rls_attr_type_date Date (time_t).
globus_rls_attr_type_flt Floating point (double).
globus_rls_attr_type_int Integer (int).
globus_rls_attr_type_str String (char *).

4.2.3.3 enum globus_rls_obj_type_t

Object types in LRC and RLI databases.

Enumerator

globus_rls_obj_lrc_lfn LRC Logical File Name.
globus_rls_obj_lrc_pfn LRC Physical File Name.
globus_rls_obj_rli_lfn RLI Logical File Name.
globus_rls_obj_rli_lrc RLI LRC URL.

4.2.3.4 enum globus_rls_attr_op_t

Attribute Value Query Operators.

Enumerator

globus_rls_attr_op_all All values returned.
globus_rls_attr_op_eq Values matching operand 1 returned.
globus_rls_attr_op_ne Values not matching operand 1.
globus_rls_attr_op_gt Values greater than operand 1.
globus_rls_attr_op_ge Values greater than or equal to op1.
globus_rls_attr_op_lt Values less than operand 1.
globus_rls_attr_op_le Values less than or equal to op1.
globus_rls_attr_op_btw Values between operand1 and 2.
globus_rls_attr_op_like Strings "like" operand1 (SQL like)

4.2.3.5 enum globus_rls_admin_cmd_t

globus_rls_client_admin() (p. 9) commands.

Enumerator

globus_rls_admin_cmd_ping Verify RLS server responding.
globus_rls_admin_cmd_quit Tell RLS server to exit.
globus_rls_admin_cmd_ssu Tell LRC server to do softstate update.

4.2.4 Function Documentation

4.2.4.1 globus_result_t globus_rls_client_admin (globus_rls_handle_t * h, globus_rls_admin_cmd_t cmd)

Miscellaneous administrative operations.

Most operations require the admin privilege.

Parameters

<i>h</i>	Handle connected to RLS server.
<i>cmd</i>	Command to be sent to RLS server.

Return values

<i>GLOBUS_SUCCESS</i>	Command succeeded.
-----------------------	--------------------

4.2.4.2 `globus_result_t globus_rls_client_get_configuration (globus_rls_handle_t * h, char * option, globus_list_t ** conf_list)`

Get server configuration.

Client needs admin privilege.

Parameters

<i>h</i>	Handle connected to RLS server.
<i>option</i>	Configuration option to get. If NULL all options are retrieved.

Return values

<i>conf_list</i>	List of configuration options.
<i>GLOBUS_SUCCESS</i>	List of retrieved config options returned in <i>conf_list</i> , each datum is of type globus_rls_string2_t (p. 41). <i>conf_list</i> should be freed with globus_rls_client_free_list() (p. 13). There may be multiple "acl" entries in the list, since the access control list can include more than one entry. Each acl configuration value is consists of a regular expression (matched against grid-mapfile users or DNs), a colon, and space separated list of permissions the matching users are granted.

References `globus_rls_client_free_list()`, and `GLOBUS_RLS_NOMEMORY`.

4.2.4.3 `globus_result_t globus_rls_client_set_configuration (globus_rls_handle_t * h, char * option, char * value)`

Set server configuration option.

Client needs admin privilege.

Parameters

<i>h</i>	Handle connected to RLS server.
<i>option</i>	Configuration option to set.
<i>value</i>	New value for option.

Return values

<i>GLOBUS_SUCCESS</i>	Option set on server.
-----------------------	-----------------------

4.2.4.4 `globus_result_t globus_rls_client_stats (globus_rls_handle_t * h, globus_rls_stats_t * rlsstats)`

Retrieve various statistics from RLS server.

Requires stats privilege.

Parameters

<i>h</i>	Handle connected to RLS server.
<i>rlsstats</i>	Stats returned here.

Return values

<i>GLOBUS_SUCCESS</i>	Stats returned in <i>rlsstats</i> .
-----------------------	-------------------------------------

References `globus_rls_stats_t::flags`.

4.2.4.5 `char* globus_rls_client_attr2s (globus_rls_attribute_t * attr, char * buf, int buflen)`

Map attribute value to string.

Parameters

<i>attr</i>	Attribute to convert. If <i>attr->type</i> is globus_rls_attr_type_date (p.9) then the resulting string will be in the format MySQL uses by default, which is YYYYMMDDHHMMSS.
<i>buf</i>	Buffer to write string value to. Note if <i>attr->type</i> is globus_rls_attr_type_str (p.9) then <i>attr->val.s</i> is returned, and <i>buf</i> is unused.
<i>buflen</i>	Size of <i>buf</i> in bytes.

Return values

<i>String Value</i>	Attribute value converted to a string.
---------------------	--

References `globus_rls_attribute_t::d`, `globus_rls_attr_type_date`, `globus_rls_attr_typeflt`, `globus_rls_attr_typeint`, `globus_rls_attr_type_str`, `globus_rls_attribute_t::i`, `globus_rls_attribute_t::s`, `globus_rls_attribute_t::t`, `globus_rls_attribute_t::type`, and `globus_rls_attribute_t::val`.

4.2.4.6 `globus_result_t globus_rls_client_s2attr (globus_rls_attr_type_t type, char * sval, globus_rls_attribute_t * attr)`

Set **globus_rls_attribute_t** (p. 38) type and val fields from a type and string value.

Parameters

<i>type</i>	Attribute value type.
<i>sval</i>	String value to convert to binary. If type is globus_rls_attr_type_date (p. 9) <i>sval</i> should be in the form YYYY-MM-DD HH:MM:SS.
<i>attr</i>	Attribute whose type and val fields are to be set.

Return values

<i>GLOBUS_SUCCESS</i>	<i>attr->type</i> and <i>attr->val</i> successfully set.
-----------------------	--

References `globus_rls_attribute_t::d`, `globus_rls_attr_type_date`, `globus_rls_attr_typeflt`, `globus_rls_attr_typeint`, `globus_rls_attr_type_str`, `GLOBUS_RLS_BADARG`, `GLOBUS_RLS_INV_ATTR_TYPE`, `GLOBUS_RLS_NOMEMORY`, `globus_rls_attribute_t::i`, `globus_rls_attribute_t::s`, `globus_rls_attribute_t::t`, `globus_rls_attribute_t::type`, and `globus_rls_attribute_t::val`.

4.2.4.7 `globus_result_t globus_rls_client_error_info (globus_result_t r, int * rc, char * buf, int buflen, globus_bool_t preserve)`

Get error code and message from `globus_result_t` returned by this API.

Parameters

<i>r</i>	Result returned by RLS API function. <i>r</i> is freed by this call and should not be referenced again. If <i>preserve</i> is set then a new <i>globus_result_t</i> is constructed with the same values and returned as the function value.
<i>rc</i>	Address to store error code at. If NULL error code is not returned.
<i>buf</i>	Address to store error message at. If NULL error message is not returned.
<i>preserve</i>	If GLOBUS_TRUE then a new <i>globus_result_t</i> is constructed with the same values as the old and returned as the function value.
<i>buflen</i>	Size of <i>buf</i> .

Return values

<i>globus_result_t</i>	If <i>preserve</i> is set a new <i>globus_result_t</i> identical to <i>r</i> is returned, otherwise GLOBUS_SUCCESS.
------------------------	---

4.2.4.8 int globus_list_len (globus_list_t * len)

Compute length of list.

globus_list_size() is implemented using recursion, besides being inefficient it can run out of stack space when the list is large.

4.2.4.9 char* globus_uls_errmsg (int rc, char * specificmsg, char * buf, int buflen)

Map RLS status code to error string.

Parameters

<i>rc</i>	Status code.
<i>specificmsg</i>	If not NULL prepended (with a colon) to error string.
<i>buf</i>	Buffer to write error message to.
<i>buflen</i>	Length of <i>buf</i> . Message will be truncated to fit if too long.

Return values

<i>char</i>	* Returns <i>buf</i> , error message written to <i>buf</i> .
-------------	--

4.3 Query Results

Data Structures

- struct **globus_rls_attribute_object_t**
- struct **globus_rls_string2_t**
- struct **globus_rls_string2_bulk_t**

Functions

- **globus_result_t globus_rls_client_free_list** (globus_list_t *list)

4.3.1 Detailed Description

List results are returned as **globus_list_t**'s, list datums depend on the type of query (eg **globus_rls_string2_t** (p. 41), **globus_rls_attribute_t** (p. 38), etc). A list result should be freed with **globus_rls_client_free_list()** (p. 13) when it's no longer needed. RLS supports limiting the number of results returned by a single query using an offset and reslimit. The offset specifies which result to begin with, reslimit specifies how many results to return. Offset should begin at 0 to retrieve all records. If reslimit is 0 then all results are returned at once, unless the server has a limit on results configured. If NULL is passed as the offset argument then the API will repeatedly call the query function until are results are retrieved. The following are equivalent examples of how to print the lfn,pfn pairs returned by **globus_rls_client_lrc_get_lfn()** (p. 27):

```
globus_list_t *str2_list;
globus_list_t *p;
globus_rls_string2_t *str2;

// Retrieve all results, API will handle looping through partial results
// if the server has a limit configured. Error handling has been omitted.
globus_rls_client_lrc_get_lfn(h, "somepfn", NULL, 0, &str2_list);
for (p = str2_list; p; p = globus_list_rest(p)) {
    str2 = (globus_rls_string2_t *) globus_list_first(p);
    printf("lfn: %s pfn:%s\n", str2->s1, str2->s2);
}
globus_rls_client_free_list(str2_list);

// This code fragment retrieves results 5 at a time. Note offset is set
// to -1 when the server has no more results to return.
int offset = 0;

while (globus_rls_client_lrc_get_lfn(h, "somepfn", &offset, 5, &str2_list) == GLOBUS_SUCCESS) {
    for (p = str2_list; p; p = globus_list_rest(p)) {
        str2 = (globus_rls_string2_t *) globus_list_first(p);
        printf("lfn: %s pfn:%s\n", str2->s1, str2->s2);
    }
    globus_rls_client_free_list(str2_list);
    if (offset == -1)
        break;
}
```

4.3.2 Function Documentation

4.3.2.1 **globus_result_t globus_rls_client_free_list** (globus_list_t * list)

Free result list returned by one of the query functions.

Parameters

<i>list</i>	List returned by one of the query functions.
-------------	--

Return values

<i>GLOBUS_SUCCESS</i>	List and contents successfully freed.
-----------------------	---------------------------------------

References GLOBUS_RLS_BADARG.

4.4 Activation

Macros

- `#define GLOBUS_RLS_CLIENT_MODULE (&globus_rls_client_module)`

Variables

- `globus_module_descriptor_t globus_rls_client_module`

4.4.1 Detailed Description

This module must be activated before any functions in this API may be used. This module depends on other Globus modules `GLOBUS_COMMON_MODULE` and `GLOBUS_IO_MODULE`, which should be activated first:

```
globus_module_activate(GLOBUS_COMMON_MODULE);  
globus_module_activate(GLOBUS_IO_MODULE);  
globus_module_activate(GLOBUS_RLS_CLIENT_MODULE);
```

When finished modules should be deactivated in reverse order.

4.4.2 Macro Definition Documentation

4.4.2.1 `#define GLOBUS_RLS_CLIENT_MODULE (&globus_rls_client_module)`

RLS Module Name.

4.4.3 Variable Documentation

4.4.3.1 `globus_module_descriptor_t globus_rls_client_module`

RLS module.

4.5 Connection Management

Macros

- `#define GLOBUS_RLS_URL_SCHEME "rls"`
- `#define GLOBUS_RLS_URL_SCHEME_NOAUTH "rlsn"`
- `#define GLOBUS_RLS_SERVER_DEFPORT 39281`
- `#define MAXERRMSG 1024`

Functions

- `void globus_rls_client_certificate (char *certfile, char *keyfile)`
- `void globus_rls_client_proxy_certificate (char *proxy)`
- `globus_result_t globus_rls_client_connect (char *url, globus_rls_handle_t **h)`
- `globus_result_t globus_rls_client_close (globus_rls_handle_t *h)`
- `int globus_rls_client_get_timeout ()`
- `void globus_rls_client_set_timeout (int seconds)`

4.5.1 Detailed Description

Functions to open and close connections to an RLS server.

4.5.2 Macro Definition Documentation

4.5.2.1 `#define GLOBUS_RLS_URL_SCHEME "rls"`

URL scheme to use when connecting to RLS server.

4.5.2.2 `#define GLOBUS_RLS_URL_SCHEME_NOAUTH "rlsn"`

URL scheme when connecting to RLS server without authentication.

4.5.2.3 `#define GLOBUS_RLS_SERVER_DEFPORT 39281`

Default port number that RLS server listens on.

4.5.2.4 `#define MAXERRMSG 1024`

Maximum length of error messages returned by server.

4.5.3 Function Documentation

4.5.3.1 `void globus_rls_client_certificate (char * certfile, char * keyfile)`

Set certificate used in authentication.

Sets environment variables X509_USER_CERT, X509_USER_KEY, and clears X509_USER_PROXY.

Parameters

<i>certfile</i>	Name of X509 certificate file.
<i>keyfile</i>	Name of X509 key file.

4.5.3.2 void globus_rls_client_proxy_certificate (char * proxy)

Set X509_USER_PROXY environment variable to specified file.

Parameters

<i>proxy</i>	Name of X509 proxy certificate file. If NULL clears X509_USER_PROXY.
--------------	--

4.5.3.3 globus_result_t globus_rls_client_connect (char * url, globus_rls_handle_t ** h)

Open connection to RLS server.

Parameters

<i>url</i>	URL of server to connect to. URL scheme should be RLS or RLSN , eg RLS://my.host . If the URL scheme is RLSN then no authentication is performed (the RLS server must be started with authentication disabled as well, this option is primarily intended for testing).
<i>h</i>	If the connection is successful * <i>h</i> will be set to the connection handle. This handle is required by all other functions in the API.

Return values

<i>GLOBUS_SUCCESS</i>	Handle <i>h</i> now connected to RLS server identified by <i>url</i> .
-----------------------	--

References GLOBUS_RLS_BADARG, GLOBUS_RLS_BADURL, GLOBUS_RLS_INVHANDLE, GLOBUS_RLS_NOMEMORY, GLOBUS_RLS_SERVER_DEFPORT, GLOBUS_RLS_SUCCESS, GLOBUS_RLS_URL_SCHEME, GLOBUS_RLS_URL_SCHEME_NOAUTH, and MAXERRMSG.

4.5.3.4 globus_result_t globus_rls_client_close (globus_rls_handle_t * h)

Close connection to RLS server.

Parameters

<i>h</i>	Connection handle to be closed, previously allocated by globus_rls_client_connect() (p. 17).
----------	---

Return values

<i>GLOBUS_SUCCESS</i>	Connection closed, <i>h</i> is no longer valid.
-----------------------	---

References globus_rls_handle_t::flags, GLOBUS_RLS_INVHANDLE, globus_rls_handle_t::handle, and globus_rls_handle_t::url.

4.5.3.5 int globus_rls_client_get_timeout ()

Get timeout for IO calls to RLS server.

If 0 IO calls do not timeout. The default is 30 seconds.

Return values

<i>timeout</i>	Seconds to wait before timing out an IO operation.
----------------	--

4.5.3.6 void globus_rls_client_set_timeout (int seconds)

Set timeout for IO calls to RLS server.

Parameters

<i>seconds</i>	Seconds to wait before timing out an IO operation. If 0 IO calls do not timeout. The default is 30 seconds.
----------------	---

4.6 LRC Operations

Data Structures

- struct **globus_rls_rli_info_t**

Macros

- #define **FRLI_BLOOMFILTER** 0x1
- #define **MAXURL** 256

Functions

- globus_result_t **globus_rls_client_lrc_attr_add** (globus_rls_handle_t *h, char *key, globus_rls_attribute_t *attr)
- globus_result_t **globus_rls_client_lrc_attr_add_bulk** (globus_rls_handle_t *h, globus_list_t *attr_obj_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_attr_create** (globus_rls_handle_t *h, char *name, globus_rls_obj_type_t objtype, globus_rls_attr_type_t type)
- globus_result_t **globus_rls_client_lrc_attr_delete** (globus_rls_handle_t *h, char *name, globus_rls_obj_type_t objtype, globus_bool_t clearvalues)
- globus_result_t **globus_rls_client_lrc_attr_get** (globus_rls_handle_t *h, char *name, globus_rls_obj_type_t objtype, globus_list_t **attr_list)
- globus_result_t **globus_rls_client_lrc_attr_modify** (globus_rls_handle_t *h, char *key, globus_rls_attribute_t *attr)
- globus_result_t **globus_rls_client_lrc_attr_remove** (globus_rls_handle_t *h, char *key, globus_rls_attribute_t *attr)
- globus_result_t **globus_rls_client_lrc_attr_remove_bulk** (globus_rls_handle_t *h, globus_list_t *attr_obj_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_attr_search** (globus_rls_handle_t *h, char *name, globus_rls_obj_type_t objtype, globus_rls_attr_op_t op, globus_rls_attribute_t *operand1, globus_rls_attribute_t *operand2, int *offset, int reslimit, globus_list_t **attr_obj_list)
- globus_result_t **globus_rls_client_lrc_attr_value_get** (globus_rls_handle_t *h, char *key, char *name, globus_rls_obj_type_t objtype, globus_list_t **attr_list)
- globus_result_t **globus_rls_client_lrc_attr_value_get_bulk** (globus_rls_handle_t *h, globus_list_t *keylist, char *name, globus_rls_obj_type_t objtype, globus_list_t **attr_obj_list)
- globus_result_t **globus_rls_client_lrc_add** (globus_rls_handle_t *h, char *lfn, char *pfn)
- globus_result_t **globus_rls_client_lrc_add_bulk** (globus_rls_handle_t *h, globus_list_t *str2_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_clear** (globus_rls_handle_t *h)
- globus_result_t **globus_rls_client_lrc_create** (globus_rls_handle_t *h, char *lfn, char *pfn)
- globus_result_t **globus_rls_client_lrc_create_bulk** (globus_rls_handle_t *h, globus_list_t *str2_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_delete** (globus_rls_handle_t *h, char *lfn, char *pfn)
- globus_result_t **globus_rls_client_lrc_delete_bulk** (globus_rls_handle_t *h, globus_list_t *str2_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_exists** (globus_rls_handle_t *h, char *key, globus_rls_obj_type_t objtype)
- globus_result_t **globus_rls_client_lrc_exists_bulk** (globus_rls_handle_t *h, globus_list_t *keylist, globus_rls_obj_type_t objtype, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_get_lfn** (globus_rls_handle_t *h, char *pfn, int *offset, int reslimit, globus_list_t **str2_list)

- globus_result_t **globus_rls_client_lrc_get_lfn_bulk** (globus_rls_handle_t *h, globus_list_t *pfnlst, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_get_lfn_wc** (globus_rls_handle_t *h, char *pfn_pattern, **globus_rls_pattern_t** type, int *offset, int reslimit, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_lrc_get_pfn** (globus_rls_handle_t *h, char *lfn, int *offset, int reslimit, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_lrc_get_pfn_bulk** (globus_rls_handle_t *h, globus_list_t *lfnlist, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_get_pfn_wc** (globus_rls_handle_t *h, char *lfn_pattern, **globus_rls_pattern_t** type, int *offset, int reslimit, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_lrc_mapping_exists** (globus_rls_handle_t *h, char *lfn, char *pfn)
- globus_result_t **globus_rls_client_lrc_renamelfn** (globus_rls_handle_t *h, char *oldname, char *newname)
- globus_result_t **globus_rls_client_lrc_renamelfn_bulk** (globus_rls_handle_t *h, globus_list_t *str2_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_renamepfn** (globus_rls_handle_t *h, char *oldname, char *newname)
- globus_result_t **globus_rls_client_lrc_renamepfn_bulk** (globus_rls_handle_t *h, globus_list_t *str2_list, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_lrc_rli_add** (globus_rls_handle_t *h, char *rli_url, int flags, char *pattern)
- globus_result_t **globus_rls_client_lrc_rli_delete** (globus_rls_handle_t *h, char *rli_url, char *pattern)
- globus_result_t **globus_rls_client_lrc_rli_get_part** (globus_rls_handle_t *h, char *rli_url, char *pattern, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_lrc_rli_info** (globus_rls_handle_t *h, char *rli_url, **globus_rls_rli_info_t** *info)
- globus_result_t **globus_rls_client_lrc_rli_list** (globus_rls_handle_t *h, globus_list_t **rliinfo_list)

4.6.1 Detailed Description

Functions to view and update data managed by a LRC server.

4.6.2 Macro Definition Documentation

4.6.2.1 #define FRLI_BLOOMFILTER 0x1

Update RLI using bloom filters (see **globus_rls_client_lrc_rli_add()** (p. 31)).

4.6.2.2 #define MAXURL 256

Maximum length of URL string.

4.6.3 Function Documentation

4.6.3.1 globus_result_t globus_rls_client_lrc_attr_add (globus_rls_handle_t * h, char * key, globus_rls_attribute_t * attr)

Add an attribute to an object in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	Logical or Physical File Name (LFN or PFN) that identifies object attribute should be added to.
<i>attr</i>	Attribute to be added to object. name, objtype, type and val (p. 38) should be set in <i>attr</i> .

Return values

<i>GLOBUS_SUCCESS</i>	Attribute successfully associated with object.
-----------------------	--

References `globus_rls_client_attr2s()`, `GLOBUS_RLS_INV_ATTR_TYPE`, `globus_rls_attribute_t::name`, `globus_rls_attribute_t::objtype`, and `globus_rls_attribute_t::type`.

4.6.3.2 `globus_result_t globus_rls_client_lrc_attr_add_bulk (globus_rls_handle_t * h, globus_list_t * attr_obj_list, globus_list_t ** str2bulk_list)`

Bulk add attributes to objects in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>attr_obj_list</i>	List of object names (LFN or PFN) and attributes to be added. Each list datum should be of type globus_rls_attribute_object_t (p. 38).
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the object name, str2.s2 the attribute name, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.3 `globus_result_t globus_rls_client_lrc_attr_create (globus_rls_handle_t * h, char * name, globus_rls_obj_type_t objtype, globus_rls_attr_type_t type)`

Define new attribute in LRC database.

Parameters

<i>h</i>	Handle connected to an LRC server.
<i>name</i>	Name of attribute.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>type</i>	Type of attribute value.

Return values

<i>GLOBUS_SUCCESS</i>	Attribute successfully created.
-----------------------	---------------------------------

4.6.3.4 `globus_result_t globus_rls_client_lrc_attr_delete (globus_rls_handle_t * h, char * name, globus_rls_obj_type_t objtype, globus_bool_t clearvalues)`

Undefine attribute in LRC database, previously created with **globus_rls_client_lrc_attr_create()** (p. 21).

Parameters

<i>h</i>	Handle connected to an LRC server.
<i>name</i>	Name of attribute.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>clearvalues</i>	If <code>GLOBUS_TRUE</code> then any any values for this attribute are first removed from the objects they're associated with. If <code>GLOBUS_FALSE</code> and any values exist then GLOBUS_RLS_ATTR_EXIST (p. 5) is returned.

Return values

<i>GLOBAL_SUCCESS</i>	Attribute successfully removed.
-----------------------	---------------------------------

4.6.3.5 `globus_result_t globus_rls_client_lrc_attr_get (globus_rls_handle_t * h, char * name, globus_rls_obj_type_t objtype, globus_list_t ** attr_list)`

Return definitions of attributes in LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>name</i>	Name of attribute. If name is NULL all attributes of the specified <i>objtype</i> are returned.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>attr_list</i>	Any attribute definitions found will be returned as a list of globus_rls_attribute_t (p.38) structures.

Return values

<i>GLOBAL_SUCCESS</i>	Attribute definitions successfully retrieved. <i>attr_list</i> should be freed with globus_rls_client_free_list() (p.13) when it is no longer needed.
-----------------------	--

References `globus_rls_client_free_list()`, `GLOBAL_RLS_NOMEMORY`, `globus_rls_attribute_t::name`, `globus_rls_attribute_t::objtype`, and `globus_rls_attribute_t::type`.

4.6.3.6 `globus_result_t globus_rls_client_lrc_attr_modify (globus_rls_handle_t * h, char * key, globus_rls_attribute_t * attr)`

Modify an attribute value.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	Name of object (LFN or PFN).
<i>attr</i>	Attribute to be modified. The objtype (p.38), name (p.38) and type (p.38) fields should be set in <i>attr</i> to identify the attribute, the val (p.38) field should be the new value.

Return values

<i>GLOBAL_SUCCESS</i>	Attribute successfully modified.
-----------------------	----------------------------------

References `globus_rls_client_attr2s()`, `globus_rls_attribute_t::name`, `globus_rls_attribute_t::objtype`, and `globus_rls_attribute_t::type`.

4.6.3.7 `globus_result_t globus_rls_client_lrc_attr_remove (globus_rls_handle_t * h, char * key, globus_rls_attribute_t * attr)`

Remove an attribute from an object (LFN or PFN) in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	Name of object (LFN or PFN).
<i>attr</i>	Attribute to be removed. The objtype (p.38) and name (p.38) fields should be set in <i>attr</i> to identify the attribute.

Return values

<i>GLOBUS_SUCCESS</i>	Attribute successfully removed.
-----------------------	---------------------------------

References `globus_rls_attribute_t::name`, and `globus_rls_attribute_t::objtype`.

4.6.3.8 `globus_result_t globus_rls_client_lrc_attr_remove_bulk (globus_rls_handle_t * h, globus_list_t * attr_obj_list, globus_list_t ** str2bulk_list)`

Bulk remove attributes from objects in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>attr_obj_list</i>	List of object names (LFN or PFN) and attributes to be removed. It is not necessary to set the attribute type or value. Each list datum should be of type globus_rls_attribute_object_t (p. 38).
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the object name, str2.s2 the attribute name, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.9 `globus_result_t globus_rls_client_lrc_attr_search (globus_rls_handle_t * h, char * name, globus_rls_obj_type_t objtype, globus_rls_attr_op_t op, globus_rls_attribute_t * operand1, globus_rls_attribute_t * operand2, int * offset, int reslimit, globus_list_t ** attr_obj_list)`

Search for objects (LFNs or PFNs) in a LRC database that have the specified attribute whose value matches a boolean expression.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>name</i>	Name of attribute.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>op</i>	Operator to be used in searching for values.
<i>operand1</i>	First operand in boolean expression. type (p. 38) and val (p. 38) should be set in globus_rls_attribute_t (p. 38).
<i>operand2</i>	Second operand in boolean expression, only used when <i>op</i> is <code>globus_rls_client_attr_op_btw</code> . type (p. 38) and val (p. 38) should be set in globus_rls_attribute_t (p. 38).
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result. If NULL then API will handle accumulating partial results transparently.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>attr_obj_list</i>	Any objects with the specified attribute will be returned, with the attribute value, in a list of globus_rls_attribute_object_t (p. 38) structures.

Return values

<i>GLOBUS_SUCCESS</i>	Objects with specified attribute returned in <i>attr_obj_list</i> . <i>attr_obj_list</i> should be freed with globus_rls_client_free_list() (p. 13) when it is no longer needed. See Query Results (p. 13).
-----------------------	---

References `globus_rls_client_attr2s()`, `globus_rls_client_free_list()`, `GLOBUS_RLS_INV_ATTR_TYPE`, and `GLOBUS_RLS_NOMEMORY`.

4.6.3.10 `globus_result_t globus_rls_client_lrc_attr_value_get (globus_rls_handle_t * h, char * key, char * name, globus_rls_obj_type_t objtype, globus_list_t ** attr_list)`

Return attributes in LRC database for specified object (LFN or PFN).

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	Logical or Physical File Name (LFN or PFN) that identifies object attributes should be retrieved for.
<i>name</i>	Name of attribute to retrieve. If NULL all attributes for <i>key</i> , <i>objtype</i> are returned.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>attr_list</i>	Any attributes found will be returned in this list of globus_rls_attribute_t (p. 38) structures.

Return values

<i>GLOBUS_SUCCESS</i>	Attributes successfully retrieved. <i>attr_list</i> should be freed with globus_rls_client_free_list() (p. 13) when it is no longer needed.
-----------------------	--

References `globus_rls_client_free_list()`, `globus_rls_client_s2attr()`, `GLOBUS_RLS_NOMEMORY`, `GLOBUS_RLS_SUCCESS`, `globus_rls_attribute_t::name`, and `globus_rls_attribute_t::objtype`.

4.6.3.11 `globus_result_t globus_rls_client_lrc_attr_value_get_bulk (globus_rls_handle_t * h, globus_list_t * keylist, char * name, globus_rls_obj_type_t objtype, globus_list_t ** attr_obj_list)`

Return attributes in LRC database for specified objects (LFN or PFN).

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>keylist</i>	Logical or Physical File Names (LFNs or PFNs) that identify object attributes should be retrieved for. Each list datum should be a string containing the LFN or PFN.
<i>name</i>	Name of attribute to retrieve. If NULL all attributes for <i>key</i> , <i>objtype</i> are returned.
<i>objtype</i>	Object (LFN or PFN) type that attribute applies to.
<i>attr_obj_list</i>	Any attributes found will be returned in this list of globus_rls_attribute_object_t (p. 38) structures.

Return values

<i>GLOBUS_SUCCESS</i>	Attributes successfully retrieved. <i>attr_obj_list</i> should be freed with globus_rls_client_free_list() (p. 13) when it is no longer needed.
-----------------------	--

References `globus_rls_handle_t::flags`, `GLOBUS_RLS_BADARG`, `globus_rls_client_free_list()`, `GLOBUS_RLS_NOMEMORY`, `GLOBUS_RLS_SUCCESS`, `globus_rls_handle_t::handle`, and `MAXERRMSG`.

4.6.3.12 `globus_result_t globus_rls_client_lrc_add (globus_rls_handle_t * h, char * lfn, char * pfn)`

Add mapping to PFN to an existing LFN.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN to add <i>pfn</i> mapping to, should already exist.
<i>pfn</i>	PFN that <i>lfn</i> should map to.

Return values

<i>GLOBUS_SUCCESS</i>	New mapping created.
-----------------------	----------------------

References `GLOBUS_RLS_BADARG`.

4.6.3.13 `globus_result_t globus_rls_client_lrc_add_bulk (globus_rls_handle_t * h, globus_list_t * str2_list, globus_list_t ** str2bulk_list)`

Bulk add LFN,PFN mappings in LRC database.

LFNs must already exist.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>str2_list</i>	LFN,PFN pairs to add mappings.
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_qls_string2_bulk_t (p.41) structure. str2.s1 will be the LFN, and str2.s2 the PFN it maps to, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.14 globus_result_t globus_qls_client_lrc_clear (globus_qls_handle_t * h)

Clear all mappings from LRC database.

User needs both ADMIN and LRCUPDATE privileges to perform this operation. Note that if the LRC is cleared this will not be reflected in any RLI servers updated by the LRC until the next softstate update, even if immediate updates are enabled.

Parameters

<i>h</i>	Handle connected to an RLS server.
----------	------------------------------------

Return values

GLOBUS_SUCCESS	Mappings cleared.
-----------------------	-------------------

4.6.3.15 globus_result_t globus_qls_client_lrc_create (globus_qls_handle_t * h, char * lfn, char * pfn)

Create mapping between a LFN and PFN.

LFN should not exist yet.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN to add <i>pfn</i> mapping to, should not already exist.
<i>pfn</i>	PFN that <i>lfn</i> should map to.

Return values

GLOBUS_SUCCESS	New mapping created.
-----------------------	----------------------

References GLOBUS_RLS_BADARG.

4.6.3.16 globus_result_t globus_qls_client_lrc_create_bulk (globus_qls_handle_t * h, globus_list_t * str2_list, globus_list_t ** str2bulk_list)

Bulk create LFN,PFN mappings in LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>str2_list</i>	LFN,PFN pairs to create mappings for.
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_qls_string2_bulk_t (p.41) structure. str2.s1 will be the LFN, and str2.s2 the PFN it maps to, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.17 globus_result_t globus_qls_client_lrc_delete (globus_qls_handle_t * h, char * lfn, char * pfn)

Delete mapping between LFN and PFN.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN to remove mapping from.
<i>pfn</i>	PFN that <i>lfn</i> maps to that is being removed.

Return values

<i>GLOBUS_SUCCESS</i>	Mapping removed.
-----------------------	------------------

References GLOBUS_RLS_BADARG.

4.6.3.18 `globus_result_t globus_rls_client_lrc_delete_bulk (globus_rls_handle_t * h, globus_list_t * str2_list, globus_list_t ** str2bulk_list)`

Bulk delete LFN,PFN mappings in LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>str2_list</i>	LFN,PFN pairs to add mappings.
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_rls_string2_bulk_t (p.41) structure. str2.s1 will be the LFN, and str2.s2 the PFN it maps to, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.19 `globus_result_t globus_rls_client_lrc_exists (globus_rls_handle_t * h, char * key, globus_rls_obj_type_t objtype)`

Check if an object exists in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	LFN or PFN that identifies object.
<i>objtype</i>	Type of object <i>key</i> refers to (globus_rls_obj_lrc_lfn (p.9) or globus_rls_obj_lrc_pfn (p.9)).

Return values

<i>GLOBUS_SUCCESS</i>	Object exists.
-----------------------	----------------

References GLOBUS_RLS_BADARG.

4.6.3.20 `globus_result_t globus_rls_client_lrc_exists_bulk (globus_rls_handle_t * h, globus_list_t * keylist, globus_rls_obj_type_t objtype, globus_list_t ** str2bulk_list)`

Bulk check if objects exist in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>keylist</i>	LFNs or PFNs that identify objects.
<i>objtype</i>	Type of object <i>key</i> refers to (globus_rls_obj_lrc_lfn (p.9) or globus_rls_obj_lrc_pfn (p.9)).
<i>str2bulk_list</i>	Results of existence check. Each list datum will be a globus_rls_string2_bulk_t (p.41) structure. str2.s1 will be the LFN or PFN, and str2.s2 empty, and rc will be the result code indicating existence.

4.6.3.21 `globus_result_t globus_rls_client_lrc_get_lfn (globus_rls_handle_t * h, char * pfn, int * offset, int reslimit, globus_list_t ** str2_list)`

Return LFNs mapped to PFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>pfn</i>	PFN to search for.
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result. If NULL then API will handle accumulating partial results transparently.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of LFNs that map to <i>pfn</i> . Each list datum will be a globus_rls_string2_t (p. 41) structure. s1 will be the LFN, and s2 the PFN it maps to.

Return values

GLOBUS_SUCCESS	List of LFNs that map to <i>pfn</i> in <i>str2_list</i> . See Query Results (p. 13).
-----------------------	---

4.6.3.22 **globus_result_t** globus_rls_client_lrc_get_lfn_bulk (**globus_rls_handle_t** * *h*, **globus_list_t** * *pfnlist*, **globus_list_t** ** *str2bulk_list*)

Bulk return LFNs mapped to PFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>pfnlist</i>	PFNs to search for.
<i>str2bulk_list</i>	Results of queries. Each list datum will be a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the LFN, and str2.s2 the PFN it maps to, and rc will be the result code from the query.

4.6.3.23 **globus_result_t** globus_rls_client_lrc_get_lfn_wc (**globus_rls_handle_t** * *h*, **char** * *pfn_pattern*, **globus_rls_pattern_t** *type*, **int** * *offset*, **int** *reslimit*, **globus_list_t** ** *str2_list*)

Return LFNs mapped to wildcarded PFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>pfn_pattern</i>	PFN pattern to search for.
<i>type</i>	Identifies wildcard characters used in <i>pfn_pattern</i> . Wildcard chars can be Unix file globbing chars (* matches 0 or more characters, ? matches any single character) or SQL "like" wildcard characters (% matches 0 or more characters, _ matches any single character).
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result. If NULL then the API will handle accumulating partial results transparently.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of LFNs that map to <i>pfn_pattern</i> . Each list datum will be a globus_rls_string2_t (p. 41) structure. s1 will be the LFN, and s2 the PFN it maps to.

Return values

GLOBUS_SUCCESS	List of LFNs that map to <i>pfn_pattern</i> in <i>str2_list</i> . See Query Results (p. 13).
-----------------------	---

4.6.3.24 **globus_result_t** globus_rls_client_lrc_get_pfn (**globus_rls_handle_t** * *h*, **char** * *lfn*, **int** * *offset*, **int** *reslimit*, **globus_list_t** ** *str2_list*)

Return PFNs mapped to LFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN to search for.
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of PFNs that map to <i>lfn</i> . Each list datum will be a globus_rls_string2_t (p. 41) structure. s1 will be the LFN, and s2 the PFN it maps to.

Return values

GLOBUS_SUCCESS	List of PFNs that map to <i>lfn</i> in <i>str2_list</i> .
-----------------------	---

4.6.3.25 **globus_result_t** globus_rls_client_lrc_get_pfn_bulk (**globus_rls_handle_t** * *h*, **globus_list_t** * *lfnlist*, **globus_list_t** ** *str2bulk_list*)

Bulk return PFNs mapped to LFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfnlist</i>	LFNs to search for.
<i>str2bulk_list</i>	Results of queries. Each list datum will be a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the LFN, and str2.s2 the PFN it maps to, and rc will be the result code from the query.

4.6.3.26 **globus_result_t** globus_rls_client_lrc_get_pfn_wc (**globus_rls_handle_t** * *h*, **char** * *lfn_pattern*, **globus_rls_pattern_t** *type*, **int** * *offset*, **int** *reslimit*, **globus_list_t** ** *str2_list*)

Return PFNs mapped to wildcarded LFN in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn_pattern</i>	LFN pattern to search for.
<i>type</i>	Identifies wildcard characters used in <i>lfn_pattern</i> . Wildcard chars can be Unix file globbing chars (* matches 0 or more characters, ? matches any single character) or SQL "like" wildcard characters (% matches 0 or more characters, _ matches any single character).
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of PFNs that map to <i>lfn_pattern</i> . Each list datum will be a globus_rls_string2_t (p. 41) structure. s1 will be the LFN, and s2 the PFN it maps to.

Return values

GLOBUS_SUCCESS	List of PFNs that map to <i>lfn_pattern</i> in <i>str2_list</i> . See Query Results (p. 13).
-----------------------	---

4.6.3.27 **globus_result_t** globus_rls_client_lrc_mapping_exists (**globus_rls_handle_t** * *h*, **char** * *lfn*, **char** * *pfn*)

Check if a mapping exists in the LRC database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN of mapping.
<i>pfm</i>	PFN of mapping.

Return values

<i>GLOBUS_SUCCESS</i>	Object exists.
-----------------------	----------------

References GLOBUS_RLS_BADARG.

4.6.3.28 `globus_result_t globus_rls_client_lrc_renamelfn (globus_rls_handle_t * h, char * oldname, char * newname)`

Rename LFN.

If immediate mode is ON, the LRC will send update messages to associated RLIs.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>oldname</i>	Existing LFN name, to be renamed.
<i>newname</i>	New LFN name, to replace existing name.

Return values

<i>GLOBUS_SUCCESS</i>	LFN renamed.
-----------------------	--------------

References GLOBUS_RLS_BADARG.

4.6.3.29 `globus_result_t globus_rls_client_lrc_renamelfn_bulk (globus_rls_handle_t * h, globus_list_t * str2_list, globus_list_t ** str2bulk_list)`

Bulk rename LFN names in LRC database.

LFNs must already exist. If immediate mode is ON, the LRC will send update messages to associated RLIs.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>str2_list</i>	oldname,newname pairs such that newname replaces oldname for LFNs.
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the old LFN name, and str2.s2 the new LFN name, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.30 `globus_result_t globus_rls_client_lrc_renamepfm (globus_rls_handle_t * h, char * oldname, char * newname)`

Rename PFN.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>oldname</i>	Existing PFN name, to be renamed.
<i>newname</i>	New PFN name, to replace existing name.

Return values

<i>GLOBUS_SUCCESS</i>	PFN renamed.
-----------------------	--------------

References GLOBUS_RLS_BADARG.

4.6.3.31 `globus_result_t globus_rls_client_lrc_renamepfns_bulk (globus_rls_handle_t * h, globus_list_t * str2_list, globus_list_t ** str2bulk_list)`

Bulk rename PFN names in LRC database.

PFNs must already exist.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>str2_list</i>	oldname,newname pairs such that newname replaces oldname for PFNs.
<i>str2bulk_list</i>	List of failed updates. Each list datum is a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the old PFN name, and str2.s2 the new PFN name, and rc will be the result code from the failed update. Only failed updates will be returned.

4.6.3.32 `globus_result_t globus_rls_client_lrc_rli_add (globus_rls_handle_t * h, char * rli_url, int flags, char * pattern)`

LRC servers send information about LFNs in their database to the the list of RLI servers in the database, added with the following function.

Updates may be partitioned amongst multiple RLIs by specifying one or more patterns for an RLI.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	URL of RLI server that LRC should send updates to.
<i>flags</i>	Should be zero or FRLI_BLOOMFILTER (p. 20).
<i>pattern</i>	If not NULL used to filter which LFNs are sent to <i>rli_url</i> . Standard Unix wildcard characters (*, ?) may be used to do wildcard matches.

Return values

<i>GLOBUS_SUCCESS</i>	RLI (with pattern if not NULL) added to LRC database.
-----------------------	---

References GLOBUS_RLS_BADARG.

4.6.3.33 `globus_result_t globus_rls_client_lrc_rli_delete (globus_rls_handle_t * h, char * rli_url, char * pattern)`

Delete an entry from the LRC rli/partition tables.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	URL of RLI server to remove from LRC partition table.
<i>pattern</i>	If not NULL then only the specific <i>rli_url/pattern</i> is removed, else all partition information for <i>rli_url</i> is removed.

Return values

<i>GLOBUS_SUCCESS</i>	RLI and pattern (if specified) removed from LRC partition table.
-----------------------	--

References GLOBUS_RLS_BADARG.

4.6.3.34 `globus_result_t globus_rls_client_lrc_rli_get_part (globus_rls_handle_t * h, char * rli_url, char * pattern, globus_list_t ** str2_list)`

Get RLI update partitions from LRC server.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	If not NULL identifies RLI that partition data will be retrieved for. If NULL then all RLIs are retrieved.
<i>pattern</i>	If not NULL returns only partitions with matching patterns, otherwise all patterns are retrieved.
<i>str2_list</i>	Results added to list. Datums in <i>str2_list</i> are of type globus_rls_string2_t (p. 41) structure. s1 will be the rli url, s2 an empty string or the pattern used to partition updates. See Query Results (p. 13).

Return values

<i>GLOBUS_SUCCESS</i>	Partition data retrieved from server, written to <i>str2_list</i> .
-----------------------	---

References globus_rls_client_free_list(), and GLOBUS_RLS_NOMEMORY.

4.6.3.35 globus_result_t globus_rls_client_lrc_rli_info (globus_rls_handle_t * *h*, char * *rli_url*, globus_rls_rli_info_t * *info*)

Get info about RLI server updated by an LRC server.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	URL of RLI server to retrieve info for.
<i>info</i>	Data about RLI server will be written here.

Return values

<i>GLOBUS_SUCCESS</i>	Info about RLI server successfully retrieved.
-----------------------	---

References globus_rls_rli_info_t::flags, GLOBUS_RLS_BADARG, globus_rls_rli_info_t::lastupdate, MAXURL, globus_rls_rli_info_t::updateinterval, and globus_rls_rli_info_t::url.

4.6.3.36 globus_result_t globus_rls_client_lrc_rli_list (globus_rls_handle_t * *h*, globus_list_t ** *rliinfo_list*)

Return URLs of RLIs that LRC sends updates to.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rliinfo_list</i>	List of RLIs updated by this LRC returned in this list. Each list datum is of type globus_rls_rli_info_t (p. 40). <i>rliinfo_list</i> should be freed with globus_rls_client_free_list() (p. 13) when no longer needed.

Return values

<i>GLOBUS_SUCCESS</i>	List of RLIs updated by this LRC returned in <i>rliinfo_list</i> .
-----------------------	--

References globus_rls_rli_info_t::flags, globus_rls_client_free_list(), GLOBUS_RLS_NOMEMORY, globus_rls_rli_info_t::lastupdate, MAXURL, globus_rls_rli_info_t::updateinterval, and globus_rls_rli_info_t::url.

4.7 RLI Operations

Data Structures

- struct **globus_rls_sender_info_t**

Functions

- globus_result_t **globus_rls_client_rli_exists** (globus_rls_handle_t *h, char *key, globus_rls_obj_type_t objtype)
- globus_result_t **globus_rls_client_rli_exists_bulk** (globus_rls_handle_t *h, globus_list_t *keylist, globus_rls_obj_type_t objtype, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_rli_get_lrc** (globus_rls_handle_t *h, char *lfn, int *offset, int reslimit, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_rli_get_lrc_bulk** (globus_rls_handle_t *h, globus_list_t *lfnlist, globus_list_t **str2bulk_list)
- globus_result_t **globus_rls_client_rli_get_lrc_wc** (globus_rls_handle_t *h, char *lfn_pattern, globus_rls_pattern_t type, int *offset, int reslimit, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_rli_sender_list** (globus_rls_handle_t *h, globus_list_t **senderinfo_list)
- globus_result_t **globus_rls_client_rli_lrc_list** (globus_rls_handle_t *h, globus_list_t **lrcinfo_list)
- globus_result_t **globus_rls_client_rli_mapping_exists** (globus_rls_handle_t *h, char *lfn, char *lrc)
- globus_result_t **globus_rls_client_rli_rli_add** (globus_rls_handle_t *h, char *rli_url, char *pattern)
- globus_result_t **globus_rls_client_rli_rli_delete** (globus_rls_handle_t *h, char *rli_url, char *pattern)
- globus_result_t **globus_rls_client_rli_rli_get_part** (globus_rls_handle_t *h, char *rli_url, char *pattern, globus_list_t **str2_list)
- globus_result_t **globus_rls_client_rli_rli_list** (globus_rls_handle_t *h, globus_list_t **rliinfo_list)

4.7.1 Detailed Description

Functions to view and update data managed by a RLI server.

4.7.2 Function Documentation

4.7.2.1 globus_result_t globus_rls_client_rli_exists (globus_rls_handle_t * h, char * key, globus_rls_obj_type_t objtype)

Check if an object exists in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>key</i>	LFN or LRC that identifies object.
<i>objtype</i>	Type of object <i>key</i> refers to (globus_rls_obj_rli_lfn (p. 9) or globus_rls_obj_rli_lrc (p. 9)).

Return values

<i>GLOBUS_SUCCESS</i>	Object exists.
-----------------------	----------------

References GLOBUS_RLS_BADARG.

4.7.2.2 globus_result_t globus_rls_client_rli_exists_bulk (globus_rls_handle_t * h, globus_list_t * keylist, globus_rls_obj_type_t objtype, globus_list_t ** str2bulk_list)

Bulk check if objects exist in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>keylist</i>	LFNs or LRCs that identify objects.
<i>objtype</i>	Type of object <i>key</i> refers to (globus_rls_obj_rli_lfn (p. 9) or globus_rls_obj_rli_lrc (p. 9)).
<i>str2bulk_list</i>	Results of existence check. Each list datum will be a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the LFN or LRC, and str2.s2 empty, and rc will be the result code indicating existence.

4.7.2.3 **globus_result_t** globus_rls_client_rli_get_lrc (**globus_rls_handle_t** * *h*, char * *lfn*, int * *offset*, int *reslimit*, **globus_list_t** ** *str2_list*)

Return LRCs mapped to LFN in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN whose list of LRCs is desired.
<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of LRCs that <i>lfn</i> maps to. Each list datum will be a globus_rls_string2_t (p. 41) structure. s1 will be the LFN, and s2 the LRC it maps to. <i>str2_list</i> should be freed with globus_rls_client_free_list() (p. 13).

Return values

GLOBUS_SUCCESS	List of LRCs that map to <i>lfn</i> in <i>str2_list</i> . See Query Results (p. 13).
-----------------------	---

4.7.2.4 **globus_result_t** globus_rls_client_rli_get_lrc_bulk (**globus_rls_handle_t** * *h*, **globus_list_t** * *lfnlist*, **globus_list_t** ** *str2bulk_list*)

Bulk return LRCs mapped to LFN in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfnlist</i>	LFNs to search for.
<i>str2bulk_list</i>	Results of queries. Each list datum will be a globus_rls_string2_bulk_t (p. 41) structure. str2.s1 will be the LFN, and str2.s2 the LRC it maps to, and rc will be the result code from the query.

4.7.2.5 **globus_result_t** globus_rls_client_rli_get_lrc_wc (**globus_rls_handle_t** * *h*, char * *lfn_pattern*, **globus_rls_pattern_t** *type*, int * *offset*, int *reslimit*, **globus_list_t** ** *str2_list*)

Return LRCs mapped to wildcarded LFN in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn_pattern</i>	LFN pattern to search for.
<i>type</i>	Identifies wildcard characters used in <i>lfn_pattern</i> . Wildcard chars can be Unix file globbing chars (* matches 0 or more characters, ? matches any single character) or SQL "like" wildcard characters (% matches 0 or more charactes, _ matches any single character).

<i>offset</i>	Offset into result list. Used in conjunction with <i>reslimit</i> to retrieve results a few at a time. Use 0 to begin with first result.
<i>reslimit</i>	Maximum number of results to return. Used in conjunction with <i>offset</i> to retrieve results a few at a time. Use 0 to retrieve all results.
<i>str2_list</i>	List of LRCs that map to <i>lfn_pattern</i> . Each list datum will be a globus_rls_string2_t (p. 41). s1 will be the LFN, and s2 the LRC it maps to. <i>str2_list</i> should be freed with globus_rls_client_free_list() (p. 13).

Return values

GLOBAL_SUCCESS	List of LRCs that map to <i>lfn_pattern</i> in <i>str2_list</i> . See Query Results (p. 13).
-----------------------	---

4.7.2.6 globus_result_t globus_rls_client_rli_sender_list (globus_rls_handle_t * h, globus_list_t ** senderinfo_list)

Get list of servers updating this RLI server.

Similar to `globus_rls_client_rli_get_part()` except no partition information is returned.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>senderinfo_list</i>	Datums in <i>senderinfo_list</i> will be of type globus_rls_sender_info_t (p. 40). <i>senderinfo_list</i> should be freed with globus_rls_client_free_list() (p. 13).

Return values

GLOBAL_SUCCESS	List of LRCs updating RLI added to <i>senderinfo_list</i> .
-----------------------	---

References `globus_rls_client_free_list()`, `GLOBAL_RLS_NOMEMORY`, `globus_rls_sender_info_t::lastupdate`, `MAXURL`, and `globus_rls_sender_info_t::url`.

4.7.2.7 globus_result_t globus_rls_client_rli_lrc_list (globus_rls_handle_t * h, globus_list_t ** lrcinfo_list)

Deprecated, use **globus_rls_client_rli_sender_list()** (p. 35).

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lrcinfo_list</i>	Datums in <i>lrcinfo_list</i> will be of type <code>globus_rls_lrc_info_t</code> . <i>lrcinfo_list</i> should be freed with globus_rls_client_free_list() (p. 13).

Return values

GLOBAL_SUCCESS	List of LRCs updating RLI added to <i>lrcinfo_list</i> .
-----------------------	--

References `globus_rls_client_rli_sender_list()`.

4.7.2.8 globus_result_t globus_rls_client_rli_mapping_exists (globus_rls_handle_t * h, char * lfn, char * lrc)

Check if a mapping exists in the RLI database.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>lfn</i>	LFN of mapping.
<i>lrc</i>	LRC of mapping.

Return values

<i>GLOBUS_SUCCESS</i>	Mapping exists.
-----------------------	-----------------

References GLOBUS_RLS_BADARG.

4.7.2.9 `globus_result_t globus_rls_client_rli_rli_add (globus_rls_handle_t * h, char * rli_url, char * pattern)`

RLI servers send information about LFNs in their database to the the list of RLI servers in the database, added with the following function.

Updates may be partitioned amongst multiple RLIs by specifying one or more patterns for an RLI.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	URL of RLI server that LRC should send updates to.
<i>pattern</i>	If not NULL used to filter which LFNs are sent to <i>rli_url</i> . Standard Unix wildcard characters (*, ?) may be used to do wildcard matches.

Return values

<i>GLOBUS_SUCCESS</i>	RLI (with pattern if not NULL) added to RLI database.
-----------------------	---

References GLOBUS_RLS_BADARG.

4.7.2.10 `globus_result_t globus_rls_client_rli_rli_delete (globus_rls_handle_t * h, char * rli_url, char * pattern)`

Delete an entry from the RLI rli/partition tables.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	URL of RLI server to remove from RLI partition table.
<i>pattern</i>	If not NULL then only the specific <i>rli_url/pattern</i> is removed, else all partition information for <i>rli_url</i> is removed.

Return values

<i>GLOBUS_SUCCESS</i>	RLI and pattern (if specified) removed from LRC partition table.
-----------------------	--

References GLOBUS_RLS_BADARG.

4.7.2.11 `globus_result_t globus_rls_client_rli_rli_get_part (globus_rls_handle_t * h, char * rli_url, char * pattern, globus_list_t ** str2_list)`

Get RLI update partitions from RLI server.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rli_url</i>	If not NULL identifies RLI that partition data will be retrieved for. If NULL then all RLIs are retrieved.
<i>pattern</i>	If not NULL returns only partitions with matching patterns, otherwise all patterns are retrieved.
<i>str2_list</i>	Results added to list. Datums in <i>str2_list</i> are of type globus_rls_string2_t (p. 41) structure. s1 will be the rli url, s2 an empty string or the pattern used to partition updates. See Query Results (p. 13).

Return values

<i>GLOBUS_SUCCESS</i>	Partition data retrieved from server, written to <i>str2_list</i> .
-----------------------	---

References `globus_rls_client_free_list()`, and `GLOBUS_RLS_NOMEMORY`.

4.7.2.12 `globus_result_t globus_rls_client_rli_rli_list (globus_rls_handle_t * h, globus_list_t ** rliinfo_list)`

Return URLs of RLIs that RLI sends updates to.

Parameters

<i>h</i>	Handle connected to an RLS server.
<i>rliinfo_list</i>	List of RLIs updated by this RLI returned in this list. Each list datum is of type globus_rls_rli_info_t (p. 40). <i>rliinfo_list</i> should be freed with globus_rls_client_free_list() (p. 13) when no longer needed.

Return values

<i>GLOBUS_SUCCESS</i>	List of RLIs updated by this LRC returned in <i>rliinfo_list</i> .
-----------------------	--

References `globus_rls_rli_info_t::flags`, `globus_rls_client_free_list()`, `GLOBUS_RLS_NOMEMORY`, `globus_rls_rli_info_t::lastupdate`, `MAXURL`, `globus_rls_rli_info_t::updateinterval`, and `globus_rls_rli_info_t::url`.

5 Data Structure Documentation

5.1 globus_rls_attribute_object_t Struct Reference

Data Fields

- **globus_rls_attribute_t attr**
- **char * key**
- **int rc**

5.1.1 Detailed Description

globus_rls_client_lrc_attr_search() (p. 23) returns a list of these structures which include the object name (LFN or PFN) and attribute value found by the query.

5.1.2 Field Documentation

5.1.2.1 globus_rls_attribute_t globus_rls_attribute_object_t::attr

Attribute value.

5.1.2.2 char* globus_rls_attribute_object_t::key

LFN or PFN.

5.1.2.3 int globus_rls_attribute_object_t::rc

Result code, only used in bulk query.

5.2 globus_rls_attribute_t Struct Reference

Data Fields

- **char * name**
- **globus_rls_obj_type_t objtype**
- **globus_rls_attr_type_t type**
- union {
 - time_t **t**
 - double **d**
 - int **i**
 - char * **s**
- } val**

5.2.1 Detailed Description

Object (LFN or PFN) attribute type.

5.2.2 Field Documentation

5.2.2.1 `char* globus_rls_attribute_t::name`

Attribute name.

5.2.2.2 `globus_rls_obj_type_t globus_rls_attribute_t::objtype`

Object type.

5.2.2.3 `globus_rls_attr_type_t globus_rls_attribute_t::type`

Attribute value type.

5.2.2.4 `time_t globus_rls_attribute_t::t`

Date value (unix time).

5.2.2.5 `double globus_rls_attribute_t::d`

Floating point value.

5.2.2.6 `int globus_rls_attribute_t::i`

Integer value.

5.2.2.7 `char* globus_rls_attribute_t::s`

String value.

5.2.2.8 `union { ... } globus_rls_attribute_t::val`

Value of attribute (depends on type).

5.3 `globus_rls_handle_t` Struct Reference

Data Fields

- `globus_url_t url`
- `globus_io_handle_t handle`
- `int flags`

5.3.1 Detailed Description

RLS Client Handle.

5.3.2 Field Documentation

5.3.2.1 `globus_url_t globus_rls_handle_t::url`

URL of RLS server (RLS://host[:port]).

5.3.2.2 `globus_io_handle_t globus_rls_handle_t::handle`

Globus IO handle.

5.3.2.3 int globus_rls_handle_t::flags

See FH_xxx flags below.

5.4 globus_rls_rli_info_t Struct Reference

Data Fields

- char **url** [256]
- int **updateinterval**
- int **flags**
- time_t **lastupdate**

5.4.1 Detailed Description

Information about RLI server, returned by **globus_rls_client_lrc_rli_info()** (p. 32) and **globus_rls_client_lrc_rli_list()** (p. 32).

5.4.2 Field Documentation

5.4.2.1 char globus_rls_rli_info_t::url[256]

URL of server.

5.4.2.2 int globus_rls_rli_info_t::updateinterval

Interval between softstate updates.

5.4.2.3 int globus_rls_rli_info_t::flags

RLI flags (see **FRLI_BLOOMFILTER** (p. 20)).

5.4.2.4 time_t globus_rls_rli_info_t::lastupdate

Time of last softstate update.

5.5 globus_rls_sender_info_t Struct Reference

Data Fields

- char **url** [256]
- time_t **lastupdate**

5.5.1 Detailed Description

Information about server sending updates to an rli, returned by **globus_rls_client_rli_sender_list()** (p. 35).

5.5.2 Field Documentation

5.5.2.1 char globus_rls_sender_info_t::url[256]

URL of server.

5.5.2.2 `time_t globus_rls_sender_info_t::lastupdate`

Time of last softstate update.

5.6 `globus_rls_stats_t` Struct Reference

Data Fields

- `int flags`

5.6.1 Detailed Description

Various configuration options and statistics about an RLS server returned in the following structures by **`globus_rls_client_stats()`** (p. 10).

See **`RLS_LRCSERVER`** (p. 8) for possible flags values.

5.6.2 Field Documentation

5.6.2.1 `int globus_rls_stats_t::flags`

See **`RLS_LRCSERVER`** (p. 8).

5.7 `globus_rls_string2_bulk_t` Struct Reference

5.7.1 Detailed Description

String pair result with return code, returned by bulk query operations.

5.8 `globus_rls_string2_t` Struct Reference

Data Fields

- `char * s1`
- `char * s2`

5.8.1 Detailed Description

String pair result.

Many of the query functions use this to return pairs of strings (eg LFN,PFN or LFN,LRC).

5.8.2 Field Documentation

5.8.2.1 `char* globus_rls_string2_t::s1`

First string in pair (eg LFN).

5.8.2.2 `char* globus_rls_string2_t::s2`

Second string in pair (eg PFN or LRC).

Index

- Activation, 15
 - GLOBUS_RLS_CLIENT_MODULE, 15
 - globus_rls_client_module, 15
- attr
 - globus_rls_attribute_object_t, 38
- Connection Management, 16
 - GLOBUS_RLS_SERVER_DEFPORT, 16
 - GLOBUS_RLS_URL_SCHEME, 16
 - GLOBUS_RLS_URL_SCHEME_NOAUTH, 16
 - globus_rls_client_certificate, 16
 - globus_rls_client_close, 17
 - globus_rls_client_connect, 17
 - globus_rls_client_get_timeout, 17
 - globus_rls_client_proxy_certificate, 17
 - globus_rls_client_set_timeout, 17
 - MAXERRMSG, 16
- d
 - globus_rls_attribute_t, 39
- FRLI_BLOOMFILTER
 - LRC Operations, 20
- flags
 - globus_rls_handle_t, 39
 - globus_rls_rli_info_t, 40
 - globus_rls_stats_t, 41
- GLOBUS_RLS_ATTR_EXIST
 - Status Codes, 5
- GLOBUS_RLS_ATTR_INUSE
 - Status Codes, 6
- GLOBUS_RLS_ATTR_NEXIST
 - Status Codes, 5
- GLOBUS_RLS_ATTR_VALUE_NEXIST
 - Status Codes, 5
- GLOBUS_RLS_BADARG
 - Status Codes, 4
- GLOBUS_RLS_BADMETHOD
 - Status Codes, 4
- GLOBUS_RLS_BADURL
 - Status Codes, 4
- GLOBUS_RLS_CLIENT_MODULE
 - Activation, 15
- GLOBUS_RLS_DBERROR
 - Status Codes, 5
- GLOBUS_RLS_GLOBUSERR
 - Status Codes, 4
- GLOBUS_RLS_INV_ATTR_OP
 - Status Codes, 5
- GLOBUS_RLS_INV_ATTR_TYPE
 - Status Codes, 5
- GLOBUS_RLS_INV_OBJ_TYPE
 - Status Codes, 5
- GLOBUS_RLS_INVHANDLE
 - Status Codes, 4
- GLOBUS_RLS_INVSERVER
 - Status Codes, 4
- GLOBUS_RLS_LFN_EXIST
 - Status Codes, 4
- GLOBUS_RLS_LFN_NEXIST
 - Status Codes, 4
- GLOBUS_RLS_LRC_EXIST
 - Status Codes, 5
- GLOBUS_RLS_LRC_NEXIST
 - Status Codes, 5
- GLOBUS_RLS_MAPPING_EXIST
 - Status Codes, 5
- GLOBUS_RLS_MAPPING_NEXIST
 - Status Codes, 4
- GLOBUS_RLS_NOMEMORY
 - Status Codes, 4
- GLOBUS_RLS_OVERFLOW
 - Status Codes, 4
- GLOBUS_RLS_PERM
 - Status Codes, 4
- GLOBUS_RLS_PFN_EXIST
 - Status Codes, 4
- GLOBUS_RLS_PFN_NEXIST
 - Status Codes, 4
- GLOBUS_RLS_RLI_EXIST
 - Status Codes, 5
- GLOBUS_RLS_RLI_NEXIST
 - Status Codes, 5
- GLOBUS_RLS_SERVER_DEFPORT
 - Connection Management, 16
- GLOBUS_RLS_SUCCESS
 - Status Codes, 4
- GLOBUS_RLS_TIMEOUT
 - Status Codes, 5
- GLOBUS_RLS_TOO_MANY_CONNECTIONS
 - Status Codes, 5
- GLOBUS_RLS_UNSUPPORTED
 - Status Codes, 5
- GLOBUS_RLS_URL_SCHEME
 - Connection Management, 16
- GLOBUS_RLS_URL_SCHEME_NOAUTH
 - Connection Management, 16
- globus_list_len
 - Miscellaneous, 12
- globus_rls_admin_cmd_ping
 - Miscellaneous, 9
- globus_rls_admin_cmd_quit
 - Miscellaneous, 9
- globus_rls_admin_cmd_ssu

- Miscellaneous, 9
- globus_rls_admin_cmd_t
 - Miscellaneous, 9
- globus_rls_attr_op_all
 - Miscellaneous, 9
- globus_rls_attr_op_btw
 - Miscellaneous, 9
- globus_rls_attr_op_eq
 - Miscellaneous, 9
- globus_rls_attr_op_ge
 - Miscellaneous, 9
- globus_rls_attr_op_gt
 - Miscellaneous, 9
- globus_rls_attr_op_le
 - Miscellaneous, 9
- globus_rls_attr_op_like
 - Miscellaneous, 9
- globus_rls_attr_op_lt
 - Miscellaneous, 9
- globus_rls_attr_op_ne
 - Miscellaneous, 9
- globus_rls_attr_op_t
 - Miscellaneous, 9
- globus_rls_attr_type_date
 - Miscellaneous, 9
- globus_rls_attr_typeflt
 - Miscellaneous, 9
- globus_rls_attr_type_int
 - Miscellaneous, 9
- globus_rls_attr_type_str
 - Miscellaneous, 9
- globus_rls_attr_type_t
 - Miscellaneous, 8
- globus_rls_attribute_object_t, 38
 - attr, 38
 - key, 38
 - rc, 38
- globus_rls_attribute_t, 38
 - d, 39
 - i, 39
 - name, 38
 - objtype, 39
 - s, 39
 - t, 39
 - type, 39
 - val, 39
- globus_rls_client_admin
 - Miscellaneous, 9
- globus_rls_client_attr2s
 - Miscellaneous, 11
- globus_rls_client_certificate
 - Connection Management, 16
- globus_rls_client_close
 - Connection Management, 17
- globus_rls_client_connect
 - Connection Management, 17

- globus_rls_client_error_info
 - Miscellaneous, 11
- globus_rls_client_free_list
 - Query Results, 13
- globus_rls_client_get_configuration
 - Miscellaneous, 10
- globus_rls_client_get_timeout
 - Connection Management, 17
- globus_rls_client_lrc_add
 - LRC Operations, 24
- globus_rls_client_lrc_add_bulk
 - LRC Operations, 24
- globus_rls_client_lrc_attr_add
 - LRC Operations, 20
- globus_rls_client_lrc_attr_add_bulk
 - LRC Operations, 21
- globus_rls_client_lrc_attr_create
 - LRC Operations, 21
- globus_rls_client_lrc_attr_delete
 - LRC Operations, 21
- globus_rls_client_lrc_attr_get
 - LRC Operations, 22
- globus_rls_client_lrc_attr_modify
 - LRC Operations, 22
- globus_rls_client_lrc_attr_remove
 - LRC Operations, 22
- globus_rls_client_lrc_attr_remove_bulk
 - LRC Operations, 23
- globus_rls_client_lrc_attr_search
 - LRC Operations, 23
- globus_rls_client_lrc_attr_value_get
 - LRC Operations, 23
- globus_rls_client_lrc_attr_value_get_bulk
 - LRC Operations, 24
- globus_rls_client_lrc_clear
 - LRC Operations, 26
- globus_rls_client_lrc_create
 - LRC Operations, 26
- globus_rls_client_lrc_create_bulk
 - LRC Operations, 26
- globus_rls_client_lrc_delete
 - LRC Operations, 26
- globus_rls_client_lrc_delete_bulk
 - LRC Operations, 27
- globus_rls_client_lrc_exists
 - LRC Operations, 27
- globus_rls_client_lrc_exists_bulk
 - LRC Operations, 27
- globus_rls_client_lrc_get_lfn
 - LRC Operations, 27
- globus_rls_client_lrc_get_lfn_bulk
 - LRC Operations, 28
- globus_rls_client_lrc_get_lfn_wc
 - LRC Operations, 28
- globus_rls_client_lrc_get_pfn
 - LRC Operations, 28

- globus_rls_client_lrc_get_pfn_bulk
 - LRC Operations, 29
- globus_rls_client_lrc_get_pfn_wc
 - LRC Operations, 29
- globus_rls_client_lrc_mapping_exists
 - LRC Operations, 29
- globus_rls_client_lrc_renamelfn
 - LRC Operations, 30
- globus_rls_client_lrc_renamelfn_bulk
 - LRC Operations, 30
- globus_rls_client_lrc_renamepfn
 - LRC Operations, 30
- globus_rls_client_lrc_renamepfn_bulk
 - LRC Operations, 30
- globus_rls_client_lrc_rli_add
 - LRC Operations, 31
- globus_rls_client_lrc_rli_delete
 - LRC Operations, 31
- globus_rls_client_lrc_rli_get_part
 - LRC Operations, 31
- globus_rls_client_lrc_rli_info
 - LRC Operations, 32
- globus_rls_client_lrc_rli_list
 - LRC Operations, 32
- globus_rls_client_module
 - Activation, 15
- globus_rls_client_proxy_certificate
 - Connection Management, 17
- globus_rls_client_rli_exists
 - RLI Operations, 33
- globus_rls_client_rli_exists_bulk
 - RLI Operations, 33
- globus_rls_client_rli_get_lrc
 - RLI Operations, 34
- globus_rls_client_rli_get_lrc_bulk
 - RLI Operations, 34
- globus_rls_client_rli_get_lrc_wc
 - RLI Operations, 34
- globus_rls_client_rli_lrc_list
 - RLI Operations, 35
- globus_rls_client_rli_mapping_exists
 - RLI Operations, 35
- globus_rls_client_rli_rli_add
 - RLI Operations, 36
- globus_rls_client_rli_rli_delete
 - RLI Operations, 36
- globus_rls_client_rli_rli_get_part
 - RLI Operations, 36
- globus_rls_client_rli_rli_list
 - RLI Operations, 37
- globus_rls_client_rli_sender_list
 - RLI Operations, 35
- globus_rls_client_s2attr
 - Miscellaneous, 11
- globus_rls_client_set_configuration
 - Miscellaneous, 10
- globus_rls_client_set_timeout
 - Connection Management, 17
- globus_rls_client_stats
 - Miscellaneous, 10
- globus_rls_errmsg
 - Miscellaneous, 12
- globus_rls_handle_t, 39
 - flags, 39
 - handle, 39
 - url, 39
- globus_rls_obj_lrc_lfn
 - Miscellaneous, 9
- globus_rls_obj_lrc_pfn
 - Miscellaneous, 9
- globus_rls_obj_rli_lfn
 - Miscellaneous, 9
- globus_rls_obj_rli_lrc
 - Miscellaneous, 9
- globus_rls_obj_type_t
 - Miscellaneous, 9
- globus_rls_pattern_t
 - Miscellaneous, 8
- globus_rls_rli_info_t, 40
 - flags, 40
 - lastupdate, 40
 - updateinterval, 40
 - url, 40
- globus_rls_sender_info_t, 40
 - lastupdate, 40
 - url, 40
- globus_rls_stats_t, 41
 - flags, 41
- globus_rls_string2_bulk_t, 41
- globus_rls_string2_t, 41
 - s1, 41
 - s2, 41
- handle
 - globus_rls_handle_t, 39
- i
 - globus_rls_attribute_t, 39
- key
 - globus_rls_attribute_object_t, 38
- LRC Operations, 19
 - FRLI_BLOOMFILTER, 20
 - globus_rls_client_lrc_add, 24
 - globus_rls_client_lrc_add_bulk, 24
 - globus_rls_client_lrc_attr_add, 20
 - globus_rls_client_lrc_attr_add_bulk, 21
 - globus_rls_client_lrc_attr_create, 21
 - globus_rls_client_lrc_attr_delete, 21
 - globus_rls_client_lrc_attr_get, 22
 - globus_rls_client_lrc_attr_modify, 22

globus_rls_client_lrc_attr_remove, 22	globus_rls_attr_type_str, 9
globus_rls_client_lrc_attr_remove_bulk, 23	globus_rls_attr_type_t, 8
globus_rls_client_lrc_attr_search, 23	globus_rls_client_admin, 9
globus_rls_client_lrc_attr_value_get, 23	globus_rls_client_attr2s, 11
globus_rls_client_lrc_attr_value_get_bulk, 24	globus_rls_client_error_info, 11
globus_rls_client_lrc_clear, 26	globus_rls_client_get_configuration, 10
globus_rls_client_lrc_create, 26	globus_rls_client_s2attr, 11
globus_rls_client_lrc_create_bulk, 26	globus_rls_client_set_configuration, 10
globus_rls_client_lrc_delete, 26	globus_rls_client_stats, 10
globus_rls_client_lrc_delete_bulk, 27	globus_rls_errmsg, 12
globus_rls_client_lrc_exists, 27	globus_rls_obj_lrc_lfn, 9
globus_rls_client_lrc_exists_bulk, 27	globus_rls_obj_lrc_pfn, 9
globus_rls_client_lrc_get_lfn, 27	globus_rls_obj_rli_lfn, 9
globus_rls_client_lrc_get_lfn_bulk, 28	globus_rls_obj_rli_lrc, 9
globus_rls_client_lrc_get_lfn_wc, 28	globus_rls_obj_type_t, 9
globus_rls_client_lrc_get_pfn, 28	globus_rls_pattern_t, 8
globus_rls_client_lrc_get_pfn_bulk, 29	RLS_INITIALIZED, 8
globus_rls_client_lrc_get_pfn_wc, 29	RLS_LRCSERVER, 8
globus_rls_client_lrc_mapping_exists, 29	RLS_RCVBLOOMFILTER, 8
globus_rls_client_lrc_renamelfn, 30	RLS_RCVLFNLIST, 8
globus_rls_client_lrc_renamelfn_bulk, 30	RLS_RLISERVER, 8
globus_rls_client_lrc_renamepfn, 30	RLS_SNDBLOOMFILTER, 8
globus_rls_client_lrc_renamepfn_bulk, 30	RLS_SNDLFNLIST, 8
globus_rls_client_lrc_rli_add, 31	rls_pattern_sql, 8
globus_rls_client_lrc_rli_delete, 31	rls_pattern_unix, 8
globus_rls_client_lrc_rli_get_part, 31	
globus_rls_client_lrc_rli_info, 32	name
globus_rls_client_lrc_rli_list, 32	globus_rls_attribute_t, 38
MAXURL, 20	
lastupdate	objtype
globus_rls_rli_info_t, 40	globus_rls_attribute_t, 39
globus_rls_sender_info_t, 40	
	Query Results, 13
MAXERRMSG	globus_rls_client_free_list, 13
Connection Management, 16	
MAXURL	RLI Operations, 33
LRC Operations, 20	globus_rls_client_rli_exists, 33
Miscellaneous, 7	globus_rls_client_rli_exists_bulk, 33
globus_list_len, 12	globus_rls_client_rli_get_lrc, 34
globus_rls_admin_cmd_ping, 9	globus_rls_client_rli_get_lrc_bulk, 34
globus_rls_admin_cmd_quit, 9	globus_rls_client_rli_get_lrc_wc, 34
globus_rls_admin_cmd_ssu, 9	globus_rls_client_rli_lrc_list, 35
globus_rls_admin_cmd_t, 9	globus_rls_client_rli_mapping_exists, 35
globus_rls_attr_op_all, 9	globus_rls_client_rli_rli_add, 36
globus_rls_attr_op_btw, 9	globus_rls_client_rli_rli_delete, 36
globus_rls_attr_op_eq, 9	globus_rls_client_rli_rli_get_part, 36
globus_rls_attr_op_ge, 9	globus_rls_client_rli_rli_list, 37
globus_rls_attr_op_gt, 9	globus_rls_client_rli_sender_list, 35
globus_rls_attr_op_le, 9	RLS_INITIALIZED
globus_rls_attr_op_like, 9	Miscellaneous, 8
globus_rls_attr_op_lt, 9	RLS_LRCSERVER
globus_rls_attr_op_ne, 9	Miscellaneous, 8
globus_rls_attr_op_t, 9	RLS_RCVBLOOMFILTER
globus_rls_attr_type_date, 9	Miscellaneous, 8
globus_rls_attr_typeflt, 9	RLS_RCVLFNLIST
globus_rls_attr_typeint, 9	Miscellaneous, 8

RLS_RLISERVER		updateinterval	
Miscellaneous, 8		globus_rls_rli_info_t, 40	
RLS_SNDBLOOMFILTER		url	
Miscellaneous, 8		globus_rls_handle_t, 39	
RLS_SNDLFNLIST		globus_rls_rli_info_t, 40	
Miscellaneous, 8		globus_rls_sender_info_t, 40	
rc		val	
globus_rls_attribute_object_t, 38		globus_rls_attribute_t, 39	
rls_pattern_sql			
Miscellaneous, 8			
rls_pattern_unix			
Miscellaneous, 8			
s			
globus_rls_attribute_t, 39			
s1			
globus_rls_string2_t, 41			
s2			
globus_rls_string2_t, 41			
Status Codes, 3			
GLOBUS_RLS_ATTR_EXIST, 5			
GLOBUS_RLS_ATTR_INUSE, 6			
GLOBUS_RLS_ATTR_NEXIST, 5			
GLOBUS_RLS_ATTR_VALUE_NEXIST, 5			
GLOBUS_RLS_BADARG, 4			
GLOBUS_RLS_BADMETHOD, 4			
GLOBUS_RLS_BADURL, 4			
GLOBUS_RLS_DBERROR, 5			
GLOBUS_RLS_GLOBUSERR, 4			
GLOBUS_RLS_INV_ATTR_OP, 5			
GLOBUS_RLS_INV_ATTR_TYPE, 5			
GLOBUS_RLS_INV_OBJ_TYPE, 5			
GLOBUS_RLS_INVHANDLE, 4			
GLOBUS_RLS_INVSERVER, 4			
GLOBUS_RLS_LFN_EXIST, 4			
GLOBUS_RLS_LFN_NEXIST, 4			
GLOBUS_RLS_LRC_EXIST, 5			
GLOBUS_RLS_LRC_NEXIST, 5			
GLOBUS_RLS_MAPPING_EXIST, 5			
GLOBUS_RLS_MAPPING_NEXIST, 4			
GLOBUS_RLS_NOMEMORY, 4			
GLOBUS_RLS_OVERFLOW, 4			
GLOBUS_RLS_PERM, 4			
GLOBUS_RLS_PFN_EXIST, 4			
GLOBUS_RLS_PFN_NEXIST, 4			
GLOBUS_RLS_RLI_EXIST, 5			
GLOBUS_RLS_RLI_NEXIST, 5			
GLOBUS_RLS_SUCCESS, 4			
GLOBUS_RLS_TIMEOUT, 5			
GLOBUS_RLS_TOO_MANY_CONNECTIONS, 5			
GLOBUS_RLS_UNSUPPORTED, 5			
t			
globus_rls_attribute_t, 39			
type			
globus_rls_attribute_t, 39			