# globus gsi credential

3.2

Generated by Doxygen 1.6.1

Fri Apr 23 02:27:25 2010

# Contents



# 1 Globus GSI Credential

The Globus GSI Credential library. This library contains functions that provide support for handling X.509 based PKI credentials

- Activation
- Credential Handle Management
- Credential Handle Attributes
- Credential Operations
- Credential Constants

# 2 Module Index

## 2.1 Modules

Here is a list of all modules:

# 3 Module Documentation

## 3.1 Credential Constants

### Enumerations

- enum globus_gsi_cred_error_t {
  GLOBUS_GSI_CRED_ERROR_SUCCESS = 0,
  GLOBUS_GSI_CRED_ERROR_READING_PROXY_CRED = 1,
  GLOBUS_GSI_CRED_ERROR_READING_HOST_CRED = 2,
  GLOBUS_GSI_CRED_ERROR_READING_SERVICE_CRED = 3,
  GLOBUS_GSI_CRED_ERROR_READING_CRED = 4,
  GLOBUS_GSI_CRED_ERROR_WRITING_CRED = 5,
  GLOBUS_GSI_CRED_ERROR_WRITING_PROXY_CRED = 6,
  GLOBUS_GSI_CRED_ERROR_CHECKING_PROXY = 7,
  GLOBUS_GSI_CRED_ERROR_VERIFYING_CRED = 8,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED = 9,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT = 10,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED_PRIVATE_KEY = 11,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT_CHAIN = 12,
  GLOBUS_GSI_CRED_ERROR_ERRNO = 13,
  GLOBUS_GSI_CRED_ERROR_SYSTEM_CONFIG = 14,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED_HANDLE_ATTRS = 15,
  GLOBUS_GSI_CRED_ERROR_WITH_SSL_CTX = 16,
  GLOBUS_GSI_CRED_ERROR_WITH_CALLBACK_DATA = 17,
  GLOBUS_GSI_CRED_ERROR_CREATING_ERROR_OBJ = 18,
  GLOBUS_GSI_CRED_ERROR_KEY_IS_PASS_PROTECTED = 19,
  GLOBUS_GSI_CRED_ERROR_NO_CRED_FOUND = 20,
  GLOBUS_GSI_CRED_ERROR_SUBJECT_CMP = 21,
  GLOBUS_GSI_CRED_ERROR_GETTING_SERVICE_NAME = 22,
  GLOBUS_GSI_CRED_ERROR_BAD_PARAMETER = 23,
  GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT_NAME = 24,
  GLOBUS_GSI_CRED_ERROR_LAST = 25 }
- enum globus_gsi_cred_type_t

### 3.1.1 Enumeration Type Documentation

#### 3.1.1.1 enum globus_gsi_cred_error_t

Credential Error codes.

**Enumerator:**

***GLOBUS_GSI_CRED_ERROR_SUCCESS*** Success - never used.

***GLOBUS_GSI_CRED_ERROR_READING_PROXY_CRED*** Failed to read proxy credential.

***GLOBUS_GSI_CRED_ERROR_READING_HOST_CRED*** Failed to read host credential.

***GLOBUS_GSI_CRED_ERROR_READING_SERVICE_CRED*** Failed to read service credential.

***GLOBUS_GSI_CRED_ERROR_READING_CRED*** Failed to read user credential.

***GLOBUS_GSI_CRED_ERROR_WRITING_CRED*** Failed to write credential.

***GLOBUS_GSI_CRED_ERROR_WRITING_PROXY_CRED*** Failed to write proxy credential.

***GLOBUS_GSI_CRED_ERROR_CHECKING_PROXY*** Error checking for proxy credential.

***GLOBUS_GSI_CRED_ERROR_VERIFYING_CRED*** Failed to verify credential.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED*** Invalid credential.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT*** Invalid certificate.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED_PRIVATE_KEY*** Invalid private key.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT_CHAIN*** Invalid certificate chain.

***GLOBUS_GSI_CRED_ERROR_ERRNO*** System error.

***GLOBUS_GSI_CRED_ERROR_SYSTEM_CONFIG*** A Globus GSI System Configuration call failed.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED_HANDLE_ATTRS*** Invalid credential handle attributes.

***GLOBUS_GSI_CRED_ERROR_WITH_SSL_CTX*** Faulty SSL context.

***GLOBUS_GSI_CRED_ERROR_WITH_CALLBACK_DATA*** Faulty callback data.

***GLOBUS_GSI_CRED_ERROR_CREATING_ERROR_OBJ*** Failed to aggregate errors.

***GLOBUS_GSI_CRED_ERROR_KEY_IS_PASS_PROTECTED*** Error reading private key - the key is password protected.

***GLOBUS_GSI_CRED_ERROR_NO_CRED_FOUND*** Couldn't find credential to read.

***GLOBUS_GSI_CRED_ERROR_SUBJECT_CMP*** Credential subjects do not compare.

***GLOBUS_GSI_CRED_ERROR_GETTING_SERVICE_NAME*** Unable to obtain service name from CN entry.

***GLOBUS_GSI_CRED_ERROR_BAD_PARAMETER*** Invalid function parameter.

***GLOBUS_GSI_CRED_ERROR_WITH_CRED_CERT_NAME*** Failed to process certificate subject.

***GLOBUS_GSI_CRED_ERROR_LAST*** End marker - never used.

#### 3.1.1.2 enum globus_gsi_cred_type_t

Credential Type

An enum representing a GSI Credential Type which holds info about the type of a particular credential. The three types of credential can be: GLOBUS_PROXY, GLOBUS_USER, or GLOBUS_HOST.

**See also:**

Credential Handle Management

## 3.2 Activation

Globus GSI Credential uses standard Globus module activation and deactivation.

### Defines

- #define GLOBUS_GSI_CREDENTIAL_MODULE

### 3.2.1 Detailed Description

Globus GSI Credential uses standard Globus module activation and deactivation. Before any Globus GSI Credential functions are called, the following function must be called:

```
globus_module_activate(GLOBUS_GSI_CREDENTIAL_MODULE)
```

This function returns GLOBUS_SUCCESS if Globus GSI Credential was successfully initialized, and you are therefore allowed to subsequently call Globus GSI Credential functions. Otherwise, an error code is returned, and Globus GSI Credential functions should not be subsequently called. This function may be called multiple times.

To deactivate Globus GSI Credential, the following function must be called:

```
globus_module_deactivate(GLOBUS_GSI_CREDENTIAL_MODULE)
```

This function should be called once for each time Globus GSI Credential was activated.

### 3.2.2 Define Documentation

#### 3.2.2.1 #define GLOBUS_GSI_CREDENTIAL_MODULE

Module descriptor.

## 3.3 Credential Handle Management

Create/Destroy/Modify a GSI Credential Handle.

### Typedefs

- typedef struct globus_l_gsi_cred_handle_s * globus_gsi_cred_handle_t

### Initializing and Destroying a Handle

- globus_result_t globus_gsi_cred_handle_init (globus_gsi_cred_handle_t *handle, globus_gsi_cred_handle_attrs_t handle_attrs)
- globus_result_t globus_gsi_cred_handle_destroy (globus_gsi_cred_handle_t handle)

### Copying a Handle

- globus_result_t globus_gsi_cred_handle_copy (globus_gsi_cred_handle_t source, globus_gsi_cred_handle_t *dest)

### Getting the Handle Attributes

- globus_result_t globus_gsi_cred_get_handle_attrs (globus_gsi_cred_handle_t handle, globus_gsi_cred_handle_attrs_t *attrs)

### Getting the Credential Expiration

- globus_result_t globus_gsi_cred_get_goodtill (globus_gsi_cred_handle_t cred_handle, time_t *goodtill)

### Getting the Credential Lifetime

- globus_result_t globus_gsi_cred_get_lifetime (globus_gsi_cred_handle_t cred_handle, time_t *lifetime)

### Getting the Credential Strength

- globus_result_t globus_gsi_cred_get_key_bits (globus_gsi_cred_handle_t cred_handle, int *key_bits)

### Setting and Getting the Certificate

- globus_result_t globus_gsi_cred_set_cert (globus_gsi_cred_handle_t handle, X509 *cert)
- globus_result_t globus_gsi_cred_get_cert (globus_gsi_cred_handle_t handle, X509 **cert)

### Setting and Getting the Credential Key

- globus_result_t globus_gsi_cred_set_key (globus_gsi_cred_handle_t handle, EVP_PKEY *key)
- globus_result_t globus_gsi_cred_get_key (globus_gsi_cred_handle_t handle, EVP_PKEY **key)

### Setting and Getting the Certificate Chain

- globus_result_t globus_gsi_cred_set_cert_chain (globus_gsi_cred_handle_t handle, STACK_OF(X509)*cert_chain)
- globus_result_t globus_gsi_cred_get_cert_chain (globus_gsi_cred_handle_t handle, STACK_OF(X509)**cert_chain)

### Get Cred Cert X509 Subject Name object

- globus_result_t globus_gsi_cred_get_X509_subject_name (globus_gsi_cred_handle_t handle, X509_NAME **subject_name)

### Get X509 Identity Name

- globus_result_t globus_gsi_cred_get_X509_identity_name (globus_gsi_cred_handle_t handle, X509_NAME **identity_name)

### Get Cred Cert Subject Name

- globus_result_t globus_gsi_cred_get_subject_name (globus_gsi_cred_handle_t handle, char **subject_name)

**Get Policies from Cert Chain**

- globus_result_t globus_gsi_cred_get_policies (globus_gsi_cred_handle_t handle, STACK **policies)

**Get Policy Languages from Cert Chain**

- globus_result_t globus_gsi_cred_get_policy_languages (globus_gsi_cred_handle_t handle, STACK_OF(ASN1_OBJECT)**policy_languages)

**Get Cred Cert X509 Issuer Name object**

- globus_result_t globus_gsi_cred_get_X509_issuer_name (globus_gsi_cred_handle_t handle, X509_NAME **issuer_name)

**Get Issuer Name**

- globus_result_t globus_gsi_cred_get_issuer_name (globus_gsi_cred_handle_t handle, char **issuer_name)

**Get Identity Name**

- globus_result_t globus_gsi_cred_get_identity_name (globus_gsi_cred_handle_t handle, char **identity_name)

**Credential validation functions**

- globus_result_t globus_gsi_cred_verify_cert_chain (globus_gsi_cred_handle_t cred_handle, globus_gsi_callback_data_t callback_data)
- globus_result_t globus_gsi_cred_verify (globus_gsi_cred_handle_t handle)

### 3.3.1 Detailed Description

Create/Destroy/Modify a GSI Credential Handle. Within the Globus GSI Credential Library, all credential operations require a handle parameter. Currenlty only one operation may be in progress at once per credential handle.

This section defines operations to create, modify and destroy GSI Credential handles.

### 3.3.2 Typedef Documentation

#### 3.3.2.1 typedef struct globus_l_gsi_cred_handle_s* globus_gsi_cred_handle_t

GSI Credential Handle. A GSI Credential handle keeps track of state relating to a credential. Handles can have immutable attributes associated with them. All credential operations take a credential handle pointer as a parameter.

**See also:**

globus_gsi_cred_handle_init(), globus_gsi_cred_handle_destroy(), globus_gsi_cred_handle_attrs_t

### 3.3.3 Function Documentation

#### 3.3.3.1 globus_result_t globus_gsi_cred_handle_init (globus_gsi_cred_handle_t * *handle*, globus_gsi_cred_handle_attrs_t *handle_attrs*)

Initializes a credential handle to be used credential handling functions. Takes a set of handle attributes that are immutable to the handle. The handle attributes are only pointed to by the handle, so the lifetime of the attributes needs to be as long as that of the handle.

**Parameters:**

***handle*** The handle to be initialized

***handle_attrs*** The immutable attributes of the handle

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.2 globus_result_t globus_gsi_cred_handle_destroy (globus_gsi_cred_handle_t *handle*)

Destroys the credential handle.

**Parameters:**

***handle*** The credential handle to be destroyed

**Returns:**

GLOBUS_SUCCESS

#### 3.3.3.3 globus_result_t globus_gsi_cred_handle_copy (globus_gsi_cred_handle_t *source*, globus_gsi_cred_handle_t * *dest*)

Copies a credential handle.

**Parameters:**

***source*** The handle to be copied

***dest*** The destination of the copy

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.4 globus_result_t globus_gsi_cred_get_handle_attrs (globus_gsi_cred_handle_t *handle*, globus_gsi_cred_handle_attrs_t * *attrs*)

This function retreives a copy of the credential handle attributes.

**Parameters:**

***handle*** The credential handle to retrieve the attributes from

***attrs*** Contains the credential attributes on return

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.5 globus_result_t globus_gsi_cred_get_goodtill (globus_gsi_cred_handle_t *cred_handle*, time_t * *goodtill*)

This function retreives the expiration time of the credential contained in the handle.

**Parameters:**

***cred_handle*** The credential handle to retrieve the expiration time from

***goodtill*** Contains the expiration time on return

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.6 globus_result_t globus_gsi_cred_get_lifetime (globus_gsi_cred_handle_t *cred_handle*, time_t * *lifetime*)

This function retreives the lifetime of the credential contained in a handle.

**Parameters:**

***cred_handle*** The credential handle to retrieve the lifetime from

***lifetime*** Contains the lifetime on return

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.7 globus_result_t globus_gsi_cred_get_key_bits (globus_gsi_cred_handle_t *cred_handle*, int * *key_bits*)

This function retreives the key strength of the credential contained in a handle.

**Parameters:**

***cred_handle*** The credential handle to retrieve the strength from

***key_bits*** Contains the number of bits in the key on return

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

#### 3.3.3.8 globus_result_t globus_gsi_cred_set_cert (globus_gsi_cred_handle_t *handle*, X509 * *cert*)

Set the Credential's Certificate. The X509 cert that is passed in should be a valid X509 certificate object

**Parameters:**

***handle*** The credential handle to set the certificate on

***cert*** The X509 cert to set in the cred handle. The cert passed in can be NULL which will set the cert in the handle to NULL, freeing the current cert in the handle.

**Returns:**

GLOBUS_SUCCESS or an error object id if an error

#### 3.3.3.9 globus_result_t globus_gsi_cred_get_cert (globus_gsi_cred_handle_t *handle*, X509 ** *cert*)

Get the certificate of a credential.

**Parameters:**

***handle*** The credential handle to get the certificate from

***cert*** The resulting X509 certificate, a duplicate of the certificate in the credential handle. This variable should be freed when the user is finished with it using the function X509_free.

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object id is returned

#### 3.3.3.10 globus_result_t globus_gsi_cred_set_key (globus_gsi_cred_handle_t *handle*, EVP_PKEY * *key*)

Set the private key of the credential handle.

**Parameters:**

***handle*** The handle on which to set the key.

***key*** The private key to set the handle's key to. This value can be NULL, in which case the current handle's key is freed.

#### 3.3.3.11 globus_result_t globus_gsi_cred_get_key (globus_gsi_cred_handle_t *handle*, EVP_PKEY ** *key*)

Get the credential handle's private key.

**Parameters:**

***handle*** The credential handle containing the private key to get

***key*** The private key which after this function returns is set to a duplicate of the private key of the credential handle. This variable needs to be freed by the user when it is no longer used via the function EVP_-PKEY_free.

**Returns:**

GLOBUS_SUCCESS or an error object identifier

#### 3.3.3.12 globus_result_t globus_gsi_cred_set_cert_chain (globus_gsi_cred_handle_t *handle*, STACK_OF(X509)∗ *cert_chain*)

Set the certificate chain of the credential handle.

**Parameters:**

***handle*** The handle containing the certificate chain field to set

***cert_chain*** The certificate chain to set the handle's certificate chain to

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object id is returned

#### 3.3.3.13 globus_result_t globus_gsi_cred_get_cert_chain (globus_gsi_cred_handle_t *handle*, STACK_OF(X509)∗∗ *cert_chain*)

Get the certificate chain of the credential handle.

**Parameters:**

***handle*** The credential handle containing the certificate chain to get

***cert_chain*** The certificate chain to set as a duplicate of the cert chain in the credential handle. This variable (or the variable it points to) needs to be freed when the user is finished with it using sk_X509_free.

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object id is returned

#### 3.3.3.14 globus_result_t globus_gsi_cred_get_X509_subject_name (globus_gsi_cred_handle_t *handle*, X509_NAME ∗∗ *subject_name*)

Get the credential handle's certificate subject name.

**Parameters:**

***handle*** The credential handle containing the certificate to get the subject name of

***subject_name*** The subject name as an X509_NAME object. This should be freed using X509_NAME_free when the user is finished with it

**Returns:**

GLOBUS_SUCCESS if no error, a error object id otherwise

#### 3.3.3.15 globus_result_t globus_gsi_cred_get_X509_identity_name (globus_gsi_cred_handle_t *handle*, X509_NAME ** *identity_name*)

Get the identity's X509 subject name from the credential handle.

**Parameters:**

***handle*** The credential handle containing the certificate to get the identity from

***identity_name*** The identity certificate's X509 subject name

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object identifier is returned

#### 3.3.3.16 globus_result_t globus_gsi_cred_get_subject_name (globus_gsi_cred_handle_t *handle*, char ** *subject_name*)

Get the credential handle's certificate subject name.

**Parameters:**

***handle*** The credential handle containing the certificate to get the subject name of

***subject_name*** The subject name as a string. This should be freed using free() when the user is finished with it

**Returns:**

GLOBUS_SUCCESS if no error, a error object id otherwise

#### 3.3.3.17 globus_result_t globus_gsi_cred_get_policies (globus_gsi_cred_handle_t *handle*, STACK ** *policies*)

Get the Policies from the Cert Chain in the handle. The policies will be null-terminated as they are added to the handle. If a policy for a cert in the chain doesn't exist, the string in the stack will be set to the static string GLOBUS_NULL_POLICIES

**Parameters:**

***handle*** the handle to get the cert chain containing the policies

***policies*** the stack of policies retrieved from the handle's cert chain

**Returns:**

GLOBUS_SUCCESS or an error object if an error occurred

#### 3.3.3.18 globus_result_t globus_gsi_cred_get_policy_languages (globus_gsi_cred_handle_t *handle*, STACK_OF(ASN1_OBJECT)** *policy_languages*)

Get the policy languages from the cert chain in the handle.

**Parameters:**

***handle*** the handle to get the cert chain containing the policies

***policy_languages*** the stack of policies retrieved from the handle's cert chain

**Returns:**

GLOBUS_SUCCESS or an error object if an error occurred

#### 3.3.3.19 globus_result_t globus_gsi_cred_get_X509_issuer_name (globus_gsi_cred_handle_t *handle*, X509_NAME ** *issuer_name*)

Get the credential handle's certificate issuer name.

**Parameters:**

***handle*** The credential handle containing the certificate to get the issuer name of

***issuer_name*** The issuer name as an X509_NAME object. This should be freed using X509_NAME_free when the user is finished with it

**Returns:**

GLOBUS_SUCCESS if no error, a error object id otherwise

#### 3.3.3.20 globus_result_t globus_gsi_cred_get_issuer_name (globus_gsi_cred_handle_t *handle*, char ** *issuer_name*)

Get the issuer's subject name from the credential handle.

**Parameters:**

***handle*** The credential handle containing the certificate to get the issuer of

***issuer_name*** The issuer certificate's subject name

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object identifier is returned

#### 3.3.3.21 globus_result_t globus_gsi_cred_get_identity_name (globus_gsi_cred_handle_t *handle*, char ** *identity_name*)

Get the identity's subject name from the credential handle.

**Parameters:**

***handle*** The credential handle containing the certificate to get the identity of

***identity_name*** The identity certificate's subject name

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object identifier is returned

#### 3.3.3.22 globus_result_t globus_gsi_cred_verify_cert_chain (globus_gsi_cred_handle_t *cred_handle*, globus_gsi_callback_data_t *callback_data*)

This function performs path valiadtion on the certificate chain contained in the credential handle.

**Parameters:**

***cred_handle*** The credential handle containing the certificate chain to be validated

***callback_data*** A initialized callback data structure

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object identifier is returned

#### 3.3.3.23 globus_result_t globus_gsi_cred_verify (globus_gsi_cred_handle_t *handle*)

This function ensures that the certificate and private key in the credential handle match.

**Parameters:**

***handle*** The credential handle containing the certificate and key to be validated

**Returns:**

GLOBUS_SUCCESS if no error, otherwise an error object identifier is returned

## 3.4 Credential Handle Attributes

Create/Destroy/Modify GSI Credential Handle Attributes.

### Typedefs

- typedef struct globus_l_gsi_cred_handle_attrs_s * globus_gsi_cred_handle_attrs_t

### Credential Handle Attributes Initialization and Destruction

- globus_result_t globus_gsi_cred_handle_attrs_init (globus_gsi_cred_handle_attrs_t *handle_attrs)
- globus_result_t globus_gsi_cred_handle_attrs_destroy (globus_gsi_cred_handle_attrs_t handle_attrs)

### Copy Credential Handle Attributes

- globus_result_t globus_gsi_cred_handle_attrs_copy (globus_gsi_cred_handle_attrs_t source, globus_gsi_cred_handle_attrs_t *dest)

### Seting and Getting the CA Cert Dir

- globus_result_t globus_gsi_cred_handle_attrs_set_ca_cert_dir (globus_gsi_cred_handle_attrs_t handle_attrs, char *ca_cert_dir)
- globus_result_t globus_gsi_cred_handle_attrs_get_ca_cert_dir (globus_gsi_cred_handle_attrs_t handle_attrs, char **ca_cert_dir)

**Setting and Getting the Search Order**

- globus_result_t globus_gsi_cred_handle_attrs_set_search_order (globus_gsi_cred_handle_attrs_t handle_attrs, globus_gsi_cred_type_t search_order[ ])
- globus_result_t globus_gsi_cred_handle_attrs_get_search_order (globus_gsi_cred_handle_attrs_t handle_attrs, globus_gsi_cred_type_t **search_order)

### 3.4.1 Detailed Description

Create/Destroy/Modify GSI Credential Handle Attributes. Within the Globus GSI Credential Library, all credential handles contain a attribute structure, which in turn contains handle instance independent attributes.

This section defines operations to create, modify and destroy GSI Credential handle attributes.

### 3.4.2 Typedef Documentation

#### 3.4.2.1 typedef struct globus_l_gsi_cred_handle_attrs_s* globus_gsi_cred_handle_attrs_t

Credential Handle Attributes. Credential handle attributes provide a set of immutable parameters for a credential handle

**See also:**

globus_gsi_cred_handle_init

### 3.4.3 Function Documentation

#### 3.4.3.1 globus_result_t globus_gsi_cred_handle_attrs_init (globus_gsi_cred_handle_attrs_t * *handle_attrs*)

Initializes the immutable Credential Handle Attributes The handle attributes are initialized as follows:.

- The search order is set to SERVICE, HOST, PROXY, USER
- All other attributes are set to 0/NULL

**Parameters:**

***handle_attrs*** the attributes to be initialized

**Returns:**

GLOBUS_SUCESS if initialization was successful, otherwise an error is returned

#### 3.4.3.2 globus_result_t globus_gsi_cred_handle_attrs_destroy (globus_gsi_cred_handle_attrs_t *handle_attrs*)

Destroy the Credential Handle Attributes. This function does some cleanup and deallocation of the handle attributes.

**Parameters:**

***handle_attrs*** The handle attributes to destroy

**Returns:**

GLOBUS_SUCCESS

#### 3.4.3.3 globus_result_t globus_gsi_cred_handle_attrs_copy (globus_gsi_cred_handle_attrs_t *source*, globus_gsi_cred_handle_attrs_t * *dest*)

Copy the Credential Handle Attributes.

**Parameters:**

***source*** The handle attribute to be copied

***dest*** The copy

**Returns:**

GLOBUS_SUCESS unless there was an error, in which case an error object is returned.

#### 3.4.3.4 globus_result_t globus_gsi_cred_handle_attrs_set_ca_cert_dir (globus_gsi_cred_handle_attrs_t *handle_attrs*, char * *ca_cert_dir*)

Set the Trusted CA Certificate Directory Location.

**Parameters:**

***handle_attrs*** the credential handle attributes to set

***ca_cert_dir*** the trusted ca certificates directory

**Returns:**

GLOBUS_SUCCESS if no errors occurred. In case of a null handle_attrs, an error object id is returned

#### 3.4.3.5 globus_result_t globus_gsi_cred_handle_attrs_get_ca_cert_dir (globus_gsi_cred_handle_attrs_t *handle_attrs*, char ** *ca_cert_dir*)

Get the trusted ca cert directory.

**Parameters:**

***handle_attrs*** the credential handle attributes to get the trusted ca cert directory from

***ca_cert_dir*** the trusted ca certificates directory

**Returns:**

GLOBUS_SUCCESS if no errors occurred. In case of a null handle_attrs or pointer to ca_cert_dir, an error object id is returned

#### 3.4.3.6 globus_result_t globus_gsi_cred_handle_attrs_set_search_order (globus_gsi_cred_handle_attrs_t *handle_attrs*, globus_gsi_cred_type_t *search_order*[ ])

Set the search order for finding a user certificate. The default value is {SERVICE, HOST, PROXY, USER}

**Parameters:**

***handle_attrs*** The handle attributes to set the search order of

***search_order*** The search order. Should be a three element array containing in some order PROXY, USER, HOST, SERVICE. The array should be terminated by the value GLOBUS_SO_END.

**Returns:**

GLOBUS_SUCCESS unless handle_attrs is null

#### 3.4.3.7 globus_result_t globus_gsi_cred_handle_attrs_get_search_order (globus_gsi_cred_handle_attrs_t *handle_attrs*, globus_gsi_cred_type_t ** *search_order*)

Get the search order of the handle attributes.

**Parameters:**

***handle_attrs*** The handle attributes to get the search order from

***search_order*** The search_order of the handle attributes

**Returns:**

GLOBUS_SUCCESS unless handle_attrs is null

## 3.5 Credential Operations

Read/Write a GSI Credential Handle.

**Read Credential**

- globus_result_t globus_gsi_cred_read (globus_gsi_cred_handle_t handle, X509_NAME *desired_subject)

**Reading Proxy Credentials**

- globus_result_t globus_gsi_cred_read_proxy (globus_gsi_cred_handle_t handle, const char *proxy_filename)
- globus_result_t globus_gsi_cred_read_proxy_bio (globus_gsi_cred_handle_t handle, BIO *bio)

**Read Key**

- globus_result_t globus_gsi_cred_read_key (globus_gsi_cred_handle_t handle, char *key_filename, int(*pw_cb)())

**Read Cert**

- globus_result_t globus_gsi_cred_read_cert (globus_gsi_cred_handle_t handle, char *cert_filename)

**Read Cert & Key in PKCS12 Format**

- globus_result_t globus_gsi_cred_read_pkcs12 (globus_gsi_cred_handle_t handle, char *pkcs12_filename)

**Write Credential**

- globus_result_t globus_gsi_cred_write (globus_gsi_cred_handle_t handle, BIO *bio)
- globus_result_t globus_gsi_cred_write_proxy (globus_gsi_cred_handle_t handle, char *proxy_filename)

**Get the X509 certificate type (EEC, CA, proxy type, etc.)**

- globus_result_t globus_gsi_cred_get_cert_type (globus_gsi_cred_handle_t handle, globus_gsi_cert_utils_-cert_type_t *type)

### 3.5.1 Detailed Description

Read/Write a GSI Credential Handle. This section defines operations to read and write GSI Credential handles.

### 3.5.2 Function Documentation

#### 3.5.2.1 globus_result_t globus_gsi_cred_read (globus_gsi_cred_handle_t *handle*, X509_NAME * *desired_subject*)

Read a Credential from a filesystem location. The credential to read will be determined by the search order specified in the handle attributes.

**Parameters:**

***handle*** The credential handle to set. This credential handle should already be initialized using globus_gsi_-cred_handle_init.

***desired_subject*** The subject to check for when reading in a credential. The desired_subject should be either a exact match of the read cert's subject or should just contain the /CN entry. If null, the credential read in is the first match based on the system configuration (paths and environment variables)

**Returns:**

GLOBUS_SUCCESS if no errors occured, otherwise, an error object identifier is returned.

**See also:**

globus_gsi_cred_read_proxy()
globus_gsi_cred_read_cert_and_key()

**Note:**

This function always searches for the desired credential. If you don't want to perform a search, then don't use this function. The search goes in the order of the handle attributes' search order.

#### 3.5.2.2 globus_result_t globus_gsi_cred_read_proxy (globus_gsi_cred_handle_t *handle*, const char * *proxy_filename*)

Read a proxy from a PEM file.

**Parameters:**

***handle*** The credential handle to set based on the proxy credential read from the file

***proxy_filename*** The file containing the proxy credential

**Returns:**

GLOBUS_SUCCESS or an error object identifier

#### 3.5.2.3 globus_result_t globus_gsi_cred_read_proxy_bio (globus_gsi_cred_handle_t *handle*, BIO * *bio*)

Read a Proxy Credential from a BIO stream and set the credential handle to represent the read credential. The values read from the stream, in order, will be the signed certificate, the private key, and the certificate chain

**Parameters:**

***handle*** The credential handle to set. The credential should handle be initialized (i.e. not NULL).

***bio*** The stream to read the credential from

**Returns:**

GLOBUS_SUCCESS unless an error occurred, in which case an error object is returned

#### 3.5.2.4 globus_result_t globus_gsi_cred_read_key (globus_gsi_cred_handle_t *handle*, char * *key_filename*, int(*)() *pw_cb*)

Read a key from a PEM file.

**Parameters:**

***handle*** the handle to set based on the key that is read

***key_filename*** the filename of the key to read

***pw_cb*** the callback for obtaining a password for decrypting the key.

**Returns:**

GLOBUS_SUCCESS or an error object identifier

#### 3.5.2.5 globus_result_t globus_gsi_cred_read_cert (globus_gsi_cred_handle_t *handle*, char * *cert_filename*)

Read a cert from a file. Cert should be in PEM format.

**Parameters:**

***handle*** the handle to set based on the certificate that is read

***cert_filename*** the filename of the certificate to read

**Returns:**

GLOBUS_SUCCESS or an error object identifier

#### 3.5.2.6 globus_result_t globus_gsi_cred_read_pkcs12 (globus_gsi_cred_handle_t *handle*, char * *pkcs12_filename*)

Read a cert & key from a file. The file should be in PKCS12 format.

**Parameters:**

***handle*** the handle to populate with the read credential

***pkcs12_filename*** the filename containing the credential to read

**Returns:**

GLOBUS_SUCCESS or an error object identifier

#### 3.5.2.7 globus_result_t globus_gsi_cred_write (globus_gsi_cred_handle_t *handle*, BIO * *bio*)

Write out a credential to a BIO. The credential parameters written, in order, are the signed certificate, the RSA private key, and the certificate chain (a set of X509 certificates). the credential is written out in PEM format.

**Parameters:**

***handle*** The credential to write out

***bio*** The BIO stream to write out to

**Returns:**

GLOBUS_SUCCESS unless an error occurred, in which case an error object ID is returned.

#### 3.5.2.8 globus_result_t globus_gsi_cred_write_proxy (globus_gsi_cred_handle_t *handle*, char * *proxy_filename*)

Write out a credential to a file. The credential parameters written, in order, are the signed certificate, the RSA private key, and the certificate chain (a set of X509 certificates). the credential is written out in PEM format.

**Parameters:**

***handle*** The credential to write out

***proxy_filename*** The file to write out to

**Returns:**

GLOBUS_SUCCESS unless an error occurred, in which case an error object ID is returned.

#### 3.5.2.9 globus_result_t globus_gsi_cred_get_cert_type (globus_gsi_cred_handle_t *handle*, globus_gsi_cert_utils_cert_type_t * *type*)

Determine the type of the given X509 certificate For the list of possible values returned, see globus_gsi_cert_-utils_cert_type_t.

**Parameters:**

***handle*** The credential handle containing the certificate

***type*** The returned X509 certificate type

**Returns:**

GLOBUS_SUCCESS or an error captured in a globus_result_t

# Index