

globus gss assist

8.1

Generated by Doxygen 1.7.6.1

Thu Jan 26 2012 15:01:15

Contents

1	Globus GSI GSS Assist	1
2	Module Index	1
2.1	Modules	1
3	Module Documentation	1
3.1	Activation	1
3.1.1	Detailed Description	2
3.1.2	Define Documentation	2
3.2	Utility Functions	3
3.2.1	Detailed Description	4
3.2.2	Define Documentation	4
3.2.3	Function Documentation	5
3.3	GSI GSS Assist Constants	15
3.3.1	Enumeration Type Documentation	15
3.4	Security Token Transport	16
3.4.1	Detailed Description	16
3.4.2	Function Documentation	16

1 Globus GSI GSS Assist

The GSS Assist code provides convenience functions for using the Globus GSS-API.

2 Module Index

2.1 Modules

Here is a list of all modules:

Activation	1
Utility Functions	3
GSI GSS Assist Constants	15
Security Token Transport	16

3 Module Documentation

3.1 Activation

Defines

- **#define GLOBUS_GSI_GSS_ASSIST_MODULE**

3.1.1 Detailed Description

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

```
globus_module_activate(GLOBUS_GSI_GSS_ASSIST_MODULE);
```

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

To deactivate Globus GSS Assist, the following function must be called:

```
globus_module_deactivate(GLOBUS_GSI_GSS_ASSIST_MODULE)
```

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

3.1.2 Define Documentation

3.1.2.1 #define GLOBUS_GSI_GSS_ASSIST_MODULE

Module descriptor.

3.2 Utility Functions

Defines

- #define **GlobusGssAssistFreeDNArray**(dn_a)
- #define **NI_MAXHOST** 255

Functions

- int **globus_gss_assist_gridmap** (char *globusidp, char **useridp)
- int **globus_gss_assist_userok** (char *globusid, char *userid)
- int **globus_gss_assist_map_local_user** (char *local_user, char **globusidp)
- globus_result_t **globus_gss_assist_lookup_all_globusid** (char *username, char **dns[], int *dn_count)
- globus_result_t **globus_gss_assist_map_and_authorize** (gss_ctx_id_t context, char *service, char *desired_identity, char *identity_buffer, unsigned int identity_buffer_length)

Accept Security Context

- OM_uint32 **globus_gss_assist_accept_sec_context** (OM_uint32 *minor_status, gss_ctx_id_t *context_handle, const gss_cred_id_t cred_handle, char **src_name_char, OM_uint32 *ret_flags, int *user_to_user_flag, int *token_status, gss_cred_id_t *delegated_cred_handle, int(*gss_assist_get_token)(void *, void **, size_t *), void *gss_assist_get_context, int(*gss_assist_send_token)(void *, void *, size_t), void *gss_assist_send_context)

Accept Security Context Asynchronous

- OM_uint32 **globus_gss_assist_accept_sec_context_async** (OM_uint32 *minor_status, gss_ctx_id_t *context_handle, const gss_cred_id_t cred_handle, char **src_name_char, OM_uint32 *ret_flags, int *user_to_user_flag, void *input_buffer, size_t input_buffer_len, void **output_bufferp, size_t *output_buffer_lenp, gss_cred_id_t *delegated_cred_handle)

Acquire Credential

- OM_uint32 **globus_gss_assist_acquire_cred** (OM_uint32 *minor_status, gss_cred_usage_t cred_usage, gss_cred_id_t *output_cred_handle)

Acquire Credential Extension

- OM_uint32 **globus_gss_assist_acquire_cred_ext** (OM_uint32 *minor_status, char *desired_name_char, OM_uint32 time_req, const gss_OID_set desired_mechs, gss_cred_usage_t cred_usage, gss_cred_id_t *output_cred_handle, gss_OID_set *actual_mechs, OM_uint32 *time_rec)

Display Status

- OM_uint32 **globus_gss_assist_display_status** (FILE *fp, char *comment, OM_uint32 major_status, OM_uint32 minor_status, int token_status)

Display Status String

- OM_uint32 **globus_gss_assist_display_status_str** (char **str, char *comment, OM_uint32 major_status, OM_uint32 minor_status, int token_status)
- OM_uint32 **globus_gss_assist_import_sec_context** (OM_uint32 *minor_status, gss_ctx_id_t *context_handle, int *token_status, int fdp, FILE *fperr)

Init Security Context

- OM_uint32 **globus_gss_assist_init_sec_context** (OM_uint32 *minor_status, const gss_cred_id_t cred_handle, gss_ctx_id_t *context_handle, char *target_name_char, OM_uint32 req_flags, OM_uint32 *ret_flags, int *token_status, int(*gss_assist_get_token)(void *, void **, size_t *), void *gss_assist_get_context, int(*gss_assist_send_token)(void *, void *, size_t), void *gss_assist_send_context)

Init Security Context Async

- OM_uint32 **globus_gss_assist_init_sec_context_async** (OM_uint32 *minor_status, const gss_cred_id_t cred_handle, gss_ctx_id_t *context_handle, char *target_name_char, OM_uint32 req_flags, OM_uint32 *ret_flags, void *input_buffer, size_t input_buffer_len, void **output_bufferp, size_t *output_buffer_lenp)

Will Handle Restrictions

- OM_uint32 **globus_gss_assist_will_handle_restrictions** (OM_uint32 *minor_status, gss_ctx_id_t *context_handle)

Get Unwrap

- OM_uint32 **globus_gss_assist_get_unwrap** (OM_uint32 *minor_status, const gss_ctx_id_t context_handle, char **data, size_t *length, int *token_status, int(*gss_assist_get_token)(void *, void **, size_t *), void *gss_assist_get_context, FILE *fperr)

Wrap

- OM_uint32 **globus_gss_assist_wrap_send** (OM_uint32 *minor_status, const gss_ctx_id_t context_handle, char *data, size_t length, int *token_status, int(*gss_assist_send_token)(void *, void *, size_t), void *gss_assist_send_context, FILE *fperr)

3.2.1 Detailed Description

Utility functions for GSSAPI.

3.2.2 Define Documentation

3.2.2.1 #define GlobusGssAssistFreeDNArray(dn.a)

Free array of distinguished names.

Free the contents of a name array created during a successful call to **globus_gss_assist_lookup_all_globusid()** (p. 9)

Parameters

<i>dn_a</i>	Array of names to free.
-------------	-------------------------

Return values

<i>void</i>

3.2.2.2 #define NI_MAXHOST 255

Create a GSS Name structure from the given hostname.

This function tries to resolve the given host name string to the canonical DNS name for the host.

Parameters

<i>hostname</i>	The host name or numerical address to be resolved and transform into a GSS Name
<i>authorization_ - hostname</i>	The resulting GSS Name

Returns

GLOBUS_SUCCESS on successful completion, a error object otherwise

3.2.3 Function Documentation

3.2.3.1 `OM_uint32 globus_gss_assist_accept_sec_context (OM_uint32 * minor_status, gss_ctx_id_t * context_handle, const gss_cred_id_t cred_handle, char ** src_name_char, OM_uint32 * ret_flags, int * user_to_user_flag, int * token_status, gss_cred_id_t * delegated_cred_handle, int (*)(void *, void **, size_t *) gss_assist_get_token, void * gss_assist_get_context, int (*)(void *, void *, size_t) gss_assist_send_token, void * gss_assist_send_context)`

This routine accepts a GSSAPI security context and is called by the gram_gatekeeper.

It isolates the GSSAPI from the rest of the gram code.

Initialize a gssapi security connection. Used by the server. The context_handle is returned, and there is one for each connection. This routine will take care of the looping and token processing, using the supplied get_token and send_token routines.

Parameters

<i>minor_status</i>	gssapi return code
<i>context_handle</i>	pointer to returned context.
<i>cred_handle</i>	the cred handle obtained by acquire_cred.
<i>src_name_char</i>	Pointer to char string representation of the client which contacted the server. Maybe NULL if not wanted. Should be freed when done.
<i>ret_flags</i>	Pointer to which services are available after the connection is established. Maybe NULL if not wanted. We will also use this to pass in flags to the globus version of gssapi_ssleay
<i>user_to_user_ - flag</i>	Pointer to flag to be set if the src_name is the same as our name. (Following are particular to this assist routine)
<i>token_status</i>	assist routine get/send token status
<i>delegated_cred_ - handle</i>	pointer to be set to the credential delegated by the client if delegation occurs during the security handshake
<i>gss_assist_get_ - token</i>	a get token routine
<i>gss_assist_get_ - context</i>	first arg for the get token routine

<i>gss_assist_send_token</i>	a send token routine
<i>gss_assist_send_context</i>	first arg for the send token routine

Returns

GSS_S_COMPLETE on success Other gss errors on failure.

3.2.3.2 OM_uint32 globus_gss_assist_accept_sec_context_async (OM_uint32 * minor_status, gss_ctx_id_t * context_handle, const gss_cred_id_t cred_handle, char ** src_name_char, OM_uint32 * ret_flags, int * user_to_user_flag, void * input_buffer, size_t input_buffer_len, void ** output_bufferp, size_t * output_buffer_lenp, gss_cred_id_t * delegated_cred_handle)

This is a asynchronous version of the **globus_gss_assist_accept_sec_context()** (p. 5) function.

Instead of looping itself it passes in and out the read and written buffers and the calling application is responsible for doing the I/O directly.

Parameters

<i>minor_status</i>	gssapi return code
<i>context_handle</i>	pointer to returned context.
<i>cred_handle</i>	the cred handle obtained by acquire_cred.
<i>src_name_char</i>	Pointer to char string representation of the client which contacted the server. Maybe NULL if not wanted. Should be freed when done.
<i>ret_flags</i>	Pointer to which services are available after the connection is established. Maybe NULL if not wanted. We will also use this to pass in flags to the globus version of gssapi_ssleay
<i>user_to_user_flag</i>	Pointer to flag to be set if the src_name is the same as our name.
<i>input_buffer</i>	pointer to a buffer received from peer.
<i>input_buffer_len</i>	length of the buffer input_buffer.
<i>output_bufferp</i>	pointer to a pointer which will be filled in with a pointer to a allocated block of memory. If non-NULL the contents of this block should be written to the peer where they will be fed into the gss_assist_init_sec_context_async() function.
<i>output_buffer_lenp</i>	pointer to an integer which will be filled in with the length of the allocated output buffer pointed to by *output_bufferp.
<i>delegated_cred_handle</i>	pointer to be set to the credential delegated by the client if delegation occurs during the security handshake

Returns

GSS_S_COMPLETE on successful completion when this function does not need to be called again.

GSS_S_CONTINUE_NEEDED when *output_bufferp should be sent to the peer and a new input_buffer read and this function called again.

Other gss errors on failure.

3.2.3.3 OM_uint32 globus_gss_assist_acquire_cred (OM_uint32 * minor_status, gss_cred_usage_t cred_usage, gss_cred_id_t * output_cred_handle)

Called once at the start of the process, to obtain the credentials the process is running under.

The

Parameters

<i>minor_status</i>	pointer for return code
<i>cred_usage</i>	GSS_C_INITIATE, GSS_C_ACCEPT, or GSS_C_BOTH
<i>output_cred_handle</i>	Pointer to the returned handle. This needs to be passed to many gss routines.

Returns

GSS_S_COMPLETE on success Other GSS return codes

3.2.3.4 OM_uint32 globus_gss_assist_acquire_cred_ext (OM_uint32 * *minor_status*, char * *desired_name_char*, OM_uint32 *time_req*, const gss_OID_set *desired_mechs*, gss_cred_usage_t *cred_usage*, gss_cred_id_t * *output_cred_handle*, gss_OID_set * *actual_mechs*, OM_uint32 * *time_rec*)

Called once at the start of the process, to obtain the credentials the process is running under.

All the parameters of the gss_acquire_cred, except the desired_name is a string of the form: [type:]name. This will be imported with the type.

Returns

GSS_S_COMPLETE on success Other GSS return codes

See also

globus_gsi_gss_acquire_cred

3.2.3.5 OM_uint32 globus_gss_assist_display_status (FILE * *fp*, char * *comment*, OM_uint32 *major_status*, OM_uint32 *minor_status*, int *token_status*)

Display the messages for the major and minor status on the file pointed at by fp.

Takes care of the overloaded major_status if there was a problem with the get_token or send_token routines.

Parameters

<i>fp</i>	a file pointer
<i>comment</i>	String to print out before other error messages.
<i>major_status</i>	The major status to display
<i>minor_status</i>	The minor status to display
<i>token_status</i>	token status to display

Returns

0

3.2.3.6 OM_uint32 globus_gss_assist_display_status_str (char ** *str*, char * *comment*, OM_uint32 *major_status*, OM_uint32 *minor_status*, int *token_status*)

Display the messages for the major and minor status and return a string with the messages.

Takes care of the overloaded major_status if there was a problem with the get_token or send_token routines.

Parameters

<i>str</i>	pointer to char * for returned string. Must be freed
<i>comment</i>	String to print out before other error messages.
<i>major_status</i>	The major status to display
<i>minor_status</i>	The minor status to display
<i>token_status</i>	token status to display

Returns

0

3.2.3.7 int globus_gss_assist_gridmap (char * globusidp, char ** useridp)

Look up the default mapping for a Grid identity in a gridmap file.

The **globus_gss_assist_gridmap()** (p. 8) function parses the default gridmap file and modifies its *useridp* parameter to point to a copy of the string containing the default local identity that the grid identity is mapped to. If successful, the caller is responsible for freeing the string pointed to by *useridp*.

By default, **globus_gss_assist_gridmap()** (p. 8) looks for the default gridmap file defined by the value of the GRIDMAP environment variable. If that is not set, it falls back to \$HOME/.gridmap.

Parameters

<i>globusidp</i>	The GSSAPI name string of the identity who requested authorization
<i>useridp</i>	A pointer to a string to be set to the default user ID for the local system. No validation is done to check that such a user exists.

Returns

On success, **globus_gss_assist_gridmap()** (p. 8) returns 0 and modifies the the string pointed to by the *useridp* parameter. If an error occurs, a non-zero value is returned and the value pointed to by *useridp* is undefined.

Return values

<i>GLOBUS_SUCCESS</i>	Success
1	Error

3.2.3.8 int globus_gss_assist_userok (char * globusid, char * userid)

Gridmap entry existence check.

The **globus_gss_assist_userok()** (p. 8) function parses the default gridmap file and checks whether any mapping exists for the grid identity passed as the *globusid* parameter and the local user identity passed as the @ *userid* parameter.

By default, **globus_gss_assist_userok()** (p. 8) looks for the default gridmap file defined by the value of the GRIDMAP environment variable. If that is not set, it falls back to \$HOME/.gridmap.

Parameters

<i>globusid</i>	The GSSAPI name string of the identity who requested authorization
<i>userid</i>	The local account name that access is sought for.

Returns

If **globus_gss_assist_userok()** (p. 8) is able to find a mapping between *globusid* and *userid*, it returns 0; otherwise it returns 1.

Return values

<i>GLOBUS_SUCCESS</i>	Success
1	Error

3.2.3.9 int globus_gss_assist_map_local_user (char * *local_user*, char ** *globusidp*)

Look up the default Grid identity associated with a local user name.

The **globus_gss_assist_map_local_user()** (p. 9) function parses the gridmap file to determine if the user name passed as the *local_user* parameter is the default local user for a Grid ID in the gridmap file. If so, it modifies *globusidp* to point to a copy of that ID. Otherwise, it searches the gridmap file for a Grid ID that has a non-default mapping for *local_user* and modifies *globusidp* to point to a copy of that ID. If successful, the caller is responsible for freeing the string pointed to by the *globusidp* pointer.

By default, **globus_gss_assist_map_local_user()** (p. 9) looks for the default gridmap file defined by the value of the GRIDMAP environment variable. If that is not set, it falls back to \$HOME/.gridmap.

Parameters

<i>local_user</i>	The local username to find a Grid ID for
<i>globusidp</i>	A Grid ID that maps from the <i>local_user</i> .

Returns

On success, **globus_gss_assist_map_local_user()** (p. 9) returns 0 and modifies *globusidp* to point to a Grid ID that maps to *local_user*; otherwise, **globus_gss_assist_map_local_user()** (p. 9) returns 1 and the value pointed to by *globusidp* is undefined.

Return values

<i>GLOBUS_SUCCESS</i>	Success
1	Error

3.2.3.10 globus_result_t globus_gss_assist_lookup_all_globusid (char * *username*, char ** *dns*[], int * *dn_count*)

Look up all Grid IDs associated with a local user ID.

The **globus_gss_assist_lookup_all_globusid()** (p. 9) function parses a gridmap file and finds all Grid IDs that map to a local user ID. The *dns* parameter is modified to point to an array of Grid ID strings from the gridmap file, and the *dn_count* parameter is modified to point to the number of Grid ID strings in the array. The caller is responsible for freeing the array using the macro **GlobusGssAssistFreeDNArray()** (p. 4).

By default, **globus_gss_assist_lookup_all_globusid()** (p. 9) looks for the default gridmap file defined by the value of the GRIDMAP environment variable. If that is not set, it falls back to \$HOME/.gridmap.

Parameters

<i>username</i>	The local username to look up in the gridmap file.
<i>dns</i>	A pointer to an array of strings. This function modifies this to point to a newly allocated array of strings. The caller must use the macro GlobusGssAssistFreeDNArray() (p. 4) to free this memory.

<i>dn_count</i>	A pointer to an integer that is modified to contain the number of entries in the array returned via the <i>dns</i> parameter.
-----------------	---

Returns

On success, **globus_gss_assist_lookup_all_globusid()** (p. 9) returns GLOBUS_SUCCESS and modifies its *dns* and *dn_count* parameters as described above. If an error occurs, **globus_gss_assist_lookup_all_globusid()** (p. 9) returns a *globus_result_t* that can be resolved to an error object and the values pointed to by *dns* and *dn_count* are undefined.

Return values

<i>GLOBUS_SUCCESS</i>	Success
<i>GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_ARGUMENTS</i>	Error with arguments
<i>GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_GRIDMAP</i>	Invalid path to gridmap
<i>GLOBUS_GSI_GSS_ASSIST_ERROR_ERRNO</i>	System error

3.2.3.11 *globus_result_t globus_gss_assist_map_and_authorize(gss_ctx_id_t context, char * service, char * desired_identity, char * identity_buffer, unsigned int identity_buffer_length)*

Authorize the peer of a security context to use a service.

The **globus_gss_assist_map_and_authorize()** (p. 10) function attempts to authorize the peer of a security context to use a particular service. If the *desired_identity* parameter is non-NULL, the authorization will succeed only if the peer is authorized for that identity. Otherwise, any valid authorized local user name will be used. If authorized, the local user name will be copied to the string pointed to by the *identity_buffer* parameter, which must be at least as long as the value passed as the *identity_buffer_length* parameter.

If authorization callouts are defined in the callout configuration file, **globus_gss_assist_map_and_authorize()** (p. 10) will invoke both the GLOBUS_GENERIC_MAPPING_TYPE callout and the GLOBUS_GENERIC_AUTHZ_TYPE callout; otherwise the default gridmap file will be used for mapping and no service-specific authorization will be done.

If **globus_gss_assist_map_and_authorize()** (p. 10) uses a gridmap file, it first looks for a file defined by the value of the GRIDMAP environment variable. If that is not set, it falls back to \$HOME/.gridmap.

Parameters

<i>context</i>	Security context to inspect for peer identity information.
<i>service</i>	A NULL-terminated string containing the name of the service that an authorization decision is being made for.
<i>desired_identity</i>	Optional. If non-NULL, perform an authorization to act as the local user named by this NULL-terminated string.
<i>identity_buffer</i>	A pointer to a string buffer into which will be copied the local user name that the peer of the context is authorized to act as.
<i>identity_buffer_length</i>	Length of the <i>identity_buffer</i> array.

Returns

On success, **globus_gss_assist_map_and_authorize()** (p. 10) returns GLOBUS_SUCCESS and copies the authorized local identity to the *identity_buffer* parameter. If an error occurs, **globus_gss_assist_map_and_authorize()** (p. 10) returns a *globus_result_t* that can be resolved to an error object.

Return values

<i>GLOBUS_SUCCESS</i>	Success
<i>GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_CALLOUT_CONFIG</i>	Invalid authorization configuration file
<i>GLOBUS_CALLOUT_ERROR_WITH_HASHTABLE</i>	Hash table operation failed.
<i>GLOBUS_CALLOUT_ERROR_CALLOUT_ERROR</i>	The callout itself returned a error.
<i>GLOBUS_CALLOUT_ERROR_WITH_DL</i>	Dynamic library operation failed.
<i>GLOBUS_CALLOUT_ERROR_OUT_OF_MEMORY</i>	Out of memory
<i>GLOBUS_GSI_GSS_ASSIST_GSSAPI_ERROR</i>	A GSSAPI function returned an error
<i>GLOBUS_GSI_GSS_ASSIST_GRIDMAP_LOOKUP_FAILED</i>	Gridmap lookup failure
<i>GLOBUS_GSI_GSS_ASSIST_BUFFER_TOO_SMALL</i>	Caller provided insufficient buffer space for local identity

3.2.3.12 OM_uint32 globus_gss_assist_import_sec_context (OM_uint32 * *minor_status*, gss_ctx_id_t * *context_handle*, int * *token_status*, int *fdp*, FILE * *fperr*)

Import the security context from a file.

Parameters

<i>minor_status</i>	GSSAPI return code. This is a Globus Error code (or GLOBUS_SUCCESS) cast to a OM_uint32 pointer. If an error has occurred, the resulting error (from calling <i>globus_error_get</i> on this variable) needs to be freed by the caller
<i>context_handle</i>	The imported context
<i>token_status</i>	Errors that occurred while reading from the file
<i>fdp</i>	the file descriptor pointing to a file containing the security context
<i>fperr</i>	FILE * to write error messages

Returns

the major status

3.2.3.13 OM_uint32 globus_gss_assist_init_sec_context (OM_uint32 * *minor_status*, const gss_cred_id_t *cred_handle*, gss_ctx_id_t * *context_handle*, char * *target_name_char*, OM_uint32 *req_flags*, OM_uint32 * *ret_flags*, int * *token_status*, int (*)(void *, void **, size_t *) *gss_assist_get_token*, void * *gss_assist_get_context*, int (*)(void *, void *, size_t) *gss_assist_send_token*, void * *gss_assist_send_context*)

Initialize a gssapi security connection.

Used by the client. The *context_handle* is returned, and there is one for each connection. This routine will take care of the looping and token processing, using the supplied *get_token* and *send_token* routines.

Parameters

<i>minor_status</i>	GSSAPI return code. The new <i>minor_status</i> is a <i>globus_result_t</i> cast to an <i>OM_uint32</i> . If the call was successful, the minor status is equivalent to <i>GLOBUS_SUCCESS</i> . Otherwise, it is a globus error object ID that can be passed to <i>globus_error_get</i> to get the error object. The error object needs to be freed with <i>globus_object_free</i> .
<i>cred_handle</i>	the cred handle obtained by <i>acquire_cred</i> .
<i>context_handle</i>	pointer to returned context.
<i>target_name_</i> - <i>char</i>	char string representation of the server to be contacted.
<i>req_flags</i>	request flags, such as <i>GSS_C_DELEG_FLAG</i> for delegation and the <i>GSS_C_MUTUAL_FLAG</i> for mutual authentication.
<i>ret_flags</i>	Pointer to which services are available after the connection is established. Maybe NULL if not wanted.

The Following are particular to this assist routine:

Parameters

<i>token_status</i>	the assist routine's get/send token status
<i>gss_assist_get_</i> - <i>token</i>	function pointer for getting the token
<i>gss_assist_get_</i> - <i>context</i>	first argument passed to the <i>gss_assist_get_token</i> function
<i>gss_assist_</i> - <i>send_token</i>	function pointer for setting the token
<i>gss_assist_</i> - <i>send_context</i>	first argument passed to the <i>gss_assist_set_token</i> function pointer

Returns

The major status

3.2.3.14 *OM_uint32 globus_gss_assist_init_sec_context_async (OM_uint32 * minor_status, const gss_cred_id_t cred_handle, gss_ctx_id_t * context_handle, char * target_name_char, OM_uint32 req_flags, OM_uint32 * ret_flags, void * input_buffer, size_t input_buffer_len, void ** output_bufferp, size_t * output_buffer_lenp)*

This is a asynchronous version of the **globus_gss_assist_init_sec_context()** (p. 11) function.

Instead of looping itself it passes in and out the read and written buffers and the calling application is responsible for doing the I/O directly.

Parameters

<i>minor_status</i>	GSSAPI return code. The new <i>minor_status</i> is a <i>globus_result_t</i> cast to a <i>OM_uint32</i> . If an error occurred (<i>GSS_ERROR(major_status)</i>) the <i>minor_status</i> is a globus error object id. The error object can be obtained via <i>globus_error_get</i> and should be destroyed with <i>globus_object_free</i> when no longer needed. If no error occurred, the minor status is equal to <i>GLOBUS_SUCCESS</i> .
<i>cred_handle</i>	the cred handle obtained by <i>acquire_cred</i> .
<i>context_handle</i>	pointer to returned context.
<i>target_name_</i> - <i>char</i>	char string representation of the server to be contacted.
<i>req_flags</i>	request flags, such as <i>GSS_C_DELEG_FLAG</i> for delegation and the <i>GSS_C_MUTUAL_FLAG</i> for mutual authentication.
<i>ret_flags</i>	Pointer to which services are available after the connection is established. Maybe NULL if not wanted.

<i>input_buffer</i>	pointer to a buffer received from peer. Should be NULL on first call.
<i>input_buffer_len</i>	length of the buffer <i>input_buffer</i> . Should be zero on first call.
<i>output_bufferp</i>	pointer to a pointer which will be filled in with a pointer to a allocated block of memory. If non-NULL the contents of this block should be written to the peer where they will be fed into the <code>gss_assist_init_sec_context_async()</code> function.
<i>output_buffer_lenp</i>	pointer to an integer which will be filled in with the length of the allocated output buffer pointed to by <i>*output_bufferp</i> .

Returns

GSS_S_COMPLETE on successful completion when this function does not need to be called again.

GSS_S_CONTINUE_NEEDED when **output_bufferp* should be sent to the peer and a new *input_buffer* read and this function called again.

Other gss errors on failure.

3.2.3.15 OM_uint32 globus_gss_assist_will_handle_restrictions (OM_uint32 * *minor_status*, gss_ctx_id_t * *context_handle*)

Sets the context to handle restrictions.

Parameters

<i>minor_status</i>	the resulting minor status from setting the context handle
<i>context_handle</i>	the context handle to set the minor status of

Returns

the major status from setting the context

3.2.3.16 OM_uint32 globus_gss_assist_get_unwrap (OM_uint32 * *minor_status*, const gss_ctx_id_t *context_handle*, char ** *data*, size_t * *length*, int * *token_status*, int(*) (void *, void **, size_t *) *gss_assist_get_token*, void * *gss_assist_get_context*, FILE * *fperr*)

Gets a token using the specific tokenizing functions, and performs the GSS unwrap of that token.

See also

`gss_unwrap`

Parameters

<i>minor_status</i>	GSSAPI return code,
---------------------	---------------------

See also

`gss_unwrap`

Parameters

<i>context_handle</i>	the context
<i>data</i>	pointer to be set to the unwrapped application data. This must be freed by the caller.
<i>length</i>	pointer to be set to the length of the <i>data</i> byte array.

<i>token_status</i>	assist routine get/send token status
<i>gss_assist_get_token</i>	a detokenizing routine
<i>gss_assist_get_context</i>	first arg for above routine
<i>fperr</i>	error stream to print to

Returns

GSS_S_COMPLETE on success Other gss errors on failure.

3.2.3.17 `OM_uint32 globus_gss_assist_wrap_send (OM_uint32 * minor_status, const gss_ctx_id_t context_handle, char * data, size_t length, int * token_status, int(*) (void *, void *, size_t) gss_assist_send_token, void * gss_assist_send_context, FILE * fperr)`

Parameters

<i>minor_status</i>	GSSAPI return code. If the call was successful, the minor status is equal to GLOBUS_SUCCESS. Otherwise, it is an error object ID for which globus_error_get() and globus_object_free() can be used to get and destroy it.
<i>context_handle</i>	the context.
<i>data</i>	pointer to application data to wrap and send
<i>length</i>	length of the <i>data</i> array
<i>token_status</i>	assist routine get/send token status
<i>gss_assist_send_token</i>	a send_token routine
<i>gss_assist_send_context</i>	first arg for the send_token
<i>fperr</i>	file handle to write error message to.

Returns

GSS_S_COMPLETE on success Other gss errors on failure.

See also

gss_wrap()

3.3 GSI GSS Assist Constants

Enumerations

- enum **globus_gsi_gss_assist_error_t** { **GLOBUS_GSI_GSS_ASSIST_ERROR_SUCCESS** = 0, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_ARGUMENTS** = 1, **GLOBUS_GSI_GSS_ASSIST_ERROR_USER_ID_DOESNT_MATCH** = 2, **GLOBUS_GSI_GSS_ASSIST_ERROR_IN_GRIDMAP_NO_USER_ENTRY** = 3, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_GRIDMAP** = 4, **GLOBUS_GSI_GSS_ASSIST_ERROR_INVALID_GRIDMAP_FORMAT** = 5, **GLOBUS_GSI_GSS_ASSIST_ERROR_ERRNO** = 6, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_INIT** = 7, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_WRAP** = 8, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_TOKEN** = 9, **GLOBUS_GSI_GSS_ASSIST_ERROR_EXPORTING_CONTEXT** = 10, **GLOBUS_GSI_GSS_ASSIST_ERROR_IMPORTING_CONTEXT** = 11, **GLOBUS_GSI_GSS_ASSIST_ERROR_INITIALIZING_CALLOUT_HANDLE** = 12, **GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_CALLOUT_CONFIG** = 13, **GLOBUS_GSI_GSS_ASSIST_CALLOUT_ERROR** = 14, **GLOBUS_GSI_GSS_ASSIST_GSSAPI_ERROR** = 15, **GLOBUS_GSI_GSS_ASSIST_GRIDMAP_LOOKUP_FAILED** = 16, **GLOBUS_GSI_GSS_ASSIST_BUFFER_TOO_SMALL** = 17, **GLOBUS_GSI_GSS_ASSIST_ERROR_CANONICALIZING_HOSTNAME** = 18 }

3.3.1 Enumeration Type Documentation

3.3.1.1 enum **globus_gsi_gss_assist_error_t**

GSI GSS Assist Error codes.

Enumerator:

- GLOBUS_GSI_GSS_ASSIST_ERROR_SUCCESS** Success.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_ARGUMENTS** No user entry in gridmap file.
- GLOBUS_GSI_GSS_ASSIST_ERROR_USER_ID_DOESNT_MATCH** Error user ID doesn't match.
- GLOBUS_GSI_GSS_ASSIST_ERROR_IN_GRIDMAP_NO_USER_ENTRY** Error with arguments passed to function.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_GRIDMAP** Error querying gridmap file.
- GLOBUS_GSI_GSS_ASSIST_ERROR_INVALID_GRIDMAP_FORMAT** Invalid gridmap file format.
- GLOBUS_GSI_GSS_ASSIST_ERROR_ERRNO** System Error.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_INIT** Error during context initialization.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_WRAP** Error during message wrap.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_TOKEN** Error with token.
- GLOBUS_GSI_GSS_ASSIST_ERROR_EXPORTING_CONTEXT** Error exporting context.
- GLOBUS_GSI_GSS_ASSIST_ERROR_IMPORTING_CONTEXT** Error importing context.
- GLOBUS_GSI_GSS_ASSIST_ERROR_INITIALIZING_CALLOUT_HANDLE** Error initializing callout handle.
- GLOBUS_GSI_GSS_ASSIST_ERROR_WITH_CALLOUT_CONFIG** Error reading callout configuration.
- GLOBUS_GSI_GSS_ASSIST_CALLOUT_ERROR** Error invoking callout.
- GLOBUS_GSI_GSS_ASSIST_GSSAPI_ERROR** A GSSAPI returned an error.
- GLOBUS_GSI_GSS_ASSIST_GRIDMAP_LOOKUP_FAILED** Gridmap lookup failure.
- GLOBUS_GSI_GSS_ASSIST_BUFFER_TOO_SMALL** Caller provided insufficient buffer space for local identity.
- GLOBUS_GSI_GSS_ASSIST_ERROR_CANONICALIZING_HOSTNAME** Failed to obtain canonical host name.

3.4 Security Token Transport

Token Get File Descriptor

- int **globus_gss_assist_token_get_fd** (void *arg, void **bufp, size_t *sizep)

Token Send File Descriptor

- int **globus_gss_assist_token_send_fd** (void *arg, void *buf, size_t size)

Token Send File Descriptor Without Length

- int **globus_gss_assist_token_send_fd_without_length** (void *arg, void *buf, size_t size)

Token Send File Descriptor Flag EX

- int **globus_gss_assist_token_send_fd_ex** (void *exp, void *buf, size_t size)

3.4.1 Detailed Description

Token routines using fread and fwrite. Additional code has been added to detect tokens which are sent without a length field. These can currently be only SSL tokens. This does require some knowledge of the underlying GSSAPI, by the application, but is within the guidelines of the GSSAPI specifications.

The get routine will automatically attempt this test, while a new send routine will check a flag. The old send routine will work as before, sending a 4-byte length.

3.4.2 Function Documentation

3.4.2.1 int globus_gss_assist_token_get_fd (void * arg, void ** bufp, size_t * sizep)

Use a open file discriptor to get a token.

This function provides parameter types that allow it to be passed to **globus_gss_assist_init_sec_context** (p. 11) and **globus_gss_assist_accept_sec_context** (p. 5)

Parameters

<i>arg</i>	the FILE * stream cast to a void pointer
<i>bufp</i>	the resulting token
<i>sizep</i>	the size (number of bytes) read into bufp

Returns

0 on success > 0 is internal return < 0 is the -errno

3.4.2.2 int globus_gss_assist_token_send_fd (void * arg, void * buf, size_t size)

Write a token to the open file descriptor.

Will write it with a 4 byte length. This function provides parameter types that allow it to be passed to **globus_gss_assist_init_sec_context** (p. 11) and **globus_gss_assist_accept_sec_context** (p. 5)

Parameters

<i>arg</i>	the FILE * stream to send the token on
<i>buf</i>	the token
<i>size</i>	the size of the token in bytes

Returns

0 on success >0 on error <0 on errno error

3.4.2.3 int globus_gss_assist_token_send_fd_without_length (void * *arg*, void * *buf*, size_t *size*)

Write a token to the open file descriptor.

Will write it without a length. so as to

3.4.2.4 int globus_gss_assist_token_send_fd_ex (void * *exp*, void * *buf*, size_t *size*)

Write a token to the open file descriptor.

will look at the flag to determine if the length field need to be written.

Parameters

<i>exp</i>	the globus_gss_assist_ex variable that holds the FILE * stream and flags to bet set
<i>buf</i>	the token buffer to send
<i>size</i>	size of the token buffer

Returns

0 on success >0 on error <0 on errno error (-errno)