globus gram protocol Reference Manual

11.3

Generated by Doxygen 1.4.7

Wed Jul 25 08:17:00 2012

Contents

1 Globus GRAM Protocol

The Globus GRAM Protocol Library implements the GRAM protocol. It is used by the GRAM Client and GRAM Job Manager. It provides the constants used by in the sending and receiving of GRAM messages. It also provides functions to encode GRAM requests and replies, and to send and receive the GRAM queries.

- **GRAM Protocol Functions** (p. 2)
- **Job States** (p. 4)
- **Signals** (p. 2)
- **GRAM Errors** (p. 4)
- **GRAM Protocol Message Format** (p. 27)

2 globus gram protocol Module Index

2.1 globus gram protocol Modules

Here is a list of all modules:

3 globus gram protocol Page Index

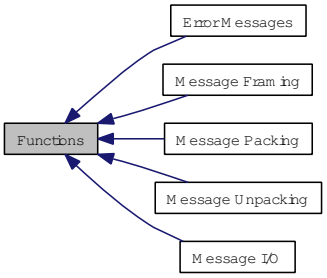
3.1 globus gram protocol Related Pages

Here is a list of all related documentation pages:

4 globus gram protocol Module Documentation

4.1 Functions

Collaboration diagram for Functions:



Modules

- **Error Messages**
- **Message Framing**
- **Message I/O**
- **Message Packing**
- **Message Unpacking**

4.2 GRAM Signals

Enumerations

- enum **globus_gram_protocol_job_signal_t {**
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_CANCEL = 1,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_SUSPEND = 2,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_RESUME = 3,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_PRIORITY = 4,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_REQUEST = 5,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_EXTEND = 6,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_UPDATE = 7,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_SIZE = 8,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STOP_MANAGER = 9,
  GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_END = 10 }

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum globus_gram_protocol_job_signal_t

GRAM Signals.

Enumerator:

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_CANCEL Cancel a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_SUSPEND Suspend a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_RESUME Resume a previously suspended job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_PRIORITY Change the priority of a job.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_REQUEST Signal the job manager to commence with a job submission if the job request was accompanied by the (two_state=yes) RSL attribute.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_EXTEND Signal the job manager to wait an additional number of seconds (specified by an integer value string as the signal's argument) before timing out a two-phase job commit.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_UPDATE Signal the job manager to change the way it is currently handling standard output and/or standard error.

The argument for this signal is an RSL containing new *stdout*, *stderr*, *stdout_position*, *stderr_position*, or *remote_io_url* relations.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STDIO_SIZE Signal the job manager to verify that streamed I/O has been completely received.

The argument to this signal contains the number of bytes of stdout and stderr received, separated by a space. The reply to this signal will be a SUCCESS message if these matched the amount sent by the job manager. Otherwise, an error reply indicating GLOBUS_GRAM_PROTOCOL_ERROR_STDIO_-SIZE is returned. If standard output and standard error are merged, only one number should be sent as an argument to this signal. An argument of -1 for either stream size indicates that the client is not interested in the size of that stream.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_STOP_MANAGER Signal the job manager to stop managing the current job and terminate.

The job continues to run as normal. The job manager will send a state change callback with the job status being FAILED and the error GLOBUS_GRAM_PROTOCOL_ERROR_JM_STOPPED.

GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_COMMIT_END Signal the job manager to clean up after the completion of the job if the job RSL contained the (two-phase = yes) relation.

4.3 GRAM Job States

The globus_gram_protocol_job_state_t contains information about the current state of the job as known by the job manager.

Enumerations

- enum **globus_gram_protocol_job_state_t {**
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING = 1,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE = 2,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED = 4,
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE = 8,

GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED = 16,
GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED = 32,
GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN = 64,
GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT = 128,
GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL = 0xFFFFF }

4.3.1 Detailed Description

The globus_gram_protocol_job_state_t contains information about the current state of the job as known by the job manager.

Job state changes are sent by the Job Manager to all registered clients. A client may ask for information from the job manager via the status request.

4.3.2 Enumeration Type Documentation

4.3.2.1 enum globus_gram_protocol_job_state_t

GRAM Job States.

Enumerator:

GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING The job is waiting for resources to become available to run.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE The job has received resources and the application is executing.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED The job terminated before completion because an error, user-triggered cancel, or system-triggered cancel.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE The job completed successfully.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED The job has been suspended.

Resources which were allocated for this job may have been released due to some scheduler-specific reason.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED The job has not been submitted to the scheduler yet, pending the reception of the GLOBUS_GRAM_PROTOCOL_JOB_SIGNAL_-COMMIT_REQUEST signal from a client.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN The job manager is staging in files to run the job.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT The job manager is staging out files generated by the job.

GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL A mask of all job states.

4.4 GRAM Error codes

Enumerations

- enum **globus_gram_protocol_error_t**

4.4.1 Detailed Description

4.4.2 Enumeration Type Documentation

4.4.2.1 enum globus_gram_protocol_error_t

GRAM Error codes.

4.5 Error Messages

Collaboration diagram for Error Messages:

Functions in this section handle converting GRAM error codes to strings which can help the user diagnose GRAM problems.

Functions

- const char * **globus_gram_protocol_error_string (int error_code)**
- **void globus_gram_protocol_error_7_hack_replace_message (const char *message)**
- **void globus_gram_protocol_error_10_hack_replace_message (const char *message)**

4.5.1 Detailed Description

Functions in this section handle converting GRAM error codes to strings which can help the user diagnose GRAM problems.

4.5.2 Function Documentation

4.5.2.1 const char* globus_gram_protocol_error_string (int *error_code*)

Get a description of a a GRAM error code.

The **globus_gram_protocol_error_string()** (p. 5) function takes a GRAM error code value and returns the associated error code string for the message. The string is statically allocated by the GRAM Protocol library and should not be modified or freed by the caller. The string is intended to complete a sentence of the form "[operation] failed because ..."

Parameters:

error_code The error code to translate into a string.

Returns:

The **globus_gram_protocol_error_string()** (p. 5) function returns a static string containing an explanation of the error.

4.5.2.2 void globus_gram_protocol_error_7_hack_replace_message (const char * *message*)

Replace the error message associated with error 7 with a custom message.

The **globus_gram_protocol_error_7_hack_replace_message()** (p. 5) function creates a custom version of the error message for the error *GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION*. The string pointed to by the *message* parameter is copied to thread-local storage, and subsequent calls to **globus_gram_protocol_error_string()** (p. 5) with this error number will return this copy of the string. Each time **globus_gram_protocol_error_7_hack_replace_message()** (p. 5) is called for a particular thread, the previous message is freed.

The purpose of this function is to allow more meaningful error messages to be generated when authentication failures occur. In particular, the specific GSSAPI error reason can be used in place of a generic authorization failure message.

Parameters:

message The new message to be associated with the GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION error code.

Note:

Since Globus 5.0.0, this function uses thread-specific storage, so that the value returned by **globus_gram_protocol_error_string()** (p. 5) for GLOBUS_GRAM_PROTOCOL_ERROR_AUTHORIZATION is that for the last authorization error where **globus_gram_protocol_error_7_hack_replace_message()** (p. 5) was called from this thread.

4.5.2.3 void globus_gram_protocol_error_10_hack_replace_message (const char * *message*)

Replace the error message associated with error 10 with a custom message.

The **globus_gram_protocol_error_10_hack_replace_message()** (p. 6) function creates a custom version of the error message for the error *GLOBUS_GRAM_PROTOCOL_ERROR_PROTOCOL_FAILED*. The string pointed to by the *message* parameter is copied to thread-local storage, and subsequent calls to **globus_gram_protocol_error_string()** (p. 5) with this error number will return this copy of the string. Each time **globus_gram_protocol_error_10_hack_replace_message()** (p. 6) is called for a particular thread, the previous message is freed.

The purpose of this function is to allow more meaningful error messages to be generated when protocol errors occur. In particular, the specific XIO error reason can be used in place of a generic protocol failure message.

Parameters:

message The new message to be associated with the GLOBUS_GRAM_PROTOCOL_ERROR_PROTOCOL_FAILED error code.

Note:

Since Globus 5.0.0, this function uses thread-specific storage, so that the value returned by **globus_gram_protocol_error_string()** (p. 5) for GLOBUS_GRAM_PROTOCOL_ERROR_PROTOCOL_FAILED is that for the last authorization error where **globus_gram_protocol_error_10_hack_replace_message()** (p. 6) was called from this thread.

4.6 Message Framing

Collaboration diagram for Message Framing:

The functions in this section frame a GRAM request, query, or reply message with HTTP headers compatible with the GRAM2 protocol parsers in GT2 GT3, and GT4.

Functions

- int **globus_gram_protocol_frame_request (const char *url, const globus_byte_t *msg, globus_size_t msgsize, globus_byte_t **framedmsg, globus_size_t *framedsize)**
- **int globus_gram_protocol_frame_reply (int code, const globus_byte_t *msg, globus_size_t msgsize, globus_byte_t **framedmsg, globus_size_t *framedsize)**

4.6.1 Detailed Description

The functions in this section frame a GRAM request, query, or reply message with HTTP headers compatible with the GRAM2 protocol parsers in GT2 GT3, and GT4.

These functions should be used when an application wants to control the way that the GRAM Protocol messages are sent, while still using the standard message formatting and framing routines. An alternative set of functions in the **Message I/O** (p. 8) section of the manual combine message framing with callback-driven I/O.

4.6.2 Function Documentation

4.6.2.1 int globus_gram_protocol_frame_request (const char * *url*, const globus_byte_t * *msg*, globus_size_t *msgsize*, globus_byte_t ** *framedmsg*, globus_size_t * *framedsize*)

Create a HTTP-framed copy of a GRAM request.

The **globus_gram_protocol_frame_request()** (p. 7) function adds HTTP 1.1 framing around the input message. The framed message includes HTTP headers relating the the destination URL and the length of the message content. The framed message is returned by modifying *framedmsg* to point to a newly allocated string. The integer pointed to by the *framedsize* parameter is set to the length of this message.

Parameters:

url The URL of the GRAM resource to contact. This is parsed and used to generate the HTTP POST operation destination and the Host HTTP header.

msg A string containing the message content to be framed.

msgsize The length of the string pointed to by *msg*

framedmsg An output parameter which will be set to a copy of the *msg* string with an HTTP frame around it.

framedsize An output parameter which will be set to the length of the framed message.

Returns:

Upon success, **globus_gram_protocol_frame_request()** (p. 7) will return GLOBUS_SUCCESS and the *framedmsg* and *framedsize* parameters will be modified to point to the new framed message string and its length respectively. When this occurs, the caller is responsible for freeing the string pointed to by *framedmsg*. If an error occurs, its value will returned and the *framedmsg* and *framedsize* parameters will be uninitialized.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT Invalid job contact

4.6.2.2 int globus_gram_protocol_frame_reply (int *code*, const globus_byte_t * *msg*, globus_size_t *msgsize*, globus_byte_t ** *framedmsg*, globus_size_t * *framedsize*)

Create a HTTP-framed copy of a GRAM reply.

The **globus_gram_protocol_frame_reply()** (p. 7) function adds HTTP 1.1 framing around the input message. The framed message includes HTTP headers relating the the status of the operation being replied to and the length of the message content. The framed message is returned by modifying *framedmsg* to point to a newly allocated string. The integer pointed to by the *framedsize* parameter is set to the length of this message.

Parameters:

code The HTTP response code to send along with this reply.

msg A string containing the reply message content to be framed.

msgsize The length of the string pointed to by *msg*.

framedmsg An output parameter which will be set to a copy of the *msg* string with an HTTP reply frame around it.

framedsize An output parameter which will be set to the length of the framed reply string pointed to by *framedmsg*.

Returns:

Upon success, **globus_gram_protocol_frame_reply()** (p. 7) will return GLOBUS_SUCCESS and the *framedmsg* and *framedsize* parameters will be modified to point to the new framed message string and its length respectively. When this occurs, the caller is responsible for freeing the string pointed to by *framedmsg*. If an error occurs, its value will returned and the *framedmsg* and *framedsize* parameters will be uninitialized.

Return values:

GLOBUS_SUCCESS Success

4.7 Message I/O

Collaboration diagram for Message I/O:

The functions in this section are related to sending and receiving GRAM protocol messages.

Typedefs

- typedef unsigned long **globus_gram_protocol_handle_t**
- **typedef globus_gram_protocol_hash_entry_s globus_gram_protocol_extension_t**

Functions

- int **globus_gram_protocol_setup_attr (globus_io_attr_t *attr)**
- **globus_bool_t globus_gram_protocol_authorize_self (gss_ctx_id_t context)**
- **int globus_gram_protocol_allow_attach (char **url, globus_gram_protocol_callback_t callback, void *callback_arg)**
- **int globus_gram_protocol_callback_disallow (char *url)**
- **int globus_gram_protocol_post (const char *url, globus_gram_protocol_handle_t *handle, globus_io_attr_t *attr, globus_byte_t *message, globus_size_t message_size, globus_gram_protocol_callback_t callback, void *callback_arg)**

- **int globus_gram_protocol_post_delegation (const char *url, globus_gram_protocol_handle_t *handle, globus_io_attr_t *attr, globus_byte_t *message, globus_size_t message_size, gss_cred_id_t cred_handle, gss_OID_set restriction_oids, gss_buffer_set_t restriction_buffers, OM_uint32 req_flags, OM_uint32 time_req, globus_gram_protocol_callback_t callback, void *callback_arg)**
- **int globus_gram_protocol_reply (globus_gram_protocol_handle_t handle, int code, globus_byte_t *message, globus_size_t message_size)**
- **int globus_gram_protocol_accept_delegation (globus_gram_protocol_handle_t handle, gss_OID_set restriction_oids, gss_buffer_set_t restriction_buffers, OM_uint32 req_flags, OM_uint32 time_req, globus_gram_protocol_delegation_callback_t callback, void *arg)**
- **int globus_gram_protocol_get_sec_context (globus_gram_protocol_handle_t handle, gss_ctx_id_t *context)**

4.7.1 Detailed Description

The functions in this section are related to sending and receiving GRAM protocol messages.

4.7.2 Typedef Documentation

4.7.2.1 globus_gram_protocol_handle_t

Unique GRAM protocol identifier.

The ***globus_gram_protocol_handle_t*** *(p. 9)* data type is used by functions in the GRAM protocol API as a unique discriminant between instances of a callback invocation.

There are no public functions that operate on these handles. They are used as identifiers for callback functions.

4.7.2.2 globus_gram_protocol_extension_t

GRAM protocol extension attribute-value pair.

The *globus_gram_protocol_extension_t* data type contains an attribute value pair that represents an extension to the GRAM2 protocol.

4.7.3 Function Documentation

4.7.3.1 int globus_gram_protocol_setup_attr (globus_io_attr_t * *attr*)

Create default I/O attribute for GRAM.

The **globus_gram_protocol_setup_attr()** (p. 9) function creates a new globus_io attribute containing the default set of values needed for communication between a GRAM client and a job manager. These attributes include:

- SO_KEEPALIVE
- GSSAPI Mutual Authentication
- GSSAPI Self Authorization
- SSL-compatible message wrapping

Parameters:

attr A pointer to a *globus_io_attr_t* structure which will be initialized by this function.

Returns:

Upon success, **globus_gram_protocol_setup_attr()** (p. 9) modifies the *attr* parameter to point to a new attribute and returns the value *GLOBUS_SUCCESS*. When this occurs, the caller must destroy the attribute

when no longer needed by calling globus_io_tcpattr_destroy(). If an error occurs, its value will be returned and the attribute pointed to by the *attr* parameter will be set to an uninitialized state.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_CONNECTION_FAILED Error initializing attribute

4.7.3.2 globus_bool_t globus_gram_protocol_authorize_self (gss_ctx_id_t *context*)

Determine if a GSSAPI context has the same source and target identities.

The **globus_gram_protocol_authorize_self()** (p. 9) function implements a predicate which returns true if the source and destination identities used to establish the GSSAPI security context are the same.

Parameters:

context A GSSAPI security context which has been previously established. The source and target names of this context will be inspected by this function.

Returns:

If the source and target identiies are the same, then ***globus_gram_protocol_authorize_self()*** *(p. 9)* returns *GLOBUS_TRUE*, otherwise, this function returns *GLOBUS_FALSE*.

Return values:

GLOBUS_TRUE The source and target identities are the same.

GLOBUS_FALSE The source and target identities are not the same or this function is unabled to inspect the security context.

4.7.3.3 int globus_gram_protocol_allow_attach (char ** *url*, globus_gram_protocol_callback_t *callback*, void * *callback_arg*)

Create a GRAM protocol service listener.

The **globus_gram_protocol_allow_attach()** (p. 10) function creates a GRAM protocol listener to which other processes can send GRAM protocol messages. The listener will automatically accept new connections on it's TCP/IP port and parse GRAM requests. The requests will be passed to the function pointed to by the *callback* parameter for the application to unpack, handle, and send a reply by calling **globus_gram_protocol_reply()** (p. 14).

Parameters:

url An output parameter that will be initialized to point to a string that will hold the URL of the new listener. This URL may be published or otherwise passed to applications which need to contact this GRAM protocol server. The URL will be of the form `https://host:port/`.

callback A pointer to a function to be called when a new request has been received by this listener. This function will be passed the request, which may be unpacked using one of the functions described in the **message packing** (p. 16) section of the documentation.

callback_arg A pointer to arbitrary user data which will be passed to the callback function as its first parameter.

Returns:

Upon success, **globus_gram_protocol_allow_attach()** (p. 10) returns *GLOBUS_SUCCESS* and modifies the *url* parameter to point to a newly allocated string. The caller is then responsible for freeing this string. If an error occurs, an integer error code will be returned and the *url* parameter value will be uninitialized.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES No resources

See also:

globus_gram_protocol_callback_disallow() (p. 11)

4.7.3.4 int globus_gram_protocol_callback_disallow (char * *url*)

Stop a GASS protocol listener from handling new requests.

The **globus_gram_protocol_callback_disallow()** (p. 11) function stops the listener named by the value of the *url* parameter from receiving any new requests. It also frees memory used internally by the GRAM protocol implementation to handle requests for this listener.

The **globus_gram_protocol_callback_disallow()** (p. 11) function will wait until all requests being processed by this listener have completed processing. Once **globus_gram_protocol_callback_disallow()** (p. 11) returns, no further request callbacks will occur for the listener.

Parameters:

url A pointer to the URL string which names the listener to disable.

Returns:

Upon success, the **globus_gram_protocol_callback_disallow()** (p. 11) function returns *GLOBUS_SUCCESS* and frees internal state associated with the listener named by the *url* parameter. If an error occurs, its integer error code value will be returned and no listener will be affected.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT Invalid job contact

GLOBUS_GRAM_PROTOCOL_ERROR_CALLBACK_NOT_FOUND Callback not found

See also:

globus_gram_protocol_allow_attach() (p. 10)

4.7.3.5 int globus_gram_protocol_post (const char * *url*, globus_gram_protocol_handle_t * *handle*, globus_io_attr_t * *attr*, globus_byte_t * *message*, globus_size_t *message_size*, globus_gram_protocol_callback_t *callback*, void * *callback_arg*)

Post a GRAM protocol request to a GRAM server.

The **globus_gram_protocol_post()** (p. 11) function initiates a GRAM protocol message exchange with a GRAM protocol listener. It returns after framing the message and initiating the connection. When the message exchange is complete, the function pointed to by *callback* is invoked either in another thread or when a non-threaded application calls the **globus_poll()** (o **??**) **globus_cond_wait()** (f **??**)nctions.

Parameters:

url A pointer to a string containing the URL of the server to post the request to. This URL must be an HTTPS URL naming a GRAM service resource.

handle A pointer to a *globus_gram_protocol_handle_t* which will be initialized with a unique handle identifier. This identifier will be passed to the *callback* function to allow the caller to differentiate replies to multiple GRAM Protocol requests. This pointer may be NULL if the caller will not have multiple simultaneous requests.

attr A pointer to a Globus I/O attribute set, which will be used as parameters when connecting to the GRAM server. The value of *attr* may be NULL, in which case, the default GRAM Protocol attributes will be used (authentication to self, SSL-compatible transport, with message integrity).

message A pointer to a message string to be sent to the GRAM server. This is normally created by calling one of the GRAM Protocol **pack** (p. 16) functions. This message need not be NULL terminated as the length is passed in the *message_size* parameter.

message_size The length of the *message* string. Typically generated as one of the output parameters to one of the GRAM Protocol **pack** (p. 16) functions.

callback A pointer to a function to call when the response to this message is received or the message exchange fails. This may be NULL, in which case no callback will be received, and the caller will be unable to verify whether the message was successfully received.

callback_arg A pointer to application-specific data which will be passed to the function pointed to by *callback* as its first parameter. This may be NULL if the application has a NULL *callback* or does not require the pointer to establish its context in the callback.

Returns:

Upon success, **globus_gram_protocol_post()** (p. 11) returns GLOBUS_SUCCESS, initiates the message exchange, registers the function pointed to by *callback* to be called when the exchange completes or fails, and modifies the *handle* parameter if it is non-NULL. If an error occurs, its error code will be returned, the *handle* parameter will be uninitialized and the function pointed to be *callback* will not be called.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT Invalid job contact

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES No resources

Note:

There is no way to time out or cancel a service request that is begun with **globus_gram_protocol_post()** (p. 11).

See also:

globus_gram_protocol_reply() (p. 14)

4.7.3.6 int globus_gram_protocol_post_delegation (const char * *url*, globus_gram_protocol_handle_t * *handle*, globus_io_attr_t * *attr*, globus_byte_t * *message*, globus_size_t *message_size*, gss_cred_id_t *cred_handle*, gss_OID_set *restriction_oids*, gss_buffer_set_t *restriction_buffers*, OM_uint32 *req_flags*, OM_uint32 *time_req*, globus_gram_protocol_callback_t *callback*, void * *callback_arg*)

Post a GRAM protocol delegation request to a GRAM server.

The **globus_gram_protocol_post_delegation()** (p. 12) function initiates a GRAM protocol delegation exchange with a GRAM protocol listener. The delegation protocol is a custom mix of HTTP and SSL records.

The **globus_gram_protocol_post_delegation()** (p. 12) function returns after framing the message and initiating the connection to be used for delegation. When the message exchange is complete, the function pointed to by *callback* is invoked either in another thread or when a non-threaded application calls the **globus_poll()** (o **??**) **globus_cond_wait()** (f **??**)nctions.

Parameters:

url A pointer to a string containing the URL of the server to post the request to. This URL must be an HTTPS URL naming a GRAM service resource.

handle A pointer to a *globus_gram_protocol_handle_t* which will be initialized with a unique handle identifier. This identifier will be passed to the *callback* function to allow the caller to differentiate replies to multiple GRAM Protocol requests. This pointer may be NULL if the caller will not have multiple simultaneous requests.

attr A pointer to a Globus I/O attribute set, which will be used as parameters when connecting to the GRAM server. The value of *attr* may be NULL, in which case, the default GRAM Protocol attributes will be used (authentication to self, SSL-compatible transport, with message integrity).

message A pointer to a message string to be sent to the GRAM server. This is normally created by calling one of the GRAM Protocol **pack** (p. 16) functions. This message need not be NULL terminated as the length is passed in the *message_size* parameter.

message_size The length of the *message* string. Typically generated as one of the output parameters to one of the GRAM Protocol **pack** (p. 16) functions.

cred_handle Handle to an existing GSSAPI security credential. If this parameter is set to *GSS_C_NO_CREDENTIAL*, then the current account's default credential will be used. A proxy credential sharing the identity of this credential will be delegated to the GRAM protocol server.

restriction_oids A set of OID values indicating the data in the *restriction_buffers* parameter. This parameter may have the value GSS_C_NO_OID_SET if there are no restriction buffers.

restriction_buffers A set of binary data buffers which will be included in the delegated credential. The type of data in these buffers is determined by the OID values in *restriction_oids*. This parameter may have the value GSS_C_EMPTY_BUFFER_SET if there are no extra restrictions to be added to the credential.

req_flags A bitwise-or of GSSAPI flag values to use when delegating the credential using gss_init_delegation().

time_req An integer value indicating the length of time (in seconds) that the delegated credential should be valid for. This is an advisory parameter: no error will be returned if a credential with the requested lifetime can not be created.

callback A pointer to a function to call when the response to this message is received or the message exchange fails. This may be NULL, in which case no callback will be received, and the caller will be unable to verify whether the message was successfully received.

callback_arg A pointer to application-specific data which will be passed to the function pointed to by *callback* as its first parameter. This may be NULL if the application has a NULL *callback* or does not require the pointer to establish its context in the callback.

Returns:

Upon success, **globus_gram_protocol_post_delegation()** (p. 12) returns GLOBUS_SUCCESS, initiates the message exchange, registers the function pointed to by *callback* to be called when the exchange completes or fails, and modifies the *handle* parameter if it is non-NULL. If an error occurs, its error code will be returned, the *handle* parameter will be uninitialized and the function pointed to be *callback* will not be called. In the case of a protocol or delegation failure, the callback function will be called with the *errorcode* parameter set to the error.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_JOB_CONTACT Invalid job contact

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES No resources

Note:

There is no way to time out or cancel a service request that is begun with **globus_gram_protocol_post_-delegation()** (p. 12).

See also:

globus_gram_protocol_reply() (p. 14)

4.7.3.7 int globus_gram_protocol_reply (globus_gram_protocol_handle_t *handle*, int *code*, globus_byte_t * *message*, globus_size_t *message_size*)

Reply to a GRAM protocol message.

The ***globus_gram_protocol_reply()*** *(*p. *14)* function sends a response message to a client which initiated a GRAM message exchange. The ***globus_gram_protocol_reply()*** *(*p. *14)* function composes the message with an HTTP message frame and then sends it to the client which initiated the exchange.

Parameters:

handle A GRAM protocol handle which is used by this function to determine the network connection to use for this reply. This must be the same value as was passed as a parameter to the callback function registered with the ***globus_gram_protocol_allow_attach()*** *(*p. *10)* function.

code The HTTP response code. The code should be one from the set described in RFC 2616.

message A pointer to a message string to be sent to the GRAM client. This is normally created by calling one of the GRAM Protocol **pack** (p. 16) functions. This message need not be NULL terminated as the length is passed in the *message_size* parameter.

message_size The length of the *message* string. Typically generated as one of the output parameters to one of the GRAM Protocol **pack** (p. 16) functions.

Returns:

Upon success, ***globus_gram_protocol_reply()*** *(*p. *14)* returns GLOBUS_SUCCESS, frames the *message* with an HTTP header and initiates sending the message to the client. The caller must not try to use the value of the *handle* parameter after this function returns. If an error occurs, its integer error code will be returned.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES No Resources

See also:

globus_gram_protocol_allow_attach() (p. 10)

4.7.3.8 int globus_gram_protocol_accept_delegation (globus_gram_protocol_handle_t *handle*, gss_OID_-set *restriction_oids*, gss_buffer_set_t *restriction_buffers*, OM_uint32 *req_flags*, OM_uint32 *time_req*, globus_gram_protocol_delegation_callback_t *callback*, void * *arg*)

Perform the server-side of the GSSAPI delegation handshake to receive a new delegated credential.

The **globus_gram_protocol_accept_delegation()** (p. 14) function performs the service side accepting of a GRAM protocol delegation exchange with a GRAM protocol client. This is performed after the delegation HTTP message has been unpacked by the application.

The **globus_gram_protocol_accept_delegation()** (p. 14) function returns after processing the GSSAPI handshake, passing the delegated credential or error information to the function pointed to by the *callback* parameter.

Parameters:

handle A GRAM protocol handle on which the server received a protocol refresh message.

restriction_oids A set of OID values indicating the data in the *restriction_buffers* parameter. This parameter may have the value GSS_C_NO_OID_SET if there are no restriction buffers.

restriction_buffers A set of binary data buffers which will be included in the delegated credential. The type of data in these buffers is determined by the OID values in *restriction_oids*. This parameter may have the value GSS_C_EMPTY_BUFFER_SET if there are no extra restrictions to be added to the credential.

req_flags A bitwise-or of GSSAPI flag values to use when delegating the credential using gss_init_-delegation().

time_req An integer value indicating the length of time (in seconds) that the delegated credential should be valid for. This is an advisory parameter: no error will be returned if a credential with the requested lifetime can not be created.

callback A pointer to a function to call when the delegation handshake has completed or failed. This function will be passed the value of *arg* as well as the handle and delegated credential or erorr that occurred processing the delegation messages.

arg A pointer to application-specific data which will be passed to the function pointed to by *callback* as its first parameter. This may be NULL if the application has a NULL *callback* or does not require the pointer to establish its context in the callback.

Returns:

Upon success, **globus_gram_protocol_accept_delegation()** (p. 14) returns GLOBUS_SUCCESS and registers the function pointed to by *callback* to be called after the delegation completes or fails. If an error occurs, **globus_gram_protocol_accept_delegation()** (p. 14) returns an integer error code and the *callback* function is not registered.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Malloc failed

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

GLOBUS_GRAM_PROTOCOL_ERROR_NO_RESOURCES No resources

4.7.3.9 int globus_gram_protocol_get_sec_context (globus_gram_protocol_handle_t *handle*, gss_ctx_id_t * *context*)

Get a reference to the GSSAPI security context associated with a GRAM protocol handle.

The ***globus_gram_protocol_get_sec_context()*** *(p. 15)* function retrieves a reference to the GSSAPI security context associated with a particular GRAM protocol handle. This context may be inspected by the caller but must not be destroyed by the caller. The ***globus_gram_protocol_get_sec_context()*** *(p. 15)* function must only be called after the GRAM protocol library has called the *callback* function associated with a GRAM protocol message exchange.

Parameters:

handle The GRAM protocol handle associated with a GRAM protocol message exchange.

context The GSSAPI security context associated with the protocol handle.

Returns:

Upon success, ***globus_gram_protocol_get_sec_context()*** *(p. 15)* returns GLOBUS_SUCCESS and modifies the *context* parameter to point to the security context associated with the *handle* parameter. If an error occurs, an interger error code is returned and the value of the *context* parameter is undefined.

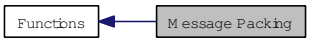
Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_INVALID_REQUEST Invalid request

4.8 Message Packing

Collaboration diagram for Message Packing:



Functions

- int **globus_gram_protocol_pack_job_request (int job_state_mask, const char *callback_url, const char *rsl, globus_byte_t **query, globus_size_t *querysize)**
- **int globus_gram_protocol_pack_job_request_reply (int status, const char *job_contact, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_job_request_reply_with_extensions (int status, const char *job_contact, globus_hashtable_t *extensions, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_status_request (const char *status_request, globus_byte_t **query, globus_size_t *querysize)**
- **int globus_gram_protocol_pack_status_reply (int job_status, int failure_code, int job_failure_code, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_status_reply_with_extensions (int job_status, int failure_code, int job_failure_code, globus_hashtable_t *extensions, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_status_update_message (char *job_contact, int status, int failure_code, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_status_update_message_with_extensions (char *job_contact, int status, int failure_code, globus_hashtable_t *extensions, globus_byte_t **reply, globus_size_t *replysize)**
- **int globus_gram_protocol_pack_version_request (char **request, size_t *requestsize)**

4.8.1 Function Documentation

4.8.1.1 int globus_gram_protocol_pack_job_request (int *job_state_mask*, const char * *callback_url*, const char * *rsl*, globus_byte_t ** *query*, globus_size_t * *querysize*)

Pack a GRAM Job Request.

The **globus_gram_protocol_pack_job_request()** (p. 16) function combines its parameters into a GRAM job request message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_post()** (p. 11) or just frame it by calling **globus_gram_protocol_frame_request()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_job_request()** (p. 16) function returns the packed message by modifying the *query* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

job_state_mask The bitwise-or of the GRAM job states which the client would like to register for job state change callbacks.

callback_url A callback contact string which will be contacted when a job state change which matches the *job_state_mask* occurs. This may be NULL, if the client does not wish to register a callback contact with this job request. Typically, this value is returned in the *url* parameter to **globus_gram_protocol_allow_attach()** (p. 10).

rsl An RSL string which contains the job request. This will be processed on the server side.

query An output parameter which will be set to a new string containing the packed job request message. The caller must free this memory by calling free()

querysize An output parameter which will be populated with the length of the job request message returned in *query*.

Returns:

Upon success, **globus_gram_protocol_pack_job_request()** (p. 16) returns GLOBUS_SUCCESS and modifies the *query* and *querysize* parameters to point to the values described above.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

4.8.1.2 int globus_gram_protocol_pack_job_request_reply (int *status*, const char * *job_contact*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM reply message.

The **globus_gram_protocol_pack_job_request_reply()** (p. 17) function combines its parameters into a GRAM reply message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_reply()** (p. 14) or just frame it by calling **globus_gram_protocol_frame_reply()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_job_request_reply()** (p. 17) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

status The job's failure code if the job failed, or 0, if the job request was processed successfully.

job_contact A string containing the job contact string. This may be NULL, if the job request was not successful.

reply A pointer which will be set to the packed reply string The caller must free this string by calling free().

replysize A pointer which will be set to the length of the reply string.

Returns:

Upon success, **globus_gram_protocol_pack_job_request_reply()** (p. 17) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Out of memory

4.8.1.3 int globus_gram_protocol_pack_job_request_reply_with_extensions (int *status*, const char * *job_contact*, globus_hashtable_t * *extensions*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM reply message with extension attributes.

The **globus_gram_protocol_pack_job_request_reply_with_extensions()** (p. 17) function combines its parameters into a GRAM reply message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_reply()** (p. 14) or just frame it by calling **globus_gram_protocol_frame_reply()** (p. 7) and send

it by some other mechanism. The **globus_gram_protocol_pack_job_request_reply_with_extensions()** (p. 17) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

status The job's failure code if the job failed, or 0, if the job request was processed successfully.

job_contact A string containing the job contact string. This may be NULL, if the job request was not successful.

extensions A pointer to a hash table keyed on a string attribute name with the hash values being pointers to *globus_gram_protocol_extension_t* structures. These will be encoded in the reply message after the standard attributes.

reply A pointer which will be set to the packed reply string The caller must free this string by calling free().

replysize A pointer which will be set to the length of the reply string.

Returns:

Upon success, **globus_gram_protocol_pack_job_request_reply_with_extensions()** (p. 17) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Out of memory

4.8.1.4 int globus_gram_protocol_pack_status_request (const char * *status_request*, globus_byte_t ** *query*, globus_size_t * *querysize*)

Pack a GRAM query message.

The **globus_gram_protocol_pack_status_request()** (p. 18) function combines its parameters into a GRAM status query message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_post()** (p. 11) or just frame it by calling **globus_gram_protocol_frame_request()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_status_request()** (p. 18) function returns the packed message by modifying the *query* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

status_request A string containing the type of query message to send, including any query parameters. The valid strings supported by GRAM in GT5 are:

- status
- register
- unregister
- signal
- renew
- cancel

query An output parameter which will be set to a new string containing the packed job query message.

querysize An output parameter which will be set to the length of the job query message returned in *query*.

Returns:

Upon success, **globus_gram_protocol_pack_status_request()** (p. 18) returns *GLOBUS_SUCCESS* and modifies the *query* and *querysize* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *query* and *querysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Out of memory

4.8.1.5 int globus_gram_protocol_pack_status_reply (int *job_status*, int *failure_code*, int *job_failure_code*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM query reply message.

The **globus_gram_protocol_pack_status_reply()** (p. 19) function combines its parameters into a GRAM status reply message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_reply()** (p. 14) or just frame it by calling **globus_gram_protocol_frame_reply()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_status_reply()** (p. 19) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

job_status The job's current **job state** (p. 4).

failure_code The error code generated by the query. This may be GLOBUS_SUCCESS if the query succeeded.

job_failure_code The error code associated with the job if it has failed. This may be GLOBUS_SUCCESS if the job has not failed.

reply An output parameter which will be set to a new string containing the packed reply message.

replysize An output parameter which will be set to the length of the reply message returned in *reply*.

Returns:

Upon success, **globus_gram_protocol_pack_status_reply()** (p. 19) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Out of memory

4.8.1.6 int globus_gram_protocol_pack_status_reply_with_extensions (int *job_status*, int *failure_code*, int *job_failure_code*, globus_hashtable_t * *extensions*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM query reply message with extensions.

The **globus_gram_protocol_pack_status_reply_with_extensions()** (p. 19) function combines its parameters into a GRAM status reply message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_reply()** (p. 14) or just frame it by calling **globus_gram_protocol_frame_reply()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_status_reply_with_extensions()** (p. 19) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

job_status The job's current **job state** (p. 4).

failure_code The error code generated by the query. This may be GLOBUS_SUCCESS if the query succeeded.

job_failure_code The error code associated with the job if it has failed. This may be GLOBUS_SUCCESS if the job has not failed.

extensions A pointer to a hash table containing the names and values of the protocol extensions to add to this message.

reply An output parameter which will be set to a new string containing the packed reply message.

replysize An output parameter which will be set to the length of the reply message returned in *reply*.

Returns:

Upon success, **globus_gram_protocol_pack_status_reply_with_extensions()** (p. 19) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_MALLOC_FAILED Out of memory

4.8.1.7 int globus_gram_protocol_pack_status_update_message (char * *job_contact*, int *status*, int *failure_code*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM status update message.

The **globus_gram_protocol_pack_status_update_message()** (p. 20) function combines its parameters into a GRAM status update message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_post()** (p. 11) or just frame it by calling **globus_gram_protocol_frame_request()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_status_update_message()** (p. 20) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

job_contact The job contact string associated with the job.

status The job's current **job state** (p. 4).

failure_code The error associated with this job request if the *status* value is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

reply An output parameter which will be set to a new string containing the packed status message. The caller must free this memory by calling free()

replysize An output parameter which will be set to the length of the status message returned in *reply*.

Returns:

Upon success, **globus_gram_protocol_pack_status_update_message()** (p. 20) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters as described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

4.8.1.8 int globus_gram_protocol_pack_status_update_message_with_extensions (char * *job_contact*, int *status*, int *failure_code*, globus_hashtable_t * *extensions*, globus_byte_t ** *reply*, globus_size_t * *replysize*)

Pack a GRAM status update message with extensions.

The **globus_gram_protocol_pack_status_update_message_with_extensions()** (p. 20) function combines its parameters into a GRAM status update message body. The caller may frame and send the resulting message by calling **globus_gram_protocol_post()** (p. 11) or just frame it by calling **globus_gram_protocol_frame_request()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_status_update_message_with_extensions()** (p. 20) function returns the packed message by modifying the *reply* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

job_contact The job contact string associated with the job.

status The job's current **job state** (p. 4).

failure_code The error associated with this job request if the *status* value is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

extensions A pointer to a hash table keyed by extension attribute names with the values being pointers to globus_gram_protocol_extension_t structures.

reply An output parameter which will be set to a new string containing the packed status message. The caller must free this memory by calling free()

replysize An output parameter which will be set to the length of the status message returned in *reply*.

Returns:

Upon success, **globus_gram_protocol_pack_status_update_message_with_extensions()** (p. 20) returns *GLOBUS_SUCCESS* and modifies the *reply* and *replysize* parameters as described above. If an error occurs, an integer error code is returned and the values pointed to by *reply* and *replysize* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

4.8.1.9 int globus_gram_protocol_pack_version_request (char ** *request*, size_t * *requestsize*)

Pack a GRAM version request message.

The **globus_gram_protocol_pack_job_request()** (p. 16) function creates a copy of the GRAM version request. The caller may frame and send the resulting message by calling **globus_gram_protocol_post()** (p. 11) or just frame it by calling **globus_gram_protocol_frame_request()** (p. 7) and send it by some other mechanism. The **globus_gram_protocol_pack_version_request()** (p. 21) function returns the packed message by modifying the *request* parameter to point to a new string containing the message. The caller is responsible for freeing that string.

Parameters:

request An output parameter which will be set to a new string containing the packed version request message. The caller must free this memory by calling free().

requestsize An output parameter which will be populated with the length of the version request message returned in *query*.

Returns:

Upon success, **globus_gram_protocol_pack_job_request()** (p. 16) returns GLOBUS_SUCCESS and modifies the *request* and *requestsize* parameters to point to the values described above. If an error occurs, **globus_gram_protocol_pack_version_request()** (p. 21) returns an integer error code and the values pointed to by *request* and *requestsize* are undefined.

Return values:

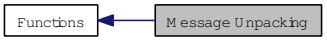
GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

4.9 Message Unpacking

Collaboration diagram for Message Unpacking:



Functions

- int **globus_gram_protocol_unpack_job_request (const globus_byte_t *query, globus_size_t querysize, int *job_state_mask, char **callback_url, char **description)**
- **int globus_gram_protocol_unpack_job_request_reply (const globus_byte_t *reply, globus_size_t replysize, int *status, char **job_contact)**
- **int globus_gram_protocol_unpack_job_request_reply_with_extensions (const globus_byte_t *reply, globus_size_t replysize, int *status, char **job_contact, globus_hashtable_t *extensions)**
- **int globus_gram_protocol_unpack_status_request (const globus_byte_t *query, globus_size_t querysize, char **status_request)**
- **int globus_gram_protocol_unpack_status_reply (const globus_byte_t *reply, globus_size_t replysize, int *job_status, int *failure_code, int *job_failure_code)**
- **int globus_gram_protocol_unpack_status_reply_with_extensions (const globus_byte_t *reply, globus_size_t replysize, globus_hashtable_t *extensions)**
- **int globus_gram_protocol_unpack_status_update_message (const globus_byte_t *reply, globus_size_t replysize, char **job_contact, int *status, int *failure_code)**
- **int globus_gram_protocol_unpack_status_update_message_with_extensions (const globus_byte_t *reply, globus_size_t replysize, globus_hashtable_t *extensions)**
- **void globus_gram_protocol_hash_destroy (globus_hashtable_t *message_hash)**

4.9.1 Function Documentation

4.9.1.1 int globus_gram_protocol_unpack_job_request (const globus_byte_t * *query*, globus_size_t *querysize*, int * *job_state_mask*, char ** *callback_url*, char ** *description*)

Unpack a GRAM Job Request.

The **globus_gram_protocol_unpack_job_request()** (p. 22) function parses the job request message packed in the *query* message and returns copies of the standard message attributes in the *job_state_mask*, *callback_url*, and *description* parameters.

Parameters:

query The unframed job request message to parse.

querysize The length of the job request message string.

job_state_mask A pointer to an integer to be set to the job state mask from the job request.

callback_url A pointer to be set with a copy of the URL of the callback contact to be registered for this job request. The caller must free this memory by calling free().

description A pointer to be set to a copy of the job description RSL string for this job request. The caller must free this memory by calling free().

Returns:

Upon success, **globus_gram_protocol_unpack_job_request()** (p. 22) will return *GLOBUS_SUCCESS* and modify the *job_state_mask*, *callback_url*, and *description* parameters to values extracted from the message in *query*. If an error occurs, an integer error code will be returned and the values of *job_state_mask*, *callback_url*, and *description* will be undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.2 int globus_gram_protocol_unpack_job_request_reply (const globus_byte_t * *reply*, globus_size_t *replysize*, int * *status*, char ** *job_contact*)

Unpack a GRAM reply message.

The **globus_gram_protocol_unpack_job_request_reply()** (p. 23) function parses the reply message packed in the *reply* message and returns copies of the standard message attributes in the *status* and *job_contact* parameters.

Parameters:

reply The unframed job reply message to parse.

replysize The length of the reply string.

status A pointer to an integer to be set to the failure code associated with the job request. This may be GLOBUS_SUCCESS, if the job request was successful.

job_contact A pointer to a string to be set to the job contact string. This may set to NULL if the job request failed. If **globus_gram_protocol_unpack_job_request_reply()** (p. 23) returns GLOBUS_SUCCESS, then the caller must free this string using free().

Returns:

Upon success, **globus_gram_protocol_unpack_job_request_reply()** (p. 23) returns *GLOBUS_SUCCESS* and modifies the *status* and *job_contact* parameters to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *status* and *job_contact* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAN_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.3 int globus_gram_protocol_unpack_job_request_reply_with_extensions (const globus_byte_t * *reply*, globus_size_t *replysize*, int * *status*, char ** *job_contact*, globus_hashtable_t * *extensions*)

Unpack a GRAM reply message, parsing all extensions.

The **globus_gram_protocol_unpack_job_request_reply_with_extensions()** (p. 23) function parses the reply message packed in the *reply* message parameter and returns copies of the standard message attributes in the *status* and *job_contact* parameters, and all other extension attributes in the hashtable pointed to by *extensions*. Each entry in the hashtable will be keyed by the attribute name and the value will be a pointer to a *globus_gram_protocol_extension_t* structure.

Parameters:

status A pointer to an integer to be set to the failure code associated with the job request. This may be GLOBUS_SUCCESS, if the job request was successful.

job_contact A pointer to a string to be set to the job contact string. This may set to NULL if the job request failed. If **globus_gram_protocol_unpack_job_request_reply_with_extensions()** (p. 23) returns GLOBUS_SUCCESS, then the caller must free this string using free().

extensions A pointer to be set to a hash table containing the names and values of all protocol extensions present in the response message. If **globus_gram_protocol_unpack_job_request_reply_with_extensions()** (p. 23) returns GLOBUS_SUCCESS, the caller must free this hash table and its values by calling **globus_gram_protocol_hash_destroy()** (p. 27).

reply The unframed job reply message to parse.

replysize The length of the reply string.

Returns:

Upon success, **globus_gram_protocol_unpack_job_request_reply_with_extensions()** (p. 23) returns *GLOBUS_SUCCESS* and modifies the *status*, *job_contact*, and *extensions* to point to the values described above. If an error occurs, an integer error code is returned and the values pointed to by *status*, *job_contact*, and *extensions* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAN_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.4 int globus_gram_protocol_unpack_status_request (const globus_byte_t * *query*, globus_size_t *querysize*, char ** *status_request*)

Unpack a GRAM query message.

The **globus_gram_protocol_unpack_status_request()** (p. 24) function parses the message packed in the *query* parameter and returns a copy of the message in the *status_request* parameter.

Parameters:

query The unframed query message to parse.

querysize The length of the query string.

status_request A pointer to a string to be set to the query value. The caller must free this string using free().

Returns:

Upon success, **globus_gram_protocol_unpack_status_request()** (p. 24) returns *GLOBUS_SUCCESS* and modifies the *status_request* parameter to point to the value described above. If an error occurs, an integer error code is returned and the value pointed to by *status_request* is undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.5 int globus_gram_protocol_unpack_status_reply (const globus_byte_t * *reply*, globus_size_t *replysize*, int * *job_status*, int * *failure_code*, int * *job_failure_code*)

Unpack a GRAM query reply.

The **globus_gram_protocol_unpack_status_reply()** (p. 25) function parses the message packed in the *reply* parameter and sets the current job state, protocol failure code, and job failure code values in its output parameters.

Parameters:

reply The unframed reply message to parse.

replysize The length of the reply message.

job_status A pointer to an integer to be set to the job's current **job state** (p. 4).

failure_code A pointer to an integer to be set to the failure code associated with the query request. This may be GLOBUS_SUCCESS, if the request was successful.

job_failure_code A pointer to an integer to be set to the failure code for the job, if the *job_status* is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

Returns:

Upon success, **globus_gram_protocol_unpack_status_reply()** (p. 25) returns *GLOBUS_SUCCESS* and modifies the *job_status*, *failure_code*, and *job_failure_code* parameters to point to the value described above. If an error occurs, an integer error code is returned and the values pointed to by *job_status*, *failure_code*, and *job_failure_code* are undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

4.9.1.6 int globus_gram_protocol_unpack_status_reply_with_extensions (const globus_byte_t * *reply*, globus_size_t *replysize*, globus_hashtable_t * *extensions*)

Unpack a GRAM query reply with extensions.

The **globus_gram_protocol_unpack_status_reply_with_extensions()** (p. 25) function parses the message packed in the *reply* parameter, storing all attributes and values in a hash table. The *extensions* parameter is modified to point to that hash table. The caller of **globus_gram_protocol_unpack_status_reply_with_extensions()** (p. 25) must free that hash table by calling **globus_gram_protocol_hash_destroy()** (p. 27).

Parameters:

reply The unframed reply message to parse.

replysize The length of the reply message.

extensions A pointer to be set to a hash table containing the names and values of all protocol attributes present in the reply message. If **globus_gram_protocol_unpack_status_reply_with_extensions()** (p. 25) returns GLOBUS_SUCCESS, the caller must free this hash table and its values by calling **globus_gram_protocol_hash_destroy()** (p. 27).

Returns:

Upon success, **globus_gram_protocol_unpack_status_reply_with_extensions()** (p. 25) returns *GLOBUS_SUCCESS* and modifies the *extensions* parameter to point to the value described above. If an error occurs, an integer error code is returned and the value pointed to by *extensions* is undefined.

Return values:

GLOBUS_SUCCESS Success

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.7 int globus_gram_protocol_unpack_status_update_message (const globus_byte_t * *reply*, globus_size_t *replysize*, char ** *job_contact*, int * *status*, int * *failure_code*)

Unpack a GRAM status update message.

The **globus_gram_protocol_unpack_status_update_message()** (p. 26) function parses the message packed in the *reply* parameter, storing the standard message attribute values in its return parameters *job_contact*, *status*, and *failure_code*. The caller is responsible for freeing the *job_contact* value.

Parameters:

reply The unframed reply message to parse.

replysize The length of the reply message.

job_contact An output parameter to be set to the job contact string. If **globus_gram_protocol_unpack_status_update_message()** (p. 26) returns GLOBUS_SUCCESS, then the caller must free this string using free().

status An output parameter to be set to the integer value of the job's current **job state** (p. 4).

failure_code An output parameter to be set to the integer failure code for the job if the *job_status* is GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED.

Returns:

Upon success, **globus_gram_protocol_unpack_status_update_message()** (p. 26) returns *GLOBUS_SUCCESS* and modifies the *job_contact*, *status*, and *failure_code* parameters as described above. If an error occurs, an integer error code is returned and the values pointed to by the *job_contact*, *status*, and *failure_code* parameters are undefined.

Return values:

GLOBUS_SUCCESS Sucess

GLOBUS_GRAM_PROTOCOL_ERROR_NULL_PARAMETER Null parameter

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Out of memory

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.8 int globus_gram_protocol_unpack_status_update_message_with_extensions (const globus_byte_t * *reply*, globus_size_t *replysize*, globus_hashtable_t * *extensions*)

Unpack a GRAM status update message with extensions.

The **globus_gram_protocol_unpack_status_update_message_with_extensions()** (p. 26) function parses the message packed in the *reply* parameter, storing the message attribute values in its return parameter *extensions*. The caller is responsible for freeing the *extensions* hash table by calling **globus_gram_protocol_hash_destroy()** (p. 27).

Parameters:

reply The unframed reply message to parse.

replysize The length of the reply message.

extensions An output parameter which will be initialized to a hashtable containing the message attributes. The caller must destroy this hashtable calling **globus_gram_protocol_hash_destroy()** (p. 27).

Returns:

Upon success, **globus_gram_protocol_unpack_status_update_message_with_extensions()** (p. 26) returns *GLOBUS_SUCCESS* and modifies the *extensions* parameter as described above. If an error occurs, an integer error code is returned and the value pointed to by the *extensions* parameters is undefined.

Return values:

GLOBUS_SUCCESS Sucess

GLOBUS_GRAM_PROTOCOL_ERROR_HTTP_UNPACK_FAILED Unpack failed

GLOBUS_GRAM_PROTOCOL_ERROR_MALLOC_FAILED Malloc failed

GLOBUS_GRAM_PROTOCOL_ERROR_VERSION_MISMATCH Version mismatch

4.9.1.9 void globus_gram_protocol_hash_destroy (globus_hashtable_t * *message_hash*)

Destroy message attribute hash.

Parameters:

message_hash Hashtable of globus_gram_protocol_extension_t * values to destroy

5 globus gram protocol Page Documentation

5.1 GRAM Protocol Definition

The GRAM Protocol is used to handle communication between the Gatekeeper, Job Manager, and GRAM Clients.

The protocol is based on a subset of the HTTP/1.1 protocol, with a small set of message types and responses sent as the body of the HTTP requests and responses. This document describes GRAM Protocol version 2.

Framing GRAM messages are framed in HTTP/1.1 messages. However, only a small subset of the HTTP specification is used or understood by the GRAM system. All GRAM requests are HTTP POST messages. Only the following HTTP headers are understood:

- Host
- Content-Type (set to "application/x-globus-gram" in all cases)
- Content-Length
- Connection (set to "close" in all HTTP responses)

Only the following status codes are supported in response's HTTP Status-Lines:

- 200 OK
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 400 Bad Request

Message Format All messages use the carriage return (ASCII value 13) followed by line feed (ASCII value 10) sequence to delimit lines. In all cases, a blank line separates the HTTP header from the message body. All **application/x-globus-gram** message bodies consist of attribute names followed by a colon, a space, and then the value of the attribute. When the value may contain a newline or double-quote character, a special escaping rule is used to encapsulate the complete string. This encapsulation consists of surrounding the string with double-quotes, and escaping all double-quote and backslash characters within the string with a backslash. All other characters are sent without modification. For example, the string

```
rsl: &( executable = "/bin/echo" )
 ( arguments = "hello" )
```

becomes

```
rsl: "&( executable = \"bin/echo\" )
  (arguments = \"hello\" )"
```

This is the only form of quoting which **application/x-globus-gram** messages support. Use of % HEX HEX escapes (such as seen in URL encodings) is not meaningful for this protocol.

Message Types

Ping Request A ping request is used to verify that the gatekeeper is configured properly to handle a named service. The ping request consists of the following:

```
POST ping/ job-manager-name HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
```

The values of the message-specific strings are

job-manager-name The name of the service to have the gatekeeper check. The service name corresponds to one of the gatekeeper's configured grid-services, and is usually of the form "jobmanager-scheduler-type".

host-name The name of the host on which the gatekeeper is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Job Request A job request is used to scheduler a job remotely using GRAM. The ping request consists of the HTTP framing described above with the request-URI consisting of *job-manager-name*, where *job-manager name* is the name of the service to use to schedule the job. The format of a job request message consists of the following:

```
POST  job-manager-name[@ user-name] HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size
```

```
protocol-version:  version
job-state-mask:  mask
callback-url:  callback-contact
rsl:  rsl-description
```

The values of the emphasized text items are as below:

job-manager-name The name of the service to submit the job request to. The service name corresponds to one of the gatekeeper's configured grid-services, and is usually of the form "jobmanager-scheduler-type".

user-name Starting with GT4.0, a client may request that a certain account by used by the gatekeeper to start the job manager. This is done optionally by appending the @ symbol and the local user name that the job should be run as to the *job-manager-name*. If the @ and username are not present, then the first grid map entry will be used. If the client credential is not authorized in the grid map to use the specified account, an authorization error will occur in the gatekeeper.

host-name The name of the host on which the gatekeeper is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

mask An integer representation of the job state mask. This value is obtained from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. These meanings of the various job state values are defined in the **GRAM Protocol API documentation** (p. 4).

callback-contact A https URL which defines a GRAM protocol listener which will receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined **below** (p. 31).

rsl-description A quoted string containing the RSL description of the job request.

Status Request A status request is used by a GRAM client to get the current job state of a running job. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"status"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Callback Register Request A callback register request is used by a GRAM client to register a new callback contact to receive GRAM job state updates. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"register <em>mask</em> <em>callback-contact</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

mask An integer representation of the job state mask. This value is obtained from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. These meanings of the various job state values are defined in the **GRAM Protocol API documentation** (p. 4).

callback-contact A https URL which defines a GRAM protocol listener which will receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined **below** (p. 31).

Callback Unregister Request A callback unregister request is used by a GRAM client to request that the job manager no longer send job state updates to the specified callback contact. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"unregister <em>callback-contact</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

callback-contact A https URL which defines a GRAM protocol listener which should no longer receive job state updates. The from a bitwise-OR of the job state values which the client wishes to receive job status callbacks about. The job status update messages are defined **below** (p. 31).

Job Cancel Request A job cancel request is used by a GRAM client to request that the job manager terminate a job. This type of message can only be sent to a job manager's job-contact (as returned in the reply to a job request message). The format of a job request message consists of the following:

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"cancel"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

Job Signal Request A job signal request is used by a GRAM client to request that the job manager process a signal for a job. The arguments to the various signals are discussed in the **globus_gram_protocol_job_signal_t** (p. 2) documentation.

```
POST  job-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"<em>signal</em>"
```

The values of the emphasized text items are as below:

job-contact The job contact string returned in a response to a job request message, or determined by querying the MDS system.

host-name The name of the host on which the job manager is running. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

signal A quoted string containing the signal number and it's parameters.

Job State Updates A job status update message is sent by the job manager to all registered callback contacts when the job's status changes. The format of the job status update messages is as follows:

```
POST  callback-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
job-manager-url:  job-contact
status:  status-code
failure-code:  failure-code
```

The values of the emphasized text items are as below:

callback-contact The callback contact string registered with the job manager either by being passed as the *callback-contact* in a job request message or in a callback register message.

host-name The host part of the callback-contact URL. This exists only for compatibility with the HTTP/1.1 protocol.

message-size The length of the content of the message, not including the HTTP/1.1 header.

version The version of the GRAM protocol which is being used. For the protocol defined in this document, the value must be the string "2".

job-contact The job contact of the job which has changed states.

Proxy Delegation A proxy delegation message is sent by the client to the job manager to initiate a delegation handshake to generate a new proxy credential for the job manager. This credential is used by the job manager or the job when making further secured connections. The format of the delegation message is as follows:

```
POST  callback-contact HTTP/1.1
Host:  host-name
Content-Type: application/x-globus-gram
Content-Length:  message-size

protocol-version:  version
"renew"
```

If a successful (200) reply is sent in response to this message, then the client will procede with a GSI delegation handshake. The tokens in this handshake will be framed with a 4 byte big-endian token length header. The framed tokens will then be wrapped using the GLOBUS_IO_SECURE_CHANNEL_MODE_SSL_WRAP wrapping mode. The job manager will frame response tokens in the same manner. After the job manager receives its final delegation token, it will respond with another response message that indicates whether the delegation was processed or not. This response message is a standard GRAM response message.

Note on Security Attributes The following security attributes are needed to communicate with the Gatekeeper:

- Authentication must be done using GSSAPI mutual authentication
- Messages must be wrapped with support for the delegation message. When using Globus I/O, this is accomplished by using the the GLOBUS_IO_SECURE_CHANNEL_MODE_GSI_WRAP wrapping mode.

Changes 2004-08-11 Added information about gridmap choosing

Index