GT 5.2.2 Component Guide
to Public Interfaces: GRAM5
GT 5.2.2 Component Guide to Public Interfaces: GRAM5
Table of Contents

1. APIs ........................................................................................................................................... 1
   1. Programming Model Overview ............................................................................................... 1
2. RSL Specification v1.1 .................................................................................................................. 21
   1. RSL Syntax Overview ................................................................................................................ 21
   2. RSL Tokenization Overview ................................................................................................... 22
   3. RSL Substitution Semantics ..................................................................................................... 23
   4. RSL Attribute Summary ........................................................................................................... 23
   5. Simple RSL Examples .............................................................................................................. 27
   6. RSL grammar and tokenization rules ....................................................................................... 28
3. RSL Validation File Specification ................................................................................................. 30
   1. RVF Syntax Overview .............................................................................................................. 30
I. GRAM5 Commands ....................................................................................................................... 35
   globus-fork-starter ....................................................................................................................... 36
   globus-gatekeeper-admin .......................................................................................................... 38
   globus-gatekeeper ....................................................................................................................... 39
   globus-gram-audit ....................................................................................................................... 42
   globus-job-cancel ......................................................................................................................... 43
   globus-job-clean ......................................................................................................................... 44
   globus-job-get-output .................................................................................................................. 45
   globus-job-manager ..................................................................................................................... 47
   globus-job-run .............................................................................................................................. 52
   globus-job-status ......................................................................................................................... 55
   globus-job-submit ......................................................................................................................... 57
   globus-personal-gatekeeper .......................................................................................................... 60
   globus-rvf-check .......................................................................................................................... 62
   globus-rvf-edit .............................................................................................................................. 63
   globus-scheduler-event-generator-admin ....................................................................................... 64
   globus-scheduler-event-generator ............................................................................................... 65
   globusrun .................................................................................................................................... 66
4. Configuring GRAM5 ....................................................................................................................... 70
   1. Gatekeeper Configuration ....................................................................................................... 70
   2. Scheduler Event Generator Configuration ............................................................................... 71
   3. Job Manager Configuration ..................................................................................................... 72
   4. LRM Adapter Configuration .................................................................................................... 73
   5. Auditing ................................................................................................................................... 75
   6. RSL Attributes ......................................................................................................................... 76
5. Job description .............................................................................................................................. 77
6. Semantics and syntax of protocols .............................................................................................. 78
   1. GRAM5 Protocol ..................................................................................................................... 78
A. Errors ........................................................................................................................................... 86
Glossary ............................................................................................................................................ 99
List of Figures

6.1. GRAM State Transitions ........................................................................................................... 85
List of Tables

1.1. GRAM Client APIs ........................................................................................................ 1
4.1. Gatekeeper Configuration Path .................................................................................... 70
4.2. Scheduler Event Generator Configuration Path .......................................................... 71
4.3. Job Manager Configuration Path .................................................................................. 72
4.4. GRAM5 Log Levels ..................................................................................................... 72
4.5. LRM Adapter Configuration Path .................................................................................. 73
4.6. GRAM Audit Configuration Path .................................................................................. 75
4.7. Audit Configuration Attributes .................................................................................... 76
6.1. GRAM Job States .......................................................................................................... 84
A.1. GRAM5 Errors ............................................................................................................. 86
List of Examples

2.1. Quoted Literal Examples ................................................................. 23
2.2. GRAM5 Job Request Examples ......................................................... 28
Chapter 1. APIs

1. Programming Model Overview

1.1. C API Documentation Links

Table 1.1. GRAM Client APIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAM Protocol¹</td>
<td>Low-level functions for processing GRAM protocol messages. Symbolic constants for RSL attributes, signals, and job states.</td>
</tr>
<tr>
<td>GRAM Client¹</td>
<td>Functions for submitting job requests, sending signals, and listening for job state updates.</td>
</tr>
<tr>
<td>RSL¹</td>
<td>Functions for parsing and manipulating job specifications in the RSL language.</td>
</tr>
<tr>
<td>Scheduler Event Generator¹</td>
<td>Functions for generating and parsing LRM-independent job state change events.</td>
</tr>
</tbody>
</table>

1.2. GRAM5 Perl API Reference

¹ http://www.globus.org/api/c-globus-5.2.2/globus_gram_protocol/html/
¹ http://www.globus.org/api/c-globus-5.2.2/globus_gram_client/html/
¹ http://www.globus.org/api/c-globus-5.2.2/globus_rsl/html/
¹ http://www.globus.org/api/c-globus-5.2.2/globus_scheduler_event_generator/html/
Name
Globus::GRAM::Error — GRAM Protocol Error Constants

DESCRIPTION
The Globus::GRAM::Error module defines symbolic names for the Error constants in the GRAM Protocol.

The Globus::GRAM::Error module methods return an object consisting of an integer error code, and (optionally) a string explaining the error.

Methods

```perl
$error = new Globus::GRAM::Error($number, $string);
Create a new error object with the given error number and string description.
This is called by the error-specific factory methods described below.

$error->string() Return the error string associated with a Globus::GRAM::Error object.

$error->value() Return the integer error code associated with a Globus::GRAM::Error object.

$error = Globus::GRAM::Error::PARAMETER_NOT_SUPPORTED();
Create a new PARAMETER_NOT_SUPPORTED GRAM error.

$error = Globus::GRAM::Error::INVALID_REQUEST();
Create a new INVALID_REQUEST GRAM error.

$error = Globus::GRAM::Error::NO_RESOURCES();
Create a new NO_RESOURCES GRAM error.

$error = Globus::GRAM::Error::BAD_DIRECTORY();
Create a new BAD_DIRECTORY GRAM error.

$error = Globus::GRAM::Error::EXECUTABLE_NOT_FOUND();
Create a new EXECUTABLE_NOT_FOUND GRAM error.

$error = Globus::GRAM::Error::INSUFFICIENT_FUNDS();
Create a new INSUFFICIENT_FUNDS GRAM error.

$error = Globus::GRAM::Error::AUTHORIZATION();
Create a new AUTHORIZATION GRAM error.

$error = Globus::GRAM::Error::USER_CANCELLED();
Create a new USER_CANCELLED GRAM error.

$error = Globus::GRAM::Error::SYSTEM_CANCELLED();
Create a new SYSTEM_CANCELLED GRAM error.

$error = Globus::GRAM::Error::PROTOCOL_FAILED();
Create a new PROTOCOL_FAILED GRAM error.

$error = Globus::GRAM::Error::STDIN_NOT_FOUND();
Create a new STDIN_NOT_FOUND GRAM error.

$error = Globus::GRAM::Error::CONNECTION_FAILED();
Create a new CONNECTION_FAILED GRAM error.
```
Create a new INVALID_MAXTIME GRAM error.
Globus::GRAM::Error::INVALID_MAXTIME()

Create a new INVALID_COUNT GRAM error.
Globus::GRAM::Error::INVALID_COUNT()

Create a new NULL_SPECIFICATION_TREE GRAM error.
Globus::GRAM::Error::NULL_SPECIFICATION_TREE()

Create a new JM_FAILED_ALLOW_ATTACH GRAM error.
Globus::GRAM::Error::JM_FAILED_ALLOW_ATTACH()

Create a new JOB_EXECUTION_FAILED GRAM error.
Globus::GRAM::Error::JOB_EXECUTION_FAILED()

Create a new INVALID_PARADYN GRAM error.
Globus::GRAM::Error::INVALID_PARADYN()

Create a new INVALID_JOBTYPE GRAM error.
Globus::GRAM::Error::INVALID_JOBTYPE()

Create a new INVALID_GRAM_MYJOB GRAM error.
Globus::GRAM::Error::INVALID_GRAM_MYJOB()

Create a new BAD_SCRIPT_ARG_FILE GRAM error.
Globus::GRAM::Error::BAD_SCRIPT_ARG_FILE()

Create a new ARG_FILE_CREATION_FAILED GRAM error.
Globus::GRAM::Error::ARG_FILE_CREATION_FAILED()

Create a new INVALID_JOBSTATE GRAM error.
Globus::GRAM::Error::INVALID_JOBSTATE()

Create a new INVALID_SCRIPT_REPLY GRAM error.
Globus::GRAM::Error::INVALID_SCRIPT_REPLY()

Create a new INVALID_SCRIPT_STATUS GRAM error.
Globus::GRAM::Error::INVALID_SCRIPT_STATUS()

Create a new JOBTYPE_NOT_SUPPORTED GRAM error.
Globus::GRAM::Error::JOBTYPE_NOT_SUPPORTED()

Create a new UNIMPLEMENTED GRAM error.
Globus::GRAM::Error::UNIMPLEMENTED()

Create a new TEMP_SCRIPT_FILE_FAILED GRAM error.
Globus::GRAM::Error::TEMP_SCRIPT_FILE_FAILED()

Create a new USER_PROXY_NOT_FOUND GRAM error.
Globus::GRAM::Error::USER_PROXY_NOT_FOUND()

Create a new OPENING_USER_PROXY GRAM error.
Globus::GRAM::Error::OPENING_USER_PROXY()

Create a new JOB_CANCEL_FAILED GRAM error.
Globus::GRAM::Error::JOB_CANCEL_FAILED()
$error = create a new MALLOC_FAILED GRAM error.
Globus::GRAM::Error::MALLOC_FAILED()

$error = create a new DUCT_INIT_FAILED GRAM error.
Globus::GRAM::Error::DUCT_INIT_FAILED()

$error = create a new DUCT_LSP_FAILED GRAM error.
Globus::GRAM::Error::DUCT_LSP_FAILED()

$error = create a new INVALID_HOST_COUNT GRAM error.
Globus::GRAM::Error::INVALID_HOST_COUNT()

$error = create a new UNSUPPORTED_PARAMETER GRAM error.
Globus::GRAM::Error::UNSUPPORTED_PARAMETER()

$error = create a new INVALID_QUEUE GRAM error.
Globus::GRAM::Error::INVALID_QUEUE()

$error = create a new INVALID_PROJECT GRAM error.
Globus::GRAM::Error::INVALID_PROJECT()

$error = create a new RSL_EVALUATION_FAILED GRAM error.
Globus::GRAM::Error::RSL_EVALUATION_FAILED()

$error = create a new BAD_RSL_ENVIRONMENT GRAM error.
Globus::GRAM::Error::BAD_RSL_ENVIRONMENT()

$error = create a new DRYRUN GRAM error.
Globus::GRAM::Error::DRYRUN()

$error = create a new ZERO_LENGTH_RSL GRAM error.
Globus::GRAM::Error::ZERO_LENGTH_RSL()

$error = create a new STAGING_EXECUTABLE GRAM error.
Globus::GRAM::Error::STAGING_EXECUTABLE()

$error = create a new STAGING_STDIN GRAM error.
Globus::GRAM::Error::STAGING_STDIN()

$error = create a new INVALID_JOB_MANAGER_TYPE GRAM error.
Globus::GRAM::Error::INVALID_JOB_MANAGER_TYPE()

$error = create a new BAD_ARGUMENTS GRAM error.
Globus::GRAM::Error::BAD_ARGUMENTS()

$error = create a new GATEKEEPER_MISCONFIGURED GRAM error.
Globus::GRAM::Error::GATEKEEPER_MISCONFIGURED()

$error = create a new BAD_RSL GRAM error.
Globus::GRAM::Error::BAD_RSL()

$error = create a new VERSION_MISMATCH GRAM error.
Globus::GRAM::Error::VERSION_MISMATCH()

$error = create a new RSL_ARGUMENTS GRAM error.
Globus::GRAM::Error::RSL_ARGUMENTS()
$error = $error = $error = $error = $error =
Globus::GRAM::Error::RSL_COUNT()
Globus::GRAM::Error::RSL_COUNT()**
Globus::GRAM::Error::RSL_DIRECTORY()
Globus::GRAM::Error::RSL_DIRECTORY()
Globus::GRAM::Error::RSL_DRYRUN()
Globus::GRAM::Error::RSL_DRYRUN()
Globus::GRAM::Error::RSL_ENVIRONMENT()
Globus::GRAM::Error::RSL_ENVIRONMENT()
Globus::GRAM::Error::RSL_EXECUTABLE()
Globus::GRAM::Error::RSL_EXECUTABLE()
Globus::GRAM::Error::RSL_HOST_COUNT()
Globus::GRAM::Error::RSL_HOST_COUNT()
Globus::GRAM::Error::RSL_JOBTYPE()
Globus::GRAM::Error::RSL_JOBTYPE()
Globus::GRAM::Error::RSL_MAXTIME()
Globus::GRAM::Error::RSL_MAXTIME()
Globus::GRAM::Error::RSL_MYJOB()
Globus::GRAM::Error::RSL_MYJOB()
Globus::GRAM::Error::RSL_PARADYN()
Globus::GRAM::Error::RSL_PARADYN()
Globus::GRAM::Error::RSL_PROJECT()
Globus::GRAM::Error::RSL_PROJECT()
Globus::GRAM::Error::RSL_QUEUE()
Globus::GRAM::Error::RSL_QUEUE()
Globus::GRAM::Error::RSL_STDERR()
Globus::GRAM::Error::RSL_STDERR()
Globus::GRAM::Error::RSL_STDIN()
Globus::GRAM::Error::RSL_STDIN()
Globus::GRAM::Error::RSL_STDOUT()
Globus::GRAM::Error::RSL_STDOUT()
Globus::GRAM::Error::OPENING_JOBMANAGER_SCRIPT()
Globus::GRAM::Error::OPENING_JOBMANAGER_SCRIPT()
Globus::GRAM::Error::CREATINGPIPE()
Globus::GRAM::Error::CREATING_PIPE()
Globus::GRAM::Error::FCNTL_FAILED()
Globus::GRAM::Error::FCNTL_FAILED()
Globus::GRAM::Error::STDOUT_FILENAME_FAILED()
Globus::GRAM::Error::STDOUT_FILENAME_FAILED()
APIs

$error =$ error = Globus::GRAM::Error::STDERR_FILENAME_FAILED()
Create a new STDERR_FILENAME_FAILED GRAM error.

$error = Globus::GRAM::Error::FORKING_EXECUTABLE()
Create a new FORKING_EXECUTABLE GRAM error.

$error = Globus::GRAM::Error::EXECUTABLE_PERMISSIONS()
Create a new EXECUTABLE_PERMISSIONS GRAM error.

$error = Globus::GRAM::Error::OPENING_STDOUT()
Create a new OPENING_STDOUT GRAM error.

$error = Globus::GRAM::Error::OPENING_STDERR()
Create a new OPENING_STDERR GRAM error.

$error = Globus::GRAM::Error::OPENING_CACHE_USER_PROXY()
Create a new OPENING_CACHE_USER_PROXY GRAM error.

$error = Globus::GRAM::Error::OPENING_CACHE()
Create a new OPENING_CACHE GRAM error.

$error = Globus::GRAM::Error::INSERTING_CLIENT_CONTACT()
Create a new INSERTING_CLIENT_CONTACT GRAM error.

$error = Globus::GRAM::Error::CLIENT_CONTACT_NOT_FOUND()
Create a new CLIENT_CONTACT_NOT_FOUND GRAM error.

$error = Globus::GRAM::Error::CONTACTING_JOB_MANAGER()
Create a new CONTACTING_JOB_MANAGER GRAM error.

$error = Globus::GRAM::Error::INVALID_JOB_CONTACT()
Create a new INVALID_JOB_CONTACT GRAM error.

$error = Globus::GRAM::Error::UNDEFINED_EXE()
Create a new UNDEFINED_EXE GRAM error.

$error = Globus::GRAM::Error::CONDOR_ARCH()
Create a new CONDOR_ARCH GRAM error.

$error = Globus::GRAM::Error::CONDOR_OS()
Create a new CONDOR_OS GRAM error.

$error = Globus::GRAM::Error::RSL_MIN_MEMORY()
Create a new RSL_MIN_MEMORY GRAM error.

$error = Globus::GRAM::Error::RSL_MAX_MEMORY()
Create a new RSL_MAX_MEMORY GRAM error.

$error = Globus::GRAM::Error::INVALID_MIN_MEMORY()
Create a new INVALID_MIN_MEMORY GRAM error.

$error = Globus::GRAM::Error::INVALID_MAX_MEMORY()
Create a new INVALID_MAX_MEMORY GRAM error.

$error = Globus::GRAM::Error::HTTP_FRAME_FAILED()
Create a new HTTP_FRAME_FAILED GRAM error.
$error = $this->createError(Globus::GRAM::Error::HTTP_UNFRAME_FAILED());

$error = $this->createError(Globus::GRAM::Error::HTTP_PACK_FAILED());

$error = $this->createError(Globus::GRAM::Error::HTTP_UNPACK_FAILED());

$error = $this->createError(Globus::GRAM::Error::INVALID_JOB_QUERY());

$error = $this->createError(Globus::GRAM::Error::SERVICE_NOT_FOUND());

$error = $this->createError(Globus::GRAM::Error::JOB_QUERY_DENIAL());

$error = $this->createError(Globus::GRAM::Error::CALLBACK_NOT_FOUND());

$error = $this->createError(Globus::GRAM::Error::BAD_GATEKEEPER_CONTACT());

$error = $this->createError(Globus::GRAM::Error::POE_NOT_FOUND());

$error = $this->createError(Globus::GRAM::Error::MPIRUN_NOT_FOUND());

$error = $this->createError(Globus::GRAM::Error::RSL_START_TIME());

$error = $this->createError(Globus::GRAM::Error::RSL_RESERVATION_HANDLE());

$error = $this->createError(Globus::GRAM::Error::RSL_MAX_WALL_TIME());

$error = $this->createError(Globus::GRAM::Error::INVALID_MAX_WALL_TIME());

$error = $this->createError(Globus::GRAM::Error::RSL_MAX_CPU_TIME());

$error = $this->createError(Globus::GRAM::Error::INVALID_MAX_CPU_TIME());

$error = $this->createError(Globus::GRAM::Error::JM_SCRIPT_NOT_FOUND());

$error = $this->createError(Globus::GRAM::Error::JM_SCRIPT_PERMISSIONS());

$error = $this->createError(Globus::GRAM::Error::SIGNALING_JOB());
$error = Globus::GRAM::Error::UNKNOWN_SIGNAL_TYPE()
Create a new UNKNOWN_SIGNAL_TYPE GRAM error.

$error = Globus::GRAM::Error::GETTING_JOBID()
Create a new GETTING_JOBID GRAM error.

$error = Globus::GRAM::Error::WAITING_FOR_COMMIT()
Create a new WAITING_FOR_COMMIT GRAM error.

$error = Globus::GRAM::Error::COMMIT_TIMED_OUT()
Create a new COMMIT_TIMED_OUT GRAM error.

$error = Globus::GRAM::Error::RSL_SAVE_STATE()
Create a new RSL_SAVE_STATE GRAM error.

$error = Globus::GRAM::Error::RSL_RESTART()
Create a new RSL_RESTART GRAM error.

$error = Globus::GRAM::Error::RSL_TWO_PHASE_COMMIT()
Create a new RSL_TWO_PHASE_COMMIT GRAM error.

$error = Globus::GRAM::Error::INVALID_TWO_PHASE_COMMIT()
Create a new INVALID_TWO_PHASE_COMMIT GRAM error.

$error = Globus::GRAM::Error::RSL_STDOUT_POSITION()
Create a new RSL_STDOUT_POSITION GRAM error.

$error = Globus::GRAM::Error::INVALID_STDOUT_POSITION()
Create a new INVALID_STDOUT_POSITION GRAM error.

$error = Globus::GRAM::Error::RSL_STDERR_POSITION()
Create a new RSL_STDERR_POSITION GRAM error.

$error = Globus::GRAM::Error::INVALID_STDERR_POSITION()
Create a new INVALID_STDERR_POSITION GRAM error.

$error = Globus::GRAM::Error::RESTART_FAILED()
Create a new RESTART_FAILED GRAM error.

$error = Globus::GRAM::Error::NO_STATE_FILE()
Create a new NO_STATE_FILE GRAM error.

$error = Globus::GRAM::Error::READING_STATE_FILE()
Create a new READING_STATE_FILE GRAM error.

$error = Globus::GRAM::Error::WRITING_STATE_FILE()
Create a new WRITING_STATE_FILE GRAM error.

$error = Globus::GRAM::Error::OLD_JM_ALIVE()
Create a new OLD_JM_ALIVE GRAM error.

$error = Globus::GRAM::Error::TTL_EXPIRED()
Create a new TTL_EXPIRED GRAM error.

$error = Globus::GRAM::Error::SUBMIT_UNKNOWN()
Create a new SUBMIT_UNKNOWN GRAM error.
$error = 
Globus::GRAM::Error::RSL_REMOTE_IO_URL()

Create a new RSL_REMOTE_IO_URL GRAM error.

$error = 
Globus::GRAM::Error::WRITING_REMOTE_IO_URL()

Create a new WRITING_REMOTE_IO_URL GRAM error.

$error = 
Globus::GRAM::Error::STDIO_SIZE()

Create a new STDIO_SIZE GRAM error.

$error = 
Globus::GRAM::Error::JM_STOPPED()

Create a new JM_STOPPED GRAM error.

$error = 
Globus::GRAM::Error::USER_PROXY_EXPIRED()

Create a new USER_PROXY_EXPIRED GRAM error.

$error = 
Globus::GRAM::Error::JOB_UNSUBMITTED()

Create a new JOB_UNSUBMITTED GRAM error.

$error = 
Globus::GRAM::Error::INVALID_COMMIT()

Create a new INVALID_COMMIT GRAM error.

$error = 
Globus::GRAM::Error::RSL_SCHEDULER_SPECIFIC()

Create a new RSL_SCHEDULER_SPECIFIC GRAM error.

$error = 
Globus::GRAM::Error::STAGE_IN_FAILED()

Create a new STAGE_IN_FAILED GRAM error.

$error = 
Globus::GRAM::Error::INVALID_SCRATCH()

Create a new INVALID_SCRATCH GRAM error.

$error = 
Globus::GRAM::Error::RSL_FILE_STAGE_IN()

Create a new RSL_FILE_STAGE_IN GRAM error.

$error = 
Globus::GRAM::Error::RSL_FILE_STAGE_IN_SHARED()

Create a new RSL_FILE_STAGE_IN_SHARED GRAM error.

$error = 
Globus::GRAM::Error::RSL_FILE_STAGE_OUT()

Create a new RSL_FILE_STAGE_OUT GRAM error.

$error = 
Globus::GRAM::Error::RSL_GASS_CACHE()

Create a new RSL_GASS_CACHE GRAM error.

$error = 
Globus::GRAM::Error::RSL_FILE_CLEANUP()

Create a new RSL_FILE_CLEANUP GRAM error.
$error = $error = $error = $error = $error =
Globus::GRAM::Error::RSL_SCRATCH()
Globus::GRAM::Error::INVALID_SCHEDULER_SPECIFIC()
Globus::GRAM::Error::UNDEFINED_ATTRIBUTE()
Globus::GRAM::Error::INVALID_CACHE()
Globus::GRAM::Error::INVALID_SAVE_STATE()
Globus::GRAM::Error::INVALID_SAVE_STATE()
Globus::GRAM::Error::OPENING_VALIDATION_FILE()
Globus::GRAM::Error::READING_VALIDATION_FILE()
Globus::GRAM::Error::RSL_PROXY_TIMEOUT()
Globus::GRAM::Error::RSL_PROXY_TIMEOUT()
Globus::GRAM::Error::STAGE_OUT_FAILED()
Globus::GRAM::Error::JOB_CONTACT_NOT_FOUND()
Globus::GRAM::Error::DELEGATION_FAILED()
Globus::GRAM::Error::LOCKING_STATE_LOCK_FILE()
Globus::GRAM::Error::INVALID_ATTR()
Globus::GRAM::Error::NULL_PARAMETER()
Globus::GRAM::Error::STILL_STREAMING()
Globus::GRAM::Error::AUTHORIZATION_DENIED()
Globus::GRAM::Error::AUTHORIZATION_SYSTEM_FAILURE()
Globus::GRAM::Error::AUTHORIZATION_DENIED_JOB_ID()
$error = Globus::GRAM::Error::AUTHORIZATION_DENIED_EXECUTABLE()

Create a new AUTHORIZATION_DENIED_EXECUTABLE GRAM error.

$error = Globus::GRAM::Error::RSL_USER_NAME()

Create a new RSL_USER_NAME GRAM error.

$error = Globus::GRAM::Error::INVALID_USER_NAME()

Create a new INVALID_USER_NAME GRAM error.

$error = Globus::GRAM::Error::LAST()

Create a new LAST GRAM error.
Name
Globus::GRAM::JobDescription — GRAM Job Description

Synopsis

use Globus::GRAM::JobDescription;

$hash = { executable => [ '/bin/echo' ], arguments => [ 'hello', 'world' ];
$description = new Globus::GRAM::JobDescription($filename);
$description = new Globus::GRAM::JobDescription($hash);
$executable = $description->executable();
$description->add($new_attribute, $new_value);
$description->save();
$description->save($filename);
$description->print_recursive($file_handle);

DESCRIPTION

This object contains the parameters of a job request in a simple object wrapper. The object may be queried to determine the value of any RSL parameter, may be updated with new parameters, and may be saved in the filesystem for later use.

Methods

new

A JobDescription is constructed from a file consisting of a Perl hash of parameter => array mappings. Every value in the Job Description is stored internally as an array, even single literals, similar to the way an RSL tree is parsed in C. An example of such a file is

$description =
{ 
  executable => [ '/bin/echo' ],
  arguments => [ 'hello', 'world' ],
  environment => [ 
    [ 
      'GLOBUSGRAMJOBCONTACT',
      'https://globus.org:1234/2345/4332'
    ]
  ]
};

which corresponds to the rsl fragment

$(executable = /bin/echo) 
(arguments = hello world) 
(environment = 
  (GLOBUSGRAMJOBCONTACT 'https://globus.org:1234/2345/4332')
)

When the library_path RSL attribute is specified, this object modifies the environment RSL attribute value to append its value to any system specific variables.
$description->add('name', $value); Add a parameter to a job description. The parameter will be normalized internally so that the access methods described below will work with this new parameter. As an example,

```
 setDescription->add('new_attribute', $new_value)
```
will create a new attribute in the JobDescription, which can be accessed by calling the $description->new_attribute() method.

$value $description->get('name'); Get a parameter from a job description. As an example,

```
 setDescription->get('attribute')
```
will return the appropriate attribute in the JobDescription by name.

$description->save([$filename]) Save the JobDescription, including any added parameters, to the file named by $filename if present, or replacing the file used in constructing the object.

$description->print_recursive($file_handle) Write the value of the job description object to the file handle specified in the argument list.

$description->parameter() For any parameter defined in the JobDescription can be accessed by calling the method named by the parameter. The method names are automatically created when the JobDescription is created, and may be invoked with arbitrary SillyCaps or underscores. That is, the parameter gram_myjob may be accessed by the GramMyJob, grammyjob, or gram_my_job method names (and others).

If the attributes does not in this object, then undef will be returned.

In a list context, this returns the list of values associated with an attribute.

In a scalar context, if the attribute's value consist of a single literal, then that literal will be returned, otherwise undef will be returned.

For example, from a JobDescription called $d constructed from a description file containing

```
{
    executable => [ '/bin/echo' ],
    arguments  => [ 'hello', 'world' ]
}
```

The following will hold:

```
$executable = $d->executable()    # '/bin/echo'
$arguments = $d->arguments()      # undef
@executable = $d->executable()    # ('/bin/echo')
@arguments = $d->arguments()      # ('hello', 'world')

$not_present = $d->not_present()  # undef
@not_present = $d->not_present()  # ()
```

To test for existence of a value:

```
@not_present = $d->not_present()
print "Not defined\n" if(!defined($not_present[0]));
```
Name
Globus::GRAM::JobManager — Base class for all Job Manager scripts

Synopsis

$manager = new Globus::GRAM::JobManager($job_description);

$manager->log("Starting new operation");
$manager->nfssync($fileobj, $createflag);
$manager->respond($hashref);
$hashref = $manager->submit();
$hashref = $manager->poll();
$hashref = $manager->cancel();
$hashref = $manager->signal();
$hashref = $manager->make_scratchdir();
$hashref = $manager->remove_scratchdir();
$hashref = $manager->rewrite_urls();
$hashref = $manager->stage_in();
$hashref = $manager->stage_out();
$hashref = $manager->cache_cleanup();
$hashref = $manager->remote_io_file_create();
$hashref = $manager->proxy_relocate();
$hashref = $manager->proxy_update();
$scalar  = $manager->pipe_out_cmd(@arglist);
($stderr, $rc) = $manager->pipe_err_cmd(@arglist);
$status  = $manager->fork_and_exec_cmd(@arglist);
$manager->append_path($hash, $variable, $path);
$scalar = $manager->setup_softenv();

DESCRIPTION

The Globus::GRAM::JobManager module implements the base behavior for a Job Manager script interface. Scheduler-specific job manager scripts must inherit from this module in order to be used by the job manager.

Methods

$manager =
Globus::GRAM::JobManager->new($JobDescription)

Each Globus::GRAM::JobManager object is created by calling the constructor with a single argument, a Globus::GRAM::JobDescription object containing the information about the job request which the script will be modifying. Modules which subclass Globus::GRAM::JobManager MUST call the super-class's constructor, as in this code fragment:

my $proto = shift;
my $class = ref($proto) || $proto;
my $self = $class->SUPER::new(@_);

bless $self, $class;

$manager->log($string)

Log a message to the job manager log file. The message is preceded by a timestamp.

$manager->nfssync($object, $create)

Send an NFS update by touching the file (or directory) in question. If the $create is true, a file will be created. If it is false, the $object will not be created.
$manager->respond($message) Send a response to the job manager program. The response may either be a hash reference consisting of a hash of (variable, value) pairs, which will be returned to the job manager, or an already formatted string. This only needs to be directly called by a job manager implementation when the script wants to send a partial response while processing one of the scheduler interface methods (for example, to indicate that a file has been staged).

The valid keys for a response are defined in the RESPONSES section.

$manager->submit() Submit a job request to the scheduler. The default implementation returns with the Globus::GRAM::Error::UNIMPLEMENTED error. Scheduler specific subclasses should reimplement this method to submit the job to the scheduler.

A scheduler which implements this method should return a hash reference containing a scheduler-specific job identifier as the value of the hash's JOB_ID key, and optionally, the a GRAM job state as the value of the hash's JOB_STATE key if the job submission was successful; otherwise a Globus::GRAM::Error value should be returned. The job state values are defined in the Globus::GRAM::JobState module. The job parameters (as found in the job rsl) are defined in Globus::GRAM::Jobdescription object in $self->{JobDescription}.

For example:

```perl
return {JOB_STATE => Globus::GRAM::JobState::PENDING,
        JOB_ID => $job_id};
```

$manager->poll() Poll a job's status. The default implementation returns with the Globus::GRAM::Error::UNIMPLEMENTED error. Scheduler specific subclasses should reimplement this method to poll the scheduler.

A scheduler which implements this method should return a hash reference containing the JOB_STATE value. The job's ID can be accessed by calling the $self->{JobDescription}->jobid() method.

$manager->cancel() Cancel a job. The default implementation returns with the Globus::GRAM::Error::UNIMPLEMENTED error. Scheduler specific subclasses should reimplement this method to remove the job from the scheduler.

A scheduler which implements this method should return a hash reference containing the JOB_STATE value. The job's ID can be accessed by calling the $self->{JobDescription}->jobid() method.

$manager->signal() Signal a job. The default implementation returns with the Globus::GRAM::Error::UNIMPLEMENTED error. Scheduler specific subclasses should reimplement this method to remove the job from the scheduler. The JobManager module can determine the job's ID, the signal number, and the (optional) signal arguments from the Job Description by calling it's job_id(), signal(), and and signal_arg() methods, respectively.

Depending on the signal, it may be appropriate for the JobManager object to return a hash reference containing a JOB_STATE update.

$manager->make_scratchdir() Create a scratch directory for a job. The scratch directory location is based on the JobDescription's scratch_dir_base() and scratch_dir() methods.
If the scratch_dir() value is a relative path, then a directory will be created as a subdirectory of scratch_dir_base()/scratch_dir(), otherwise, it will be created as a subdirectory of scratch_dir(). This method will return a hash reference containing mapping SCRATCH_DIR to the absolute path of newly created scratch directory if successful.

$manager->remove_scratchdir() Delete a job's scratch directory. All files and subdirectories of the JobDescription's scratch_directory() will be deleted.

$manager->file_cleanup() Delete some job-related files. All files listed in the JobDescription's file_cleanup() array will be deleted.

$manager->rewrite_urls() Looks up URLs listed in the JobDescription's stdin() and executable(), and replaces them with paths to locally cached copies.

$manager->stage_in() Stage input files need for the job from remote storage. The files to be staged are defined by the array of [URL, path] pairs in the job description's file_stage_in() and file_stage_in_shared() methods. The Globus::GRAM::JobManager module provides an implementation of this functionality using the globus-url-copy and globus-gass-cache programs. Files which are staged in are not automatically removed when the job terminates.

This function returns intermediate responses using the Globus::GRAM::JobManager::response() method to let the job manager know when each individual file has been staged.

$manager->stage_out() Stage output files generated by this job to remote storage. The files to be staged are defined by the array of [URL, destination] pairs in the job description's file_stage_out() method. The Globus::GRAM::JobManager module provides an implementation of this functionality using the globus-url-copy program. Files which are staged out are not removed by this method.

$manager->cache_cleanup() Clean up cache references in the GASS which match this job's cache tag.

$manager->remote_io_file_create() Create the remote I/O file in the job dir which will contain the remote_io_url RSL attribute's value.

$manager->proxy_relocate() Relocate the delegated proxy for job execution. Job Managers need to override the default if they intend to relocate the proxy into some common file system other than the cache. The job manager program does not depend on the new location of the proxy. Job Manager modules must not remove the default proxy.

$hashref = $manager->proxy_update();

$manager->append_path($ref, $var, $path) Append $path to the value of $ref->{$var}, dealing with the case where $ref->{$var} is not yet defined.

$manager->pipe_out_cmd(@arg) Create a new process to run the first argument application with the remaining arguments (which may be empty). No shell metacharacter will be evaluated, avoiding a shell invocation. Stderr is redirected to /dev/null and stdout is being captured by the parent process, which is also the result returned. In list mode, all lines are returned, in scalar mode, only the first line is being re-
turned. The line termination character is already cut off. Use this function as more efficient backticks, if you do not need shell metacharacter evaluation.

Caution: This function deviates in two manners from regular backticks. Firstly, it chomps the line terminator from the output. Secondly, it returns only the first line in scalar context instead of a multiline concatenated string. As with regular backticks, the result may be undefined in scalar context, if no result exists.

A child error code with an exit code of 127 indicates that the application could not be run. The scalar result returned by this function is usually undef'ed in this case.

($stderr, $rc) = $manager->pipe_err_cmd(@arg)

Create a new process to run the first argument application with the remaining arguments (which may be empty). No shell metacharacter will be evaluated, avoiding a shell invocation.

This method returns a list of two items, the standard error of the program, and the exit code of the program. If the error code is 127, then the application could not be run. Standard output is discarded.

$manager->fork_and_exec_cmd(@arg)

Fork off a child to run the first argument in the list. Remaining arguments will be passed, but shell interpolation is avoided. Signals SIGINT and SIGQUIT are ignored in the child process. Stdout is appended to /dev/null, and stderr is dup2 from stdout. The parent waits for the child to finish, and returns the value for the CHILD_ERROR variable as result. Use this function as more efficient system() call, if you can do not need shell metacharacter evaluation.

Note that the inability to execute the program will result in a status code of 127.

$manager->job_dir()

Return the temporary directory to store job-related files, which have no need for file caching.

$manager->setup_softenv()

Either add a line to the specified command script file handle to load the user's default SoftEnv configuration, or create a custom SoftEnv script and add commands to the specified command script file handle to load it.

**RESPONSES**

When returning from a job interface method, or when sending an intermediate response via the `response()` method, the following hash keys are valid:

* **JOB_STATE**
  An integer job state value. These are enumerated in the Globus::GRAM::JobState module.

* **ERROR**
  An integer error code. These are enumerated in the Globus::GRAM::Error module.

* **JOB_ID**
  A string containing a job identifier, which can be used to poll, cancel, or signal a job in progress. This response should only be returned by the `submit` method.

* **SCRATCH_DIR**
  A string containing the path to a newly-created scratch directory. This response should only be returned by the `make_scratchdir` method.

* **STAGED_IN**
  A string containing the (URL, path) pair for a file which has now been staged in. This response should only be returned by the `stage_in` method.
* STAGED_IN_SHARED  A string containing the (URL, path) pair for a file which has now been staged in and symlinked from the cache. This response should only be returned by the `stage_in_shared` method.

* STAGED_OUT  A string containing the (path, URL) pair for a file which has now been staged out by the script. This response should only be returned by the `stage_out` method.
Name
Globus::GRAM::JobSignal — GRAM Protocol JobSignal Constants

DESCRIPTION
The Globus::GRAM::JobSignal module defines symbolic names for the JobSignal constants in the GRAM Protocol.

Methods

$value = \text{Globus::GRAM::CANCEL()}$
Return the value of the CANCEL constant.

$value = \text{Globus::GRAM::SUSPEND()}$
Return the value of the SUSPEND constant.

$value = \text{Globus::GRAM::RESUME()}$
Return the value of the RESUME constant.

$value = \text{Globus::GRAM::PRIORITY()}$
Return the value of the PRIORITY constant.

$value = \text{Globus::GRAM::COMMIT_REQUEST()}$
Return the value of the COMMIT_REQUEST constant.

$value = \text{Globus::GRAM::COMMIT_EXTEND()}$
Return the value of the COMMIT_EXTEND constant.

$value = \text{Globus::GRAM::STDIO_UPDATE()}$
Return the value of the STDIO_UPDATE constant.

$value = \text{Globus::GRAM::STDIO_SIZE()}$
Return the value of the STDIO_SIZE constant.

$value = \text{Globus::GRAM::STOP_MANAGER()}$
Return the value of the STOP_MANAGER constant.

$value = \text{Globus::GRAM::COMMIT_END()}$
Return the value of the COMMIT_END constant.
Name
Globus::GRAM::JobState — GRAM Protocol JobState Constants

DESCRIPTION
The Globus::GRAM::JobState module defines symbolic names for the JobState constants in the GRAM Protocol.

Methods

$value = Globus::GRAM::PENDING()$value = Globus::GRAM::ACTIVE()$value = Globus::GRAM::FAILED()$value = Globus::GRAM::DONE()$value = Globus::GRAM::SUSPENDED()$value = Globus::GRAM::UNSUBMITTED()$value = Globus::GRAM::STAGE_IN()$value = Globus::GRAM::STAGE_OUT()$value = Globus::GRAM::ALL()

Return the value of the PENDING constant.
Return the value of the ACTIVE constant.
Return the value of the FAILED constant.
Return the value of the DONE constant.
Return the value of the SUSPENDED constant.
Return the value of the UNSUBMITTED constant.
Return the value of the STAGE_IN constant.
Return the value of the STAGE_OUT constant.
Return the value of the ALL constant.
Chapter 2. RSL Specification v1.1

This is a document to specify the existing RSL v1.0 implementation and interfaces, as they are provided in the GT 5.2.2 release. This document serves as a reference, and more introductory text.

The Globus Resource Specification Language (RSL) provides a common interchange language to describe resources. The various components of the Globus Resource Management architecture manipulate RSL strings to perform their management functions in cooperation with the other components in the system. The RSL provides the skeletal syntax used to compose complicated resource descriptions, and the various resource management components introduce specific ATTRIBUTE,VALUE pairings into this common structure. Each attribute in a resource description serves as a parameter to control the behavior of one or more components in the resource management system.

1. RSL Syntax Overview

The core syntax of the RSL syntax is the relation. Relations associate an attribute name with a value, eg the relation executable=a.out provides the name of an executable in a resource request. There are two generative syntactic structures in the RSL that are used to build more complicated resource descriptions out of the basic relations: compound requests and value sequences. In addition, the RSL syntax includes a facility to both introduce and dereference string substitution variables.

The simplest form of compound request, utilized by all resource management components, is the conjunct-request. The conjunct-request expresses a conjunction of simple relations or compound requests (like a boolean AND). The most common conjunct-request in Globus RSL strings is the combination of multiple relations such as executable name, node count, executable arguments, and output files for a basic GRAM job request. Similarly, the core RSL syntax includes a disjunct-request form to represent disjunctive relations (like a boolean OR). Currently, however, no resource management component utilizes the disjunct-request form.

The last form of compound request is the multi-request. The multi-request expresses multiple parallel resources that make up a resource description. The multi-request form differs from the conjunction and disjunction in two ways: multi-requests introduce new variable scope, meaning variables defined in one clause of a multi-request are not visible to the other clauses, and multi-requests introduce a non-reducible hierarchy to the resource description. Whereas relations within a conjunct-request can be thought of as constraints on the resource being described, the subclauses of a multi-request are best thought of as individual resource descriptions that together constitute an abstract resource collection; the same attributes may be constrained in different ways in each subclause without causing a logical contradiction. An example of a contradiction would be to constrain the executable attribute to be two conflicting values within a conjunction. Currently, however, no resource management component utilizes the disjunct-request form.

The simplest form of value in the RSL syntax is the string literal. When explicitly quoted, literals can contain any character, and many common literals that don't contain special characters can appear without quotes. Values can also be variable references, in which case the variable reference is in essence replaced with the string value defined for that variable. RSL descriptions can also express string-concatenation of values, especially useful to construct long strings out of several variable references. String concatenation is supported with both an explicit concatenation operator and implicit concatenation for many idiomatic constructions involving variable references and literals.

In addition to the simple value forms given above, the RSL syntax includes the value sequence to express ordered sets of values. The value sequence syntax is used primarily for defining variables and for providing the argument list for a program.
2. RSL Tokenization Overview

Each RSL string consists of a sequence of RSL tokens, whitespace, and comments. The RSL tokens are either special syntax or regular unquoted literals, where special syntax contains one or more of the following listed special characters and unquoted literals are made of sequences of characters excluding the special characters.

The complete set of special characters that cannot appear as part of an unquoted literal is:

- + (plus)
- & (ampersand)
- | (pipe)
- ( (left paren)
- ) (right paren)
- = (equal)
- < (left angle)
- > (right angle)
- ! (exclamation)
- " (double quote)
- ' (apostrophe)
- ^ (carat)
- # (pound)
- $ (dollar)

These characters can only be used for the special syntactic forms described in the section and in the section or as within quoted literals.

Quoted literals are introduced with the " (double quote) or ' (single quote/apostrophe) and consist of all the characters up to (but not including) the next solo double or single quote, respectively. To escape a quote character within a quoted literal, the appearance of the quote character twice in a row is converted to a single instance of the character and the literal continues until the next solo quote character. For any quoted literal, there is only one possible escape sequence, eg within a literal delimited by the single quote character only the single quote character uses the escape notation and the double quote character can appear without escape.

Quoted literals can also be introduced with an alternate user delimiter notation. User delimited literals are introduced with the ^ (carat) character followed immediately by a user-provided delimiter; the literal consists of all the characters after the user's delimiter up to (but not including) the next solo instance of the delimiter. The delimiter itself may be escaped within the literal by providing two instances in a row, just as the regular quote delimiters are escaped in regular quoted literals.

RSL string comments use a notation similar to comments in the C programming language. Comments are introduced by the prefix (* . Comments continue to the first terminating suffix *) and cannot be nested. Comments are stripped from the RSL string during processing and are syntactically equivalent to whitespace.
Example 2.1. Quoted Literal Examples

Assign the value Hello. Welcome to "The Grid" to the attribute arguments, using double-quote as the delimiter and the escaping sequence.

arguments = "Hello. Welcome to ""The Grid""

Assign the value Hello. Welcome to "The Grid" to the attribute arguments using the single-quote delimiter.

arguments = 'Hello. Welcome to "The Grid'

Assign the value Hello. Welcome to "The Grid" to the attribute arguments using a user-defined quoting character !.

arguments = ^!Hello. Welcome to "The Grid"!

3. RSL Substitution Semantics

RSL strings can introduce and reference string variables. String substitution variables are defined in a special relation using the rsl_substitution attribute, and the definitions affect variable references made in the same conjunct-request (or disjunct-request), as well as references made within any multi-request nested inside one of the clauses of the conjunction (or disjunction). Each multi-request introduces a new variable scope for each subrequest, and variable definitions do not escape the closest enclosing scope.

Within any given scope, variable definitions are processed left-to-right in the resource description. Outermost scopes are processed before inner scopes, and the definitions in inner scopes augment the inherited definitions with new and/or updated variable definitions.

Variable definitions and variable references are processed in a single pass, with each definition updating the environment prior to processing the next definition. The value provided in a variable definition may include a reference to a previously-defined variable. References to variables that are not yet provided with definitions in the standard RSL variable processing order are replaced with an empty literal string.

4. RSL Attribute Summary

The RSL syntax is extensible because it defines structure without too many keywords. Each Globus resource management component introduces additional attributes to the set recognized by RSL-aware components, so it is difficult to provide a complete listing of attributes which might appear in a resource description. Resource management components are designed to utilize attributes they recognize and pass unrecognized relations through unchanged. This allows powerful compositions of different resource management functions.

The following listing summarizes the attribute names utilized by existing resource management components in the standard Globus release. Please see the individual component documentation for discussion of the attribute semantics.
Name
rsl — GRAM5 RSL Attributes

Description

arguments
The command line arguments for the executable. Use quotes, if a space is required in a single argument.

count
The number of executions of the executable. [Default: 1]

directory
Specifies the path of the directory the jobmanager will use as the default directory for the requested job. [Default: $(HOME)]

dry_run
If dryrun = yes then the jobmanager will not submit the job for execution and will return success. [Default: no]

environment
The environment variables that will be defined for the executable in addition to default set that is given to the job by the jobmanager.

executable
The name of the executable file to run on the remote machine. If the value is a GASS URL, the file is transferred to the remote gass cache before executing the job and removed after the job has terminated.

expiration
Time (in seconds) after a a job fails to receive a two-phase commit end signal before it is cleaned up. [Default: 14400]

file_clean_up
Specifies a list of files which will be removed after the job is completed.

file_stage_in
Specifies a list of ("remote URL" "local file") pairs which indicate files to be staged to the nodes which will run the job.

file_stage_in_shared
Specifies a list of ("remote URL" "local file") pairs which indicate files to be staged into the cache. A symlink from the cache to the "local file" path will be made.

file_stage_out
Specifies a list of ("local file" "remote URL") pairs which indicate files to be staged from the job to a GASS-compatible file server.

gass_cache
Specifies location to override the GASS cache location.

gram_my_job
Obsolete and ignored. [Default: collective]

host_count
Only applies to clusters of SMP computers, such as newer IBM SP systems. Defines the number of nodes ("pizza boxes") to distribute the "count" processes across.

job_type
This specifies how the jobmanager should start the job. Possible values are single (even if the count > 1, only start 1 process or thread), multiple (start count processes or threads), mpi (use the appropriate method (e.g. mpirun) to start a program compiled with a vendor-provided MPI library. Program is started with count nodes), and condor (starts condor jobs in the "condor" universe.) [Default: multiple]

library_path
Specifies a list of paths to be appended to the system-specific library path environment variables. [Default: $(GLOBUS_LOCATION)/lib]

loglevel
Override the default log level for this job. The value of this attribute consists of a combination of the strings FATAL, ERROR, WARN, INFO, DEBUG, TRACE joined by the | character
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logpattern</td>
<td>Override the default log path pattern for this job. The value of this attribute is a string (potentially containing RSL substitutions) that is evaluated to the path to write the log to. If the resulting string contains the string $(DATE) (or any other RSL substitution), it will be reevaluated at log time.</td>
</tr>
<tr>
<td>max_cpu_time</td>
<td>Explicitly set the maximum cputime for a single execution of the executable. The units is in minutes. The value will go through an atoi() conversion in order to get an integer. If the GRAM scheduler cannot set cputime, then an error will be returned.</td>
</tr>
<tr>
<td>max_memory</td>
<td>Explicitly set the maximum amount of memory for a single execution of the executable. The units is in Megabytes. The value will go through an atoi() conversion in order to get an integer. If the GRAM scheduler cannot set maxMemory, then an error will be returned.</td>
</tr>
<tr>
<td>max_time</td>
<td>The maximum walltime or cputime for a single execution of the executable. Walltime or cputime is selected by the GRAM scheduler being interfaced. The units is in minutes. The value will go through an atoi() conversion in order to get an integer.</td>
</tr>
<tr>
<td>max_wall_time</td>
<td>Explicitly set the maximum walltime for a single execution of the executable. The units is in minutes. The value will go through an atoi() conversion in order to get an integer. If the GRAM scheduler cannot set walltime, then an error will be returned.</td>
</tr>
<tr>
<td>min_memory</td>
<td>Explicitly set the minimum amount of memory for a single execution of the executable. The units is in Megabytes. The value will go through an atoi() conversion in order to get an integer. If the GRAM scheduler cannot set minMemory, then an error will be returned.</td>
</tr>
<tr>
<td>project</td>
<td>Target the job to be allocated to a project account as defined by the scheduler at the defined (remote) resource.</td>
</tr>
<tr>
<td>proxy_timeout</td>
<td>Obsolete and ignored. Now a job-manager-wide setting.</td>
</tr>
<tr>
<td>queue</td>
<td>Target the job to a queue (class) name as defined by the scheduler at the defined (remote) resource.</td>
</tr>
<tr>
<td>remote_io_url</td>
<td>Writes the given value (a URL base string) to a file, and adds the path to that file to the environment through the GLOBUS_REMOTE_IO_URL environment variable. If this is specified as part of a job restart RSL, the job manager will update the file's contents. This is intended for jobs that want to access files via GASS, but the URL of the GASS server has changed due to a GASS server restart.</td>
</tr>
<tr>
<td>restart</td>
<td>Start a new job manager, but instead of submitting a new job, start managing an existing job. The job manager will search for the job state file created by the original job manager. If it finds the file and successfully reads it, it will become the new manager of the job, sending callbacks on status and streaming stdout/err if appropriate. It will fail if it detects that the old jobmanager is still alive (via a timestamp in the state file). If stdout or stderr was being streamed over the network, new stdout and stderr attributes can be specified in the restart RSL and the jobmanager will stream to the new locations (useful when output is going to a GASS server started by the client that's listening on a dynamic port, and the client was restarted). The new job manager will return a new contact string that should be used to communicate with it. If a jobmanager is restarted multiple times, any of the previous contact strings can be given for the restart attribute.</td>
</tr>
<tr>
<td>rsl_substitution</td>
<td>Specifies a list of values which can be substituted into other rsl attributes' values through the $(SUBSTITUTION) mechanism.</td>
</tr>
</tbody>
</table>
**save_state**
Causes the jobmanager to save it's job state information to a persistent file on disk.
If the job manager exits or is suspended, the client can later start up a new job manager which can continue monitoring the job.

**savejobdescription**
Save a copy of the job description to $HOME [Default: no]

**scratch_dir**
Specifies the location to create a scratch subdirectory in. A SCRATCH_DIRECTORY RSL substitution will be filled with the name of the directory which is created.

**stderr**
The name of the remote file to store the standard error from the job. If the value is a GASS URL, the standard error from the job is transferred dynamically during the execution of the job. There are two accepted forms of this value. It can consist of a single destination: stderr = URL, or a sequence of destinations: stderr = (DESTINATION) (DESTINATION). In the latter case, the DESTINATION may itself be a URL or a sequence of an x-gass-cache URL followed by a cache tag. [Default: /dev/null]

**stderr_position**
Specifies where in the file remote standard error streaming should be restarted from. Must be 0.

**stdin**
The name of the file to be used as standard input for the executable on the remote machine. If the value is a GASS URL, the file is transferred to the remote gass cache before executing the job and removed after the job has terminated. [Default: /dev/null]

**stdout**
The name of the remote file to store the standard output from the job. If the value is a GASS URL, the standard output from the job is transferred dynamically during the execution of the job. There are two accepted forms of this value. It can consist of a single destination: stdout = URL, or a sequence of destinations: stdout = (DESTINATION) (DESTINATION). In the latter case, the DESTINATION may itself be a URL or a sequence of an x-gass-cache URL followed by a cache tag. [Default: /dev/null]

**stdout_position**
Specifies where in the file remote output streaming should be restarted from. Must be 0.

**two_phase**
Use a two-phase commit for job submission and completion. The job manager will respond to the initial job request with a WAITING_FOR_COMMIT error. It will then wait for a signal from the client before doing the actual job submission. The integer supplied is the number of seconds the job manager should wait before timing out. If the job manager times out before receiving the commit signal, or if a client issues a cancel signal, the job manager will clean up the job's files and exit, sending a callback with the job status as GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED. After the job manager sends a DONE or FAILED callback, it will wait for a commit signal from the client. If it receives one, it cleans up and exits as usual. If it times out and save_state was enabled, it will leave all of the job's files in place and exit (assuming the client is down and will attempt a job restart later). The timeoutvalue can be extended via a signal. When one of the following errors occurs, the job manager does not delete the job state file when it exits: GLOBUS_GRAM_PROTOCOL_ERROR_COMMIT_TIMED_OUT, GLOBUS_GRAM_PROTOCOL_ERROR_TTL_EXPIRED, GLOBUS_GRAM_PROTOCOL_ERROR_JM_STOPPED, GLOBUS_GRAM_PROTOCOL_ERROR_USER_PROXY_EXPIRED. In these
cases, it can not be restarted, so the job manager will not wait for the commit signal after sending the FAILED callback

**username**

Verify that the job is running as this user.

## 5. Simple RSL Examples

The following are some simple example RSL strings to illustrate idiomatic usage with existing tools and to make concrete some of the more interesting cases of tokenization, concatenation, and variable semantics. These are meant to illustrate the use of the RSL notation without much regard for the specific details of a particular resource management component.

Typical GRAM5 resource descriptions contain at least a few relations in a conjunction:
Example 2.2. GRAM5 Job Request Examples

This example shows a conjunct request containing values that are unquoted literals and ordered sequences of a mix of quoted and unquoted literals.

```plaintext
(* this is a comment *)
& (executable = a.out (* <-- that is an unquoted literal *))
  (directory = /home/nobody )
  (arguments = arg1 "arg 2")
  (count = 1)
```

This example demonstrates RSL substitutions, which can be used to make sure a string is used consistently multiple times in a resource description:

```plaintext
& (rsl_substitution = (TOPDIR "/home/nobody")
  (DATADIR $(TOPDIR)/"data")
  (EXECDIR $(TOPDIR)/bin )
  (executable = $(EXECDIR)/a.out
     (* ^-- implicit concatenation *))
  (directory = $(TOPDIR) )
  (arguments = $(DATADIR)/file1
     (* ^-- implicit concatenation *)
     $(DATADIR) # /file2
     (* ^-- explicit concatenation *)
     "$(FOO)"
     (* <-- a quoted literal *))
  (environment = (DATADIR $(DATADIR)))
  (count = 1)
```

Performing all variable substitution and removing comments yields an equivalent RSL string:

```plaintext
& (rsl_substitution = (TOPDIR "/home/nobody")
  (DATADIR "/home/nobody/data")
  (EXECDIR "/home/nobody/bin")
  (executable = "/home/nobody/bin/a.out"
     )
  (directory = "/home/nobody"
     )
  (arguments = "/home/nobody/data/file1"
      "/home/nobody/data/file2"
      "$(FOO)"
    )
  (environment = (DATADIR "/home/nobody/data"))
  (count = 1)
```

Note in the above variable-substitution example, the variable substitution definitions are not automatically made a part of the job's environment. And explicit environment attribute must be used to add environment variables for the job. Also note that the third value in the arguments clause is not a variable reference but only quoted literal that happens to contain one of the special characters.

6. RSL grammar and tokenization rules

The following is a modified BNF grammar for the Resource Specification Language. Lexical rules are provided for the implicit concatenation sequences in the form of conventional regular expressions; for the implicit-concat non-terminal rules, whitespace is not allowed between juxtaposed non-terminals. Grammar comments are provided in square brackets in a column to the right of the productions, eg [comment] to help relate productions in the grammar to the terminology used in the above discussion.
Regular expressions are provided for the terminal class `string-literal` and for RSL comments. These regular expressions make use of a common inverted character-class notation, as popularized by the various lex tools. Comments are syntactically equivalent to whitespace and can only appear where the comment prefix cannot be mistaken for the trailing part of a multi-character unquoted literal.

**RSL Grammar**

1. `specification ::= relation | '+' spec-list | '&' spec-list | '|' spec-list`  /* relation */  /* multi-request */  /* conjunct-request */  /* disjunct-request */
2. `spec-list ::= (' specification ') spec-list | '(' specification ')'`  /* Substitution variable definition */  /* Attribute binding relation */
3. `relation ::= 'rsl_substitution'=' binding-sequence`  /* Substitution variable definition */  /* Attribute binding relation */
4. `binding-sequence ::= binding binding-sequence | binding`  /* Substitution variable definition */
5. `binding ::= '(' string-literal simple-value ')'`  /* Substitution variable definition */
6. `attribute ::= string-literal`  /* attribute */
7. `op ::= '=' | '!=' | '>' | '>=' | '<' | '<='`  /* attribute */
8. `value-sequence ::= value value-sequence | value`  /* String */  /* Concatenation */
9. `value ::= '(' value-sequence ')' | simple-value`  /* String */  /* Concatenation */
10. `simple-value ::= string-literal | simple-value '#' simple-value | implicit-concat | variable-reference`  /* String */  /* Concatenation */
11. `variable-reference ::= '$(' string-literal ')'`  /* Variable Reference */
12. `implicit-concat ::= (unquoted-literal)? (implicit-concat-core)+`  /* Implicit concatenation */
13. `implicit-concat-core ::= variable-reference | (variable-reference) (unquoted-literal)`  /* Implicit concatenation */
14. `string-literal ::= quoted-literal | unquoted-literal`  /* Single-quote delimiter with escaping */  /* Double-quote delimiter with escaping */  /* User defined delimiter c with escaping */  /* Non-special characters */
15. `quoted-literal ::= '' (("")) ('''')* ''`  /* Single-quote delimiter with escaping */  /* Double-quote delimiter with escaping */  /* User defined delimiter c with escaping */  /* Non-special characters */
16. `unquoted-literal ::= (\t\v\n+&|()=<>"'^#$\)\)+`  /* Comment */
17. `comment ::= '(\* ((\*\*)|('[^'])|('[^'])\* ))\* '  /* Comment */
Chapter 3. RSL Validation File Specification

This is a document to specify the file format and semantics of the RSL validation files (RVF) used by GRAM5 to validate an Resource Specification Language job description document in various contexts. This validations ensures that the RSL attribute used in the document are understood by GRAM5, fills in any default values for missing RSL attributes, and also matches the type of the RSL value with that of the attribute to ensure it is valid.

1. RVF Syntax Overview

The core syntax of the RSL syntax is the attribute definition record. Each RSL attribute definition record can define one or more aspects of the attribute, and all but the Attribute aspect are optional.

1.1. Attribute Definition Record Syntax

Syntactically, the attribute definition record consists of a series of line-oriented attribute aspect definitions, with records separated by a blank line. Additionally, comment strings may begin a line when the first non-whitespace character in the line is #.

1.1.1. Aspect Name

The aspect name syntax is an aspect name token, which may be any character other than the : character, then its value, which may be either a Simple String or a Quoted String. When parsed, the Aspect Name is transformed into a lowercase string.

1.1.2. Simple String

The parser detects a Simple String by scanning the first non-whitespace character after the : character and seeing it is not " . A Simple String's value is parsed from the first non-whitespace character until the end of line character. Thus, in a record, the line

Attribute: executable

will parse the aspect name as Attribute and the simple string value as executable without any leading whitespace. There is no way to indicate an empty value with a simple string.

1.1.3. Quoted String

The parser detects a Quoted String by scanning the first non-whitespace character after the : character and seeing it is ". A Quoted String's value is parsed from the " character to the next " not preceded by the \ character. Thus, a Quoted String may contain an empty value, or span multiple lines.

Thus, the aspect definition

Description: "The value of the \"Quoted String\"
  It may span multiple lines"

will yield an aspect named Description with the value

The value of the "Quoted String"
1.1.4. Record Delimiter

Records are separated by blank lines which are not parts of Quoted String values. The rvf sequence

Name: record-1
Aspect_1: with an aspect

Name: some other simple value
Aspect_1: with the same aspect

will yield two records, each with two aspects: Name and Aspect_1.

1.2. Aspects used by GRAM5

The GRAM5 RVF parser supports the following set of attributes in record definitions.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>A description of the RSL attribute. This can be used to generate RSL documentation (see rsl.5), but is not otherwise used by GRAM5.</td>
</tr>
<tr>
<td>RequiredWhen</td>
<td>A list indicating when the attribute is required to be included in the RSL document. If it is not present, and the RVF does not include a default value for it, the RSL will be rejected. See Section 1.2.1, “RVF When Values” for a list of valid values for this aspect.</td>
</tr>
<tr>
<td>DefaultWhen</td>
<td>A list indicating when the attribute's default value will be provided when not in the RSL. The default value is defined by the Default aspect described below. See Section 1.2.1, “RVF When Values” for a list of valid values for this aspect.</td>
</tr>
<tr>
<td>ValidWhen</td>
<td>A list indicating when the attribute is valid in an RSL document. If the RSL attribute is included in a document which is not being used for the purpose described by this aspect's value, it will be rejected. See Section 1.2.1, “RVF When Values” for a list of valid values for this aspect.</td>
</tr>
<tr>
<td>Default</td>
<td>A default value for the RSL attribute if it's used in a context which matches the DefaultWhen aspect for this attribute's record. The value of this attribute must parsable into be a valid RSL value-sequence.</td>
</tr>
<tr>
<td>Values</td>
<td>An enumeration of values for the RSL attribute. The value must be simple single-word strings (such as yes no). RSL documents which include the attribute this record is for and do not match one of the single-word values will be rejected by the RSL validator.</td>
</tr>
<tr>
<td>Publish</td>
<td>A flag, which if it equals true, will cause the RSL attribute to be included in the documentation produced by the create_rsl_documentation.pl program. This is generally only useful for core RSL attributes included in GRAM5</td>
</tr>
</tbody>
</table>

1.2.1. RVF When Values

Several RVF aspect values are defined to include a list of contexts when the RVF record is valid, required, or should be assigned a default value. The list of contexts may include any number of the following strings, separated by whitespace:
GLOBUSGRAM_JOB_SUBMIT The aspect relates to initial job submission RSL documents.

GLOBUSGRAM_JOB_MANAGER_RESTART The aspect relates to GRAM5 restart RSL documents.

GLOBUSGRAM_JOB_MANAGER_STDIO_UPDATE The aspect relates to a STDIO_UPDATE signal.

For example, the following RVF record will be valid for all three contexts, and will provide a default value /dev/null for the GLOBUSGRAM_JOB_SUBMIT context but not the other contexts:

Attribute: stdout
Description: "The name of the remote file to store the standard output from the job. If the value is a GASS URL, the standard output from the job is transferred dynamically during the execution of the job. There are two accepted forms of this value. It can consist of a single destination: stdout = URL, or a sequence of destinations: stdout = (DESTINATION) (DESTINATION). In the latter case, the DESTINATION may itself be a URL or a sequence of an x-gass-cache URL followed by a cache tag."
Default: "/dev/null"
ValidWhen: GLOBUSGRAM_JOB_SUBMIT GLOBUSGRAM_JOB_MANAGER_RESTART GLOBUSGRAM_JOB_MANAGER_STDIO_UPDATE
DefaultWhen: GLOBUSGRAM_JOB_SUBMIT

1.3. RVF Merging

GRAM5 will look in multiple locations for RVF records, allowing for the default core set of RSL attributes to be modified on a per-LRM case, as well as on a site-specific basis. The RVF parser looks in the following locations for RVF records in sequential order:

/usr/share/globus/globus_gram_job_manager/globus-gram-job-manager.rvf

/usr/share/globus/globus_gram_job_manager/$LRM.rvf

/etc/globus/gram/job-manager.rvf

/etc/globus/gram/$LRM.rvf

All but the core RVF file are optional.

When processing multiple RVF files, GRAM5 will perform a merge with override of RVF aspects for each record based on the record's Attribute aspect. Thus, each subsequent record for a particular RSL attribute will replace the value of those aspects which are included in the new record, leaving aspects which are not mentioned in the new RVF record unchanged. To remove an aspect defined in a previous RVF record, include the aspect with an empty Quoted Value. LRM-specific and site RVF files can also define records for new RSL attributes.

For example, if the core RVF records contain the following record:

Attribute: directory
Description: "Specifies the path of the directory the jobmanager will use as the default directory for the requested job."
Default: $(HOME)
ValidWhen: GLOBUS_GRAM_JOB_SUBMIT
DefaultWhen: GLOBUS_GRAM_JOB_SUBMIT

a site-specific RVF entry could replace the default value by including an RVF record like this:

Attribute: directory
Default: /scratch/ # $(LOGNAME)

Similarly, a LRM which does not support memory-related resource limits could add this record to an LRM-specific RVF file to disable those RSL attributes for that LRM:

Attribute: min_memory
ValidWhen: ""

Attribute: max_memory
ValidWhen: ""
1.4. Grammar Definition

Grammar for RVF

[18] records ::= record record_separator records
       | record
[19] record ::= aspect_list
[20] aspect_list ::= aspect aspect_list
       | aspect
[21] aspect ::= comment* aspect_name aspect_delimiter aspect_value
       | aspect_separator
[22] aspect_name ::= whitespace* ("Attribute" | "Default" | "DefaultWhen" | "Description" | "Publish" | "RequiredWhen" | "ValidWhen")
[23] aspect_delimiter ::= ":" whitespace*
[24] aspect_value ::= """ quoted_value ""
       | unquoted_value whitespace
[25] quoted_value ::= when_value_list
       | bool_value
       | quoted_text
[26] unquoted_value ::= when_value_list
       | bool_value
       | unquoted_text
[27] when_value_list ::= when_value_list whitespace when_value
       | when_value
[28] when_value ::= "GLOBUSGRAM_JOB_SUBMIT"
       | "GLOBUSGRAM_JOB_MANAGER_RESTART"
       | "GLOBUSGRAM_JOB_MANAGER_STDIO_UPDATE"
[29] bool_value ::= "True" | "False"
[30] quoted_text ::= ((\"\n\" | \"\") | (\"\"))*

/* Quoted text consists of \ followed by a non-quote character, a non-backslash or non-quote character, or a backslash followed by a quote. In the final case, the backslash is discarded by the parser. */

/* Unquoted text value extends until the last non-whitespace character on the line */

[31] unquoted_text ::= [\^\t\n\r]*

/* Comment strings begin with # and extend until the end of the current line. */

[32] comment ::= WHITESPACE* "#" [\n] newline

[33] whitespace ::= [\t\r]
[34] record_separator ::= newline (comment | newline)*
[35] aspect_separator ::= newline
[36] newline ::= \"\n"
GRAM5 Commands
Name
globus-fork-starter — Start and monitor a fork job

Synopsis
globus-fork-starter

Description
The globus-fork-starter program is executes jobs specified on its standard input stream, recording the job state changes to a file defined in the $GLOBUS_LOCATION/etc/globus-fork.conf configuration file. It runs until its standard input stream is closed and all jobs it is managing have terminated. The log generated by this program can be used by the SEG to provide job state changes and exit codes to the GRAM service. The globus-fork-starter program is typically started by the fork GRAM module.

The globus-fork-starter program expects its input to be a series of task definitions, separated by the newline character, each representing a separate job. Each task definition contains a number of fields, separated by the colon character. The first field is always the literal string 100 indicating the message format, the second field is a unique job tag that will be distinguish the reply from this program when multiple jobs are submitted. The rest of fields contain attribute bindings. The supported attributes are:

- **directory** | Working directory of the job
- **environment** | Comma-separated list of strings defining environment variables. The form of these strings is `var=value`
- **count** | Number of processes to start
- **executable** | Full path to the executable to run
- **arguments** | Comma-separated list of command-line arguments for the job
- **stdin** | Full path to a file containing the input of the job
- **stdout** | Full path to a file to write the output of the job to
- **stderr** | Full path to a file to write the error stream of the job

Within each field, the following characters may be escaped by preceding them with the backslash character:

- backslash (`\`)
- semicolor (`;`)
- comma (`,`)
- equal (`=`)

Additionally, newline can be represented within a field by using the escape sequence `\n`.

For each job the globus-fork-starter processes, it replies by writing a single line to standard output. The replies again consist of a number of fields separated by the semicolon character.

For a successful job start, the first field of the reply is the literal 101, the second field is the tag from the input, and the third field is a comma-separated list of SEG job identifiers which consist the concatenation of a UUID and a process id. The globus-fork-starter program will write state changes to the SEG log using these job identifiers.
For a failure, the first field of the reply is the literal 102, the second field is the tag from the input, the third field is the integer representation of a GRAM error code, and the fourth field is a string explaining the error.

**ENVIRONMENT**

If the following variables affect the execution of *globus-fork-starter*

**GLOBUS_LOCATION** Path to Globus Toolkit installation. This is used to locate the `globus-fork.conf` configuration file.

**Files**

```
$GLOBUS_LOCATION/etc/   Path to fork SEG configuration file.
globus-fork.conf
```
Name
globus-gatekeeper-admin — Manage globus-gatekeeper services

Synopsis

globus-gatekeeper-admin [-h]
globus-gatekeeper-admin [-l] [-n NAME]
globus-gatekeeper-admin [-e SERVICE] [-n NAME]
globus-gatekeeper-admin [-E]
globus-gatekeeper-admin [-d SERVICE]

Description

The globus-gatekeeper-admin program manages service entries which are used by the globus-gatekeeper to execute services. Service entries are located in the /etc/grid-services directory. The globus-gatekeeper-admin can list, enable, or disable specific services, or set a service as the default. The -h command-line option shows a brief usage message.

Listing services

The -l command-line option to globus-gatekeeper-admin will cause it to list all of the services which are available to be run by the globus-gatekeeper. In the output, the service name will be followed by its status in brackets. Possible status strings are ENABLED, DISABLED, and ALIAS to NAME, where NAME is another service name.

If the -n NAME is used, then only information about the service named NAME is printed.

Enabling services

The -e SERVICE command-line option to globus-gatekeeper-admin will cause it to enable a service so that it may be run by the globus-gatekeeper.

If the -n NAME option is used as well, then the service will be enabled with the alias NAME.

Enabling a default service

The -E command-line option to globus-gatekeeper-admin will cause it to enable a service alias with the name jobmanager. The globus-gatekeeper-admin program will choose the first service it finds as the default. To enable a particular service as the default, use the -e parameter described above with the -n parameter.

Disabling services

The -d SERVICE command-line option to globus-gatekeeper-admin will cause it to disable a service so that it may not be run by the globus-gatekeeper. All aliases to a disabled service are also disabled.

Files

/etc/grid-services Default location of enabled gatekeeper service descriptions.
Name

globus-gatekeeper — Authorize and execute a grid service on behalf of a user

Synopsis

globus-gatekeeper [-help]
[ -conf PARAMETER_FILE]
[ -test] [ -d | -debug ]
[ -inetd | -f ]
[ -p PORT | -port PORT ]
[ -home PATH ] [ -l LOGFILE | -logfile LOGFILE ] [ -lf LOG_FACILITY]
[ -acctfile ACCTFILE]
[ -e LIBEXECDIR]
[ -launch_method { fork_and_exit | fork_and_wait | dont_fork } ]
[ -grid_services SERVICEDIR]
[ -globusid GLOBUSID]
[ -gridmap GRIDMAP]
[ -x509_cert_dir TRUSTED_CERT_DIR]
[ -x509_cert_file TRUSTED_CERT_FILE]
[ -x509_user_cert CERT_PATH]
[ -x509_user_key KEY_PATH]
[ -x509_user_proxy PROXY_PATH]
[ -k]
[ -globuskmap KMAP]
[ -pidfile PIDFILE]

Description

The globus-gatekeeper program is a meta-server similar to inetd or xinetd that starts other services after authenticating a TCP connection using GSSAPI and mapping the client’s credential to a local account.

The most common use for the globus-gatekeeper program is to start instances of the globus-job-manager(8) service. A single globus-gatekeeper deployment can handle multiple different service configurations by having entries in the /etc/grid-services directory.

Typically, users interact with the globus-gatekeeper program via client applications such as globusrun(1), globus-job-submit, or tools such as CoG jglobus or Condor-G.

The full set of command-line options to globus-gatekeeper consists of:

-help
   Display a help message to standard error and exit

-conf PARAMETER_FILE
   Load configuration parameters from PARAMETER_FILE. The parameters in that file are treated as additional command-line options.

-test
   Parse the configuration file and print out the POSIX user id of the globus-gatekeeper process, service home directory, service execution directory, and X.509 subject name and then exits.

-d, -debug
   Run the globus-gatekeeper process in the foreground.

-inetd
   Flag to indicate that the globus-gatekeeper process was started via inetd or a similar super-server. If this flag is set and the globus-gatekeeper was not started via inetd, a warning will be printed in the gatekeeper log.
-f Flag to indicate that the `globus-gatekeeper` process should run in the foreground. This flag has no effect when the `globus-gatekeeper` is started via inetd.

-p PORT, -port PORT Listen for connections on the TCP/IP port `PORT`. This option has no effect if the `globus-gatekeeper` is started via inetd or a similar service. If not specified and the gatekeeper is running as root, the default of 2119 is used. Otherwise, the gatekeeper defaults to an ephemeral port.

-home PATH Sets the gatekeeper deployment directory to `PATH`. This is used to interpret relative paths for accounting files, libexecdir, certificate paths, and also to set the `GLOBUS_LOCATION` environment variable in the service environment. If not specified, the gatekeeper looks for service executables in `/usr/sbin`, configuration in `/etc`, and writes logs and accounting files to `/var/log`.

-l LOGFILE, -logfile LOGFILE Write log entries to `LOGFILE`. If `LOGFILE` is equal to `logoff` or `LOGOFF`, then logging will be disabled, both to file and to syslog.

-if LOG_FACILITY Open syslog using the `LOG_FACILITY`. If not specified, `LOG_DAEMON` will be used as the default when using syslog.

-acctfile ACCTFILE Set the path to write accounting records to `ACCTFILE`. If not set, records will be written to the log file.

-e LIBEXECDIR Look for service executables in `LIBEXECDIR`. If not specified, the `sbin` subdirectory of the parameter to `-home` is used, or `/usr/sbin` if that is not set.

-launch_method fork_and_exit|fork_and_wait|dont_fork (the service runs completely independently of the gatekeeper, which exits after creating the new service process), fork_and_wait (the service is run in a separate process from the gatekeeper but the gatekeeper does not exit until the service terminates), or dont_fork, where the gatekeeper process becomes the service process via the `exec()` system call.

-grid_services SERVICEDIR Look for service descriptions in `SERVICEDIR`.

-globusid GLOBUSID Sets the `GLOBUSID` environment variable to `GLOBUSID`. This variable is used to construct the gatekeeper contact string if it cannot be parsed from the service credential.

-gridmap GRIDMAP Use the file at `GRIDMAP` to map GSSAPI names to POSIX user names.

-x509_cert_dir TRUSTED_CERT_DIR Use the directory `TRUSTED_CERT_DIR` to locate trusted CA X.509 certificates. The gatekeeper sets the environment variable `X509_CERT_DIR` to this value.

-x509_user_cert CERT_PATH Read the service X.509 certificate from `CERT_PATH`. The gatekeeper sets the `X509_USER_CERT` environment variable to this value.

-x509_user_key KEY_PATH Read the private key for the service from `KEY_PATH`. The gatekeeper sets the `X509_USER_KEY` environment variable to this value.

-x509_user_proxy PROXY_PATH Read the X.509 proxy certificate from `PROXY_PATH`. The gatekeeper sets the `X509_USER_PROXY` environment variable to this value.
-k
Use the **globus-k5** command to acquire Kerberos 5 credentials before starting the service.

-globuskmap **KMAP**
Use **KMAP** as the path to the Grid credential to kerberos initialization mapping file.

-pidfile **PIDFILE**
Write the process id of the **globus-gatekeeper** to the file named by **PIDFILE**.

**ENVIRONMENT**

If the following variables affect the execution of **globus-gatekeeper**:

**X509_CERT_DIR** Directory containing X.509 trust anchors and signing policy files.

**X509_USER_PROXY** Path to file containing an X.509 proxy.

**X509_USER_CERT** Path to file containing an X.509 user certificate.

**X509_USER_KEY** Path to file containing an X.509 user key.

**GLOBUS_LOCATION** Default path to gatekeeper service files.

**Files**

- `/etc/grid-services/SERVICENAME` Service configuration for **SERVICENAME**.
- `/etc/grid-security/grid-mapfile` Default file mapping Grid identities to POSIX identities.
- `/etc/globuskmap` Default file mapping Grid identities to Kerberos 5 principals.
- `/etc/globus-nologin` File to disable the **globus-gatekeeper** program.
- `/var/log/globus-gatekeeper.log` Default gatekeeper log.

**See also**

**globus-k5(8), globusrun(1), globus-job-manager(8)**
Name
globus-gram-audit — Load GRAM4 and GRAM5 audit records into a database

Synopsis

Description
The globus-gram-audit program loads audit records to an SQL-based database. It reads $GLOBUS_LOCATION/etc/globus-job-manager.conf by default to determine the audit directory and then uploads all files in that directory that contain valid audit records to the database configured by the globus_gram_job_manager_auditing_setup_scripts package. If the upload completes successfully, the audit files will be removed.

The full set of command-line options to globus-gram-audit consist of:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--conf CONFIG_FILE</td>
<td>Use CONFIG_FILE instead of the default from the configuration file for audit database configuration.</td>
</tr>
<tr>
<td>--check</td>
<td>Check whether the insertion of a record was successful by querying the database after inserting the records. This is used in tests.</td>
</tr>
<tr>
<td>--delete</td>
<td>Delete audit records from the database right after inserting them. This is used in tests to avoid filling the database with test records.</td>
</tr>
<tr>
<td>--audit-directory DIR</td>
<td>Look for audit records in DIR, instead of looking in the directory specified in the job manager configuration. This is used in tests to control which records are loaded to the database and then deleted.</td>
</tr>
<tr>
<td>--query SQL</td>
<td>Perform the given SQL query on the audit database. This uses the database information from the configuration file to determine how to contact the database.</td>
</tr>
<tr>
<td>--quiet</td>
<td>Reduce the amount of output for common operations.</td>
</tr>
</tbody>
</table>

FILES
The globus-gram-audit uses the following files (paths relative to $GLOBUS_LOCATION).

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>etc/globus-gram-job-manager.conf</td>
<td>GRAM5 job manager configuration. It includes the default path to the audit directory</td>
</tr>
<tr>
<td>etc/globus-gram-audit.conf</td>
<td>Audit configuration. It includes the information needed to contact the audit database.</td>
</tr>
</tbody>
</table>
Name
globus-job-cancel — Cancel a GRAM batch job

Synopsis
globus-job-cancel [-f | -force] [-q | -quiet] JOBID

Description
The globus-job-cancel program cancels the job named by JOBID. Any cached files associated with the job will remain until globus-job-clean is executed for the job.

By default, globus-job-cancel prompts the user prior to canceling the job. This behavior can be overridden by specifying the -f or -force command-line options.

Options
The full set of options to globus-job-cancel are:

- help, -usage
  Display a help message to standard error and exit.

- version
  Display the software version of the globus-job-cancel program to standard output.

- version
  Display the software version of the globus-job-cancel program including DiRT information to standard output.

- force, -f
  Do not prompt to confirm job cancel and clean-up.

- quiet, -q
  Do not print diagnostics for succesful cancel. Implies -f

ENVIRONMENT
If the following variables affect the execution of globus-job-cancel.

X509_USER_PROXY  Path to proxy credential.
X509_CERT_DIR  Path to trusted certificate directory.
Name
globus-job-clean — Cancel and clean up a GRAM batch job

Synopsis

globus-job-clean [-r RESOURCE | -resource RESOURCE]
[-f | -force] [-q | -quiet] JOBID


Description

The globus-job-clean program cancels the job named by JOBID if it is still running, and then removes any cached files on the GRAM service node related to that job. In order to do the file clean up, it submits a job which removes the cache files. By default this cleanup job is submitted to the default GRAM resource running on the same host as the job. This behavior can be controlled by specifying a resource manager contact string as the parameter to the -r or -resource option.

By default, globus-job-clean prompts the user prior to canceling the job. This behavior can be overridden by specifying the -f or -force command-line options.

Options

The full set of options to globus-job-clean are:

- help, -usage
  Display a help message to standard error and exit.

- version
  Display the software version of the globus-job-clean program to standard output.

- version
  Display the software version of the globus-job-clean program including DiRT information to standard output.

- resource RESOURCE, - r RESOURCE
  Submit the clean-up job to the resource named by RESOURCE instead of the default GRAM service on the same host as the job contact.

- force, -f
  Do not prompt to confirm job cancel and clean-up.

- quiet, -q
  Do not print diagnostics for successful clean-up. Implies -f

ENVIRONMENT

If the following variables affect the execution of globus-job-clean.

X509_USER_PROXY Path to proxy credential.
X509_CERT_DIR Path to trusted certificate directory.
Name
globus-job-get-output — Retrieve the output and error streams from a GRAM job

Synopsis
globus-job-get-output [-r RESOURCE | resource RESOURCE]


Description
The globus-job-get-output program retrieves the output and error streams of the job named by JOBID. By default, globus-job-get-output will retrieve all output and error data from the job and display them to its own output and error streams. Other behavior can be controlled by using command-line options. The data retrieval is implemented by submitting another job which simply displays the contents of the first job's output and error streams. By default this retrieval job is submitted to the default GRAM resource running on the same host as the job. This behavior can be controlled by specifying a particular resource manager contact string as the RESOURCE parameter to the -r or --resource option.

Options
The full set of options to globus-job-get-output are:

- `-help, -usage` Display a help message to standard error and exit.
- `-version` Display the software version of the globus-job-get-output program to standard output.
- `-version` Display the software version of the globus-job-get-output program including DiRT information to standard output.
- `-resource RESOURCE, -r RESOURCE` Submit the retrieval job to the resource named by RESOURCE instead of the default GRAM service on the same host as the job contact.
- `-out` Retrieve only the standard output stream of the job. The default is to retrieve both standard output and standard error.
- `-err` Retrieve only the standard error stream of the job. The default is to retrieve both standard output and standard error.
- `-tail LINES, -t LINES` Print only the last LINES count lines of output from the data streams being retrieved. By default, the entire output and error file data is retrieved. This option can not be used along with the -f or --follow options.
- `-follow LINES, -f LINES` Print the last LINES count lines of output from the data streams being retrieved and then wait until canceled, printing any subsequent job output that occurs. By default, the entire output and error file data is retrieved. This option can not be used along with the -t or --tail options.

ENVIRONMENT
If the following variables affect the execution of globus-job-get-output.
X509_USER_PROXY  Path to proxy credential.
X509_CERT_DIR     Path to trusted certificate directory.
Name
globus-job-manager — Execute and monitor jobs

Synopsis

Description
The globus-job-manager program is a service which starts and controls GRAM jobs which are executed by a local resource management system, such as LSF or Condor. The globus-job-manager program is typically started by the globus-gatekeeper program and not directly by a user. It runs until all jobs it is managing have terminated or its delegated credentials have expired.

Typically, users interact with the globus-job-manager program via client applications such as globusrun, globus-job-submit, or tools such as CoG jglobus or Condor-G.

The full set of command-line options to globus-job-manager consists of:

- **-help**
  Display a help message to standard error and exit

- **-type LRM**
  Execute jobs using the local resource manager named LRM.

- **-conf CONFIG_PATH**
  Read additional command-line arguments from the file CONFIG_PATH. If present, this must be the first command-line argument to the globus-job-manager program.

- **-globus-host-manufacturer MANUFACTURER**
  Indicate the manufacturer of the system which the jobs will execute on. This parameter sets the value of the $(GLOBUS_HOST_MANUFACTURER) RSL substitution to MANUFACTURER.

- **-globus-host-cputype CPUTYPE**
  Indicate the CPU type of the system which the jobs will execute on. This parameter sets the value of the $(GLOBUS_HOST_CPUTYPE) RSL substitution to CPUTYPE.

- **-globus-host-osname OSNAME**
  Indicate the operating system type of the system which the jobs will execute on. This parameter sets the value of the $(GLOBUS_HOST_OSNAME) RSL substitution to OSNAME.

- **-globus-host-osversion OSVERSION**
  Indicate the operating system version of the system which the jobs will execute on. This parameter sets the value of the $(GLOBUS_HOST_OSVERSION) RSL substitution to OSVERSION.

- **-globus-gatekeeper-host HOST**
  Indicate the host name of the machine which the job was submitted to. This parameter sets the value of the $(GLOBUS_GATEKEEPER_HOST) RSL substitution to HOST.
-globus-gatekeeper-port

Indicate the TCP port number of gatekeeper to which jobs are submitted to.
This parameter sets the value of the $\text{(GLOBUS_GATEKEEPER_PORT)}$ RSL substitution to \text{PORT}.

- globus-gatekeeper-subject

Indicate the X.509 identity of the gatekeeper to which jobs are submitted to.
This parameter sets the value of the $\text{(GLOBUS_GATEKEEPER_SUBJECT)}$ RSL substitution to \text{SUBJECT}.

-home

Indicate the path where the Globus Toolkit(r) is installed on the service node.
This is used by the job manager to locate its support and configuration files.

-target-globus-location

Indicate the path where the Globus Toolkit(r) is installed on the execution host. If this is omitted, the value specified as a parameter to \text{-home} is used.
This parameter sets the value of the $\text{(GLOBUS_LOCATION)}$ RSL substitution to \text{TARGET_GLOBUS_LOCATION}.

-history

Configure the job manager to write job history files to \text{HISTORY_DIRECTORY}. These files are described in the FILES section below.

-scratch-dir-base

Configure the job manager to use \text{SCRATCH_DIRECTORY} as the default scratch directory root if a relative path is specified in the job RSL's \text{scratch_dir} attribute.

-enable-syslog

Configure the job manager to write log messages via syslog. Logging is further controlled by the argument to the \text{-log-levels} parameter described below.

-log-pattern

Configure the job manager to write log messages to files named by the string \text{PATTERN}. The \text{PATTERN} string may contain job-independent RSL substitutions such as $\text{(HOME)}$, $\text{(LOGNAME)}$, etc, as well as the special RSL substitution $\text{(DATE)}$ which will be resolved at log time to the date in YYYYM-MDD form.

-stdio-log

Configure the job manager to write log messages to files in the \text{LOG_DIRECTORY} directory. This is a backward-compatible parameter, equivalent to \text{-log-pattern LOG_DIRECTORY/gram_$(DATE).log}.

-log-levels

Configure the job manager to write log messages of certain levels to syslog and/or log files. The available log levels are \text{FATAL}, \text{ERROR}, \text{WARN}, \text{INFO}, \text{DEBUG}, and \text{TRACE}. Multiple values can be combined with the | character. The default value of logging when enabled is \text{FATAL|ERROR}.

-state-file-dir

Configure the job manager to write state files to \text{STATE_DIRECTORY}. If not specified, the job manager uses the default of $\text{GLOBUS_LOCATION/tmp/gram_job_state/}$. This directory must be writable by all users and be on a file system which supports POSIX advisory file locks.

-globus-tcp-port-range

Configure the job manager to restrict its TCP/IP communication to use ports in the range described by \text{PORT_RANGE}. This value is also made available in the job environment via the \text{GLOBUS_TCP_PORT_RANGE} environment variable.

-globus-tcp-source-range

Configure the job manager to restrict its TCP/IP communication to use source ports in the range described by \text{SOURCE_RANGE}. This value is also made
available in the job environment via the `GLOBUS_TCP_SOURCE_RANGE` environment variable.

`-x509-cert-dir TRUSTED_CERTIFICATE_DIRECTORY` Configure the job manager to search `TRUSTED_CERTIFICATE_DIRECTORY` for its list of trusted CA certificates and their signing policies. This value is also made available in the job environment via the `X509_CERT_DIR` environment variable.

`-cache-location GASS_CACHE_DIRECTORY` Configure the job manager to use the path `GASS_CACHE_DIRECTORY` for its temporary GASS-cache files. This value is also made available in the job environment via the `GLOBUS_GASS_CACHE_DEFAULT` environment variable.

`-k` Configure the job manager to assume it is using Kerberos for authentication instead of X.509 certificates. This disables some certificate-specific processing in the job manager.

`-extra-envvars VAR=VAL,...` Configure the job manager to define a set of environment variables in the job environment beyond those defined in the base job environment. The format of the parameter to this argument is a comma-separated sequence of `VAR=VAL` pairs, where `VAR` is the variable name and `VAL` is the variable's value. If the value is not specified, then the value of the variable in the job manager's environment is used. This option may be present multiple times on the command-line or the job manager configuration file to append multiple environment settings.

`-seg-module SEG_MODULE` Configure the job manager to use the schedule event generator module named by `SEG_MODULE` to detect job state changes events from the local resource manager, in place of the less efficient polling operations used in GT2. To use this, one instance of the `globus-job-manager-event-generator` must be running to process events for the LRM into a generic format that the job manager can parse.

`-audit-directory AUDIT_DIRECTORY` Configure the job manager to write audit records to the directory named by `AUDIT_DIRECTORY`. This records can be loaded into a database using the `globus-gram-audit` program.

`-globus-toolkit-version TOOLKIT_VERSION` Configure the job manager to use `TOOLKIT_VERSION` as the version for audit and usage stats records.

`-service-tag SERVICE_TAG` Configure the job manager to use `SERVICE_TAG` as a unique identifier to allow multiple GRAM instances to use the same job state directories without interfering with each other's jobs. If not set, the value `untagged` will be used.

`-disable-streaming` Configure the job manager to disable file streaming. This is propagated to the LRM script interface but has no effect in GRAM5.

`-disable-usagestats` Disable sending of any usage stats data, even if `-usagestats-targets` is present in the configuration.

`-usagestats-targets TARGET` Send usage packets to a data collection service for analysis. The `TARGET` string consists of a comma-separated list of `HOST:PORT` combinations, each containing an optional list of data to send. See Usage Stats Packets¹ for more information about the tags. Special tag strings of all (which enables all

¹ http://confluence.globus.org/display/~bester/GRAM5+Usage+Stats+Packets
tags) and default may be used, or a sequence of characters for the various
tags. If this option is not present in the configuration, then the default of us-
age-stats.globus.org:4810 is used.

-`condor-arch ARCH` Set the architecture specification for condor jobs to be `ARCH` in job classi-
fied ads generated by the GRAM5 codnor LRM script. This is required for the
condor LRM but ignored for all others.

-`condor-os OS` Set the operating system specification for condor jobs to be `OS` in job classi-
fied ads generated by the GRAM5 codnor LRM script. This is required for the
condor LRM but ignored for all others.

### Environment

If the following variables affect the execution of `globus-job-manager`

**HOME** User's home directory.

**LOGNAME** User's name.

**JOBMANAGER_SYSLOG_ID** String to prepend to syslog audit messages.

**JOBMANAGER_SYSLOG_FAC** Facility to log syslog audit messages as.

**JOBMANAGER_SYSLOG_LVL** Priority level to use for syslog audit messages.

**GATEKEEPER_JM_ID** Job manager ID to be used in syslog audit records.

**GATEKEEPER_PEER** Peer information to be used in syslog audit records.

**GLOBUS_ID** Credential information to be used in syslog audit records.

**GLOBUS_JOB_MANAGER_SLEEP** Time (in seconds) to sleep when the job manager is started. [For debugging
purposes only]

**GRID_SECURITY_HTTP_BODY_FD** File descriptor of an open file which contains the initial job request and to
which the initial job reply should be sent. This file descriptor is inherited from
the `globus-gatekeeper`.

**X509_USER_PROXY** Path to the X.509 user proxy which was delegated by the client to the `globus-
gatekeeper` program to be used by the job manager.

**GRID_SECURITY_CONTEXT_FD** File descriptor containing an exported security context that the job manager
should use to reply to the client which submitted the job.

**GLOBUS_USAGE_TARGETS** Default list of usagstats services to send usage packets to.

**GLOBUS_TCP_PORT_RANGE** Default range of allowed TCP ports to listen on. The `-globus-tcp-
port-range` command-line option overrides this.

**GLOBUS_TCP_SOURCE_RANGE** Default range of allowed TCP ports to bind to. The `-globus-tcp-
source-range` command-line option overrides this.

### Files

```
$HOME/.globus/job/HOST-
NAME/LRM.TAG.red
```

Job manager delegated user credential.
globus-job-manager

$HOME/.globus/job/HOST-NAME/LRM.TAG.lock
Job manager state lock file.

$HOME/.globus/job/HOST-NAME/LRM.TAG.pid
Job manager pid file.

$HOME/.globus/job/HOST-NAME/LRM.TAG.sock
Job manager socket for inter-job manager communications.

$HOME/.globus/job/HOST-NAME/JOB_ID/
Job-specific state directory.

$HOME/.globus/job/HOST-NAME/JOB_ID/stdin
Standard input which has been staged from a remote URL.

$HOME/.globus/job/HOST-NAME/JOB_ID/stdout
Standard output which will be staged from a remote URL.

$HOME/.globus/job/HOST-NAME/JOB_ID/stderr
Standard error which will be staged from a remote URL.

$HOME/.globus/job/HOSTNAME/JOB_ID/x509_user_proxy
Job-specific delegated credential.

$GLOBUS_LOCATION/tmp/gram_job_state/job.HOSTNAME.JOB_ID
Job state file.

$GLOBUS_LOCATION/tmp/gram_job_state/job.HOSTNAME.JOB_ID.lock
Job state lock file. In most cases this will be a symlink to the job manager lock file.

$GLOBUS_LOCATION/etc/globus-job-manager.conf
Default location of the global job manager configuration file.

$GLOBUS_LOCATION/etc/grid-services/jobmanager-LRM
Default location of the LRM-specific gatekeeper configuration file.

$GLOBUS_LOCATION/etc/globus/gram/job--manager.rvf
Default location of the site-specific job manager RSL validation file.

$GLOBUS_LOCATION/etc/globus/gram/lrm.rvf
Default location of the site-specific job manager RSL validation file for the named lrm.

See Also
globusrun(1), globus-gatekeeper(8), globus-personal-gatekeeper(1), globus-gram-audit(8)
Name
globus-job-run — Execute a job using GRAM

Synopsis

SERVICE_CONTACT
[-np PROCESSES | -count PROCESSES]
[-m MAX_TIME | -maxtime MAX_TIME]
[-p PROJECT | -project PROJECT]
[-q QUEUE | -queue QUEUE]
[-d DIRECTORY | -directory DIRECTORY] [-env NAME=VALUE]...
[-x RSL_CLAUSE]
[-l | -s] EXECUTABLE [ARGUMENT...]


Description

The globus-job-run program constructs a job description from its command-line options and then submits the job to the GRAM service running at SERVICE_CONTACT. The executable and arguments to the executable are provided on the command-line after all other options. Note that the -dumprsl, -dryrun, -verify, and -file command-line options must occur before the first non-option argument, the SERVICE_CONTACT.

The globus-job-run provides similar functionality to globusrun in that it allows interactive start-up of GRAM jobs. However, unlike globusrun, it uses command-line parameters to define the job instead of RSL expressions.

Options

The full set of options to globus-job-run are:

- **-help, -usage**
  Display a help message to standard error and exit.

- **-version**
  Display the software version of the globus-job-run program to standard output.

- **-version**
  Display the software version of the globus-job-run program including DiRT information to standard output.

- **-dumprsl**
  Translate the command-line options to globus-job-run into an RSL expression that can be used with tools such as globusrun.

- **-dryrun**
  Submit the job request to the GRAM service with the dryrun option enabled. When this option is used, the GRAM service prepares to execute the job but stops before submitting the job to the LRM. This can be used to diagnose some problems such as missing files.

- **-verify**
  Submit the job request to the GRAM service with the dryrun option enabled and then without it enabled if the dryrun is successful.

- **-file ARGUMENT_FILE**
  Read additional command-line options from ARGUMENT_FILE.
globus-job-run

-\texttt{np \text{-count \hspace{1em} PROCESSES}}

\hspace{1em}Start \texttt{PROCESSES} instances of the executable as a single job.

-\texttt{m MAX\_TIME, -maxtime MAX\_TIME}

\hspace{1em}Schedule the job to run for a maximum of \texttt{MAX\_TIME} minutes.

-\texttt{p PROJECT, -project PROJECT}

\hspace{1em}Request that the job use the allocation \texttt{PROJECT} when submitting the job to the LRM.

-\texttt{q QUEUE, -queue QUEUE}

\hspace{1em}Request that the job be submitted to the LRM using the named \texttt{QUEUE}.

-\texttt{d DIRECTORY, -directory DIRECTORY}

\hspace{1em}Run the job in the directory named by \texttt{DIRECTORY}. Input and output files will be interpreted relative to this directory. This directory must exist on the file system on the LRM-managed resource. If not specified, the job will run in the home directory of the user the job is running as.

-\texttt{env NAME=VALUE}

\hspace{1em}Define an environment variable named by \texttt{NAME} with the value \texttt{VALUE} in the job environment. This option may be specified multiple times to define multiple environment variables.

-\texttt{stdin [-l | -s] \hspace{1em} STDIN\_FILE}

\hspace{1em}Use the file named by \texttt{STDIN\_FILE} as the standard input of the job. If the \texttt{-l} option is specified, then this file is interpreted to be on a file system local to the LRM. If the \texttt{-s} option is specified, then this file is interpreted to be on the file system where \texttt{globus-job-run} is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

-\texttt{stdout [-l | -s] \hspace{1em} STDOUT\_FILE}

\hspace{1em}Use the file named by \texttt{STDOUT\_FILE} as the destination for the standard output of the job. If the \texttt{-l} option is specified, then this file is interpreted to be on a file system local to the LRM. If the \texttt{-s} option is specified, then this file is interpreted to be on the file system where \texttt{globus-job-run} is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

-\texttt{stderr [-l | -s] \hspace{1em} STDERR\_FILE}

\hspace{1em}Use the file named by \texttt{STDERR\_FILE} as the destination for the standard error of the job. If the \texttt{-l} option is specified, then this file is interpreted to be on a file system local to the LRM. If the \texttt{-s} option is specified, then this file is interpreted to be on the file system where \texttt{globus-job-run} is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

-\texttt{x RSL\_CLAUSE}

\hspace{1em}Add a set of custom RSL attributes described by \texttt{RSL\_CLAUSE} to the job description. The clause must be an RSL conjunction and may contain one or more attributes. This can be used to include attributes which can not be defined by other command-line options of \texttt{globus-job-run}.

-\texttt{l}

\hspace{1em}When included outside the context of \texttt{stdin, stdout, or stderr} command-line options, \texttt{-l} option alters the interpretation of the executable path. If the \texttt{-l} option is specified, then the executable is interpreted to be on a file system local to the LRM.

-\texttt{s}

\hspace{1em}When included outside the context of \texttt{stdin, stdout, or stderr} command-line options, \texttt{-s} option alters the interpretation of the executable path. If the \texttt{-s} option is specified, then the executable is interpreted to be on the file system where \texttt{globus-job-run} is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.
ENVIRONMENT

If the following variables affect the execution of `globus-job-run`.

`X509_USER_PROXY` Path to proxy credential.
`X509_CERT_DIR` Path to trusted certificate directory.

See Also

globusrun(1), globus-job-submit(1), globus-job-clean(1), globus-job-get-output(1), globus-job-cancel(1)
Name
globus-job-status — Check the status of a GRAM5 job

Synopsis
globus-job-status JOBID

Description
The globus-job-status program checks the status of a GRAM job by sending a status request to the job manager contact for that job specified by the JOBID parameter. If successful, it will print the job status to standard output. The states supported by globus-job-status are:

- **PENDING**: The job has been submitted to the LRM but has not yet begun execution.
- **ACTIVE**: The job has begun execution.
- **FAILED**: The job has failed.
- **SUSPENDED**: The job is currently suspended by the LRM.
- **DONE**: The job has completed.
- **UNSUBMITTED**: The job has been accepted by GRAM, but not yet submitted to the LRM.
- **STAGE_IN**: The job has been accepted by GRAM and is currently staging files prior to being submitted to the LRM.
- **STAGE_OUT**: The job has completed execution and is currently staging files from the service node to other http, GASS, or GridFTP servers.

Options
The full set of options to globus-job-status are:

- **-help, -usage**: Display a help message to standard error and exit.
- **-version**: Display the software version of the globus-job-status program to standard output.
- **-versions**: Display the software version of the globus-job-status program including DiRT information to standard output.

ENVIRONMENT
If the following variables affect the execution of globus-job-status.

- **X509_USER_PROXY**: Path to proxy credential.
- **X509_CERT_DIR**: Path to trusted certificate directory.
Bugs

The `globus-job-status` program can not distinguish between the case of the job manager terminating for any reason and the job being in the DONE state.

See Also

globusrun(1)
Name
globus-job-submit — Submit a batch job using GRAM

Synopsis
SERVICE_CONTACT
[np PROCESSES| -count PROCESSES ]
[-m MAX_TIME | -maxtime MAX_TIME ]
[-p PROJECT | -project PROJECT ]
[-q QUEUE ] [-queue QUEUE ]
[-d DIRECTORY | -directory DIRECTORY ] [-env NAME=VALUE]...
[-x RSL_CLAUSE ]
[-l | -s ] EXECUTABLE [ARGUMENT...]

Description
The globus-job-submit program constructs a job description from its command-line options and then submits the job to the GRAM service running at SERVICE_CONTACT. The executable and arguments to the executable are provided on the command-line after all other options. Note that the -dumprsl, -dryrun, -verify, and -file command-line options must occur before the first non-option argument, the SERVICE_CONTACT.

The globus-job-submit provides similar functionality to globusrun in that it allows batch submission of GRAM jobs. However, unlike globusrun, it uses command-line parameters to define the job instead of RSL expressions.

To retrieve the output and error streams of the job, use the program globus-job-get-output. To reclaim resources used by the job by deleting cached files and job state, use the program globus-job-clean. To cancel a batch job submitted by globus-job-submit, use the program globus-job-cancel.

Options
The full set of options to globus-job-submit are:

-help, -usage
Display a help message to standard error and exit.

-version
Display the software version of the globus-job-submit program to standard output.

-versions
Display the software version of the globus-job-submit program including DiRT information to standard output.

-dumprsl
Translate the command-line options to globus-job-submit into an RSL expression that can be used with tools such as globusrun.

-dryrun
Submit the job request to the GRAM service with the dryrun option enabled. When this option is used, the GRAM service prepares to execute the job but stops before submitting the job to the LRM. This can be used to diagnose some problems such as missing files.

-verify
Submit the job request to the GRAM service with the dryrun option enabled and then without it enabled if the dryrun is successful.
globus-job-submit

- file ARGUMENT_FILE
  Read additional command-line options from ARGUMENT_FILE.

- np PROCESSES, -count PROCESSES
  Start PROCESSES instances of the executable as a single job.

- m MAX_TIME, -maxtime MAX_TIME
  Schedule the job to run for a maximum of MAX_TIME minutes.

- p PROJECT, -project PROJECT
  Request that the job use the allocation PROJECT when submitting the job to the LRM.

- q QUEUE, -queue QUEUE
  Request that the job be submitted to the LRM using the named QUEUE.

- d DIRECTORY, -directory DIRECTORY
  Run the job in the directory named by DIRECTORY. Input and output files will be interpreted relative to this directory. This directory must exist on the file system on the LRM-managed resource. If not specified, the job will run in the home directory of the user the job is running as.

- env NAME=VALUE
  Define an environment variable named by NAME with the value VALUE in the job environment. This option may be specified multiple times to define multiple environment variables.

- stdin [-l | -s] STDIN_FILE
  Use the file named by STDIN_FILE as the standard input of the job. If the -l option is specified, then this file is interpreted to be on a file system local to the LRM. If the -s option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

- stdout [-l | -s] STDOUT_FILE
  Use the file named by STDOUT_FILE as the destination for the standard output of the job. If the -l option is specified, then this file is interpreted to be on a file system local to the LRM. If the -s option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

- stderr [-l | -s] STDERR_FILE
  Use the file named by STDERR_FILE as the destination for the standard error of the job. If the -l option is specified, then this file is interpreted to be on a file system local to the LRM. If the -s option is specified, then this file is interpreted to be on the file system where globus-job-submit is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

- x RSL_CLAUSE
  Add a set of custom RSL attributes described by RSL_CLAUSE to the job description. The clause must be an RSL conjunction and may contain one or more attributes. This can be used to include attributes which can not be defined by other command-line options of globus-job-submit.

- l
  When included outside the context of -stdin, -stdout, or -stderr command-line options, -l option alters the interpretation of the executable path. If the -l option is specified, then the executable is interpreted to be on a file system local to the LRM.

- s
  When included outside the context of -stdin, -stdout, or -stderr command-line options, -s option alters the interpretation of the executable path. If the -s option is specified, then the executable is interpreted to be on
the file system where `globus-job-run` is being executed, and the file will be staged via GASS. If neither is specified, the local behavior is assumed.

**ENVIRONMENT**

If the following variables affect the execution of `globus-job-submit`.

- `X509_USER_PROXY`  Path to proxy credential.
- `X509_CERT_DIR`     Path to trusted certificate directory.

**See Also**

globusrun(1), globus-job-run(1), globus-job-clean(1), globus-job-get-output(1), globus-job-cancel(1)
Name
globus-personal-gatekeeper — Manage a user's personal gatekeeper daemon

Synopsis
globus-personal-gatekeeper [-killall] [-kill]

Description
The globus-personal-gatekeeper command is a utility which manages a gatekeeper and job manager service for a single user. Depending on the command-line arguments it will operate in one of several modes. In the first set of arguments indicated in the synopsis, the program provides information about the globus-personal-gatekeeper command or about instances of the globus-personal-gatekeeper that are running currently. The second set of arguments indicated in the synopsis provide control over starting a new globus-personal-gatekeeper instance. The final set of arguments provide control for terminating one or more globus-personal-gatekeeper instances.

The -start mode will create a new subdirectory of $HOME/.globus and write the configuration files needed to start a globus-gatekeeper daemon which will invoke the globus-job-manager service when new authenticated connections are made to its service port. The globus-personal-gatekeeper then exits, printing the contact string for the new gatekeeper prefixed by GRAM contact: to standard output. In addition to the arguments described above, any arguments described in globus-job-manager(8) can be appended to the command-line and will be added to the job manager configuration for the service started by the globus-gatekeeper.

The new globus-gatekeeper will continue to run in the background until killed by invoking globus-personal-gatekeeper with the -kill or -killall argument. When killed, it will kill the globus-gatekeeper and globus-job-manager processes, remove state files and configuration data, and then exit. Jobs which are running when the personal gatekeeper is killed will continue to run, but their job directory will be destroyed so they may fail in the LRM.

The full set of command-line options to globus-personal-gatekeeper consists of:

- help, -usage 
  Print command-line option summary and exit
- version 
  Print software version
- versions 
  Print software version including DiRT information
- list 
  Print a list of all currently running personal gatekeepers. These entries will be printed one per line.
- directory CONTACT 
  Print the configuration directory for the personal gatekeeper with the contact string CONTACT.
- debug 
  Print additional debugging information when starting a personal gatekeeper. This option is ignored in other modes.
- start 
  Start a new personal gatekeeper process.
- jmtype LRM 
  Use LRM as the local resource manager interface. If not provided when starting a personal gatekeeper, the job manager will use the default fork LRM.
globus-personal-gatekeeper

-auditdir  
AUDIT_DIRECTORY  
Write audit report files to AUDIT_DIRECTORY. If not provided, the job manager will not write any audit files.

-port  
PORT  
Listen for gatekeeper TCP/IP connections on the port PORT. If not provided, the gatekeeper will let the operating system choose.

-log[=DIRECTORY]  
Write job manager log files to DIRECTORY. If DIRECTORY is omitted, the default of $HOME will be used. If this option is not present, the job manager will not write any log files.

-seg  
Try to use the SEG mechanism to receive job state change information, instead of polling for these. These require either the system administrator or the user to run an instance of the globus-job-manager-event-generator program for the LRM specified by the -jmtype option.

-acctfile  
ACCOUNTING_FILE  
Write gatekeeper accounting entries to ACCOUNTING_FILE. If not provided, no accounting records are written.

Examples

This example shows the output when starting a new personal gatekeeper which will schedule jobs via the lsf LRM, with debugging enabled.

% globus-personal-gatekeeper -start -jmtype lsf

verifying setup...
done.
GRAM contact: personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User

This example shows the output when listing the current active personal gatekeepers.

% globus-personal-gatekeeper -list

personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe User

This example shows the output when querying the configuration directory for the above personal gatekeeper.

% globus-personal-gatekeeper -directory "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe"

/home/juser/.globus/.personal-gatekeeper.personal-grid.example.org.1337

% globus-personal-gatekeeper -kill "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe"

killing gatekeeper: "personal-grid.example.org:57846:/DC=org/DC=example/CN=Joe"

See Also

globusrun(1), globus-job-manager(8), globus-gatekeeper(8)
Name
globus-rvf-check — Edit a GRAM5 RSL validation file

Synopsis
globus-rvf-check [-h] [-help]
globus-rvf-check [-d] {FILENAME...}

Description
The globus-rvf-check command is a utility which checks the syntax of a RSL validation file, and prints out parse errors when encountered. It can also parse the RVF file contents and then dump file's contents to stdout, after canonicalizing values and quoting. The exit code of globus-rvf-check is 0 if all files specified on the command line exist and have no parse errors.

The full set of command-line options to globus-rvf-check consists of:

- **-h, --help** Print command-line option summary and exit
- **-d** Dump the RVF contents to stdout. In the output, Each file which is parsed will be prefixed by an RVF comment which contains the input filename. If not specified, globus-rvf-check just prints a diagnostic message to standard output indicating whether the file could be parsed.
**Name**
globus-rvf-edit — Edit a GRAM5 RSL validation file

**Synopsis**

globus-rvf-edit [-h]

globus-rvf-edit [--s | [-l LRM] | [-f PATH]]

**Description**

The `globus-rvf-edit` command is a utility which opens the default editor on a specified RSL validation file, and then, when editing completes, runs the `globus-rvf-check` command to verify that the RVF file syntax is correct. If a parse error occurs, the user will be given an option to rerun the editor or discard the modifications.

The full set of command-line options to `globus-rvf-edit` consists of:

- **-h**  
  Print command-line option summary and exit

- **-s**  
  Edit of the site-specific RVF file, which provides override values applicable to all LRMs installed on the system.

- **-l LRM**  
  Edit the site-specific LRM overrides for the LRM named by the `LRM` parameter to the option.

- **-f PATH**  
  Edit the RVF file located at `PATH`
Name
globus-scheduler-event-generator-admin — Manage SEG modules

Synopsis
globus-scheduler-event-generator-admin [-h]
globus-scheduler-event-generator-admin [-l]
globus-scheduler-event-generator-admin [-e MODULE]
globus-scheduler-event-generator-admin [-d MODULE]

Description
The globus-scheduler-event-generator-admin program manages SEG modules which are used by the globus-scheduler-event-generator to monitor a local resource manager or batch system for events. The globus-scheduler-event-generator-admin can list, enable, or disable specific SEG modules. The -h command-line option shows a brief usage message.

Listing SEG Modules
The -l command-line option to globus-scheduler-event-generator-admin will cause it to list all of the SEG modules which are available to be run by the globus-scheduler-event-generator. In the output, the service name will be followed by its status in brackets. Possible status strings are ENABLED and DISABLED.

Enabling SEG Modules
The -e MODULE command-line option to globus-scheduler-event-generator-admin will cause it to enable the module so that the init script for the globus-scheduler-event-generator will run it.

Disabling SEG Modules
The -d MODULE command-line option to globus-scheduler-event-generator-admin will cause it to disable the module so that it will not be started by the globus-scheduler-event-generator init script.

Files
/etc/globus/scheduler-event-generator Default location of enabled SEG modules.

See Also
globus-scheduler-event-generator(8)
Name
globus-scheduler-event-generator — Process LRM events into a common format for use with GRAM

Synopsis
globus-scheduler-event-generator -s LRM
  [-t TIMESTAMP] [-d DIRECTORY]
  [-b] [-p PIDFILE]

Description
The globus-scheduler-event-generator program processes information from a local resource manager to generate LRM-independent events which GRAM can use to track job state changes. Typically, the globus-scheduler-event-generator is started at system boot time for all LRM adapters which have been installed. The only required parameter to globus-scheduler-event-generator is -s LRM, which indicates what LRM-specific module to load. A list of available modules can be found by using the globus-scheduler-event-generator-admin -l command.

Other options control how the globus-scheduler-event-generator program runs and where its output goes. These options are:

- `-t TIMESTAMP` Start processing events which start at TIMESTAMP in seconds since the UNIX epoch. If not present, the globus-scheduler-event-generator will process events from the time it was started, and not look for historical events.

- `-d DIRECTORY` Write the event log to files in DIRECTORY, instead of printing them to standard output. Within DIRECTORY, logs will be named by the time when they were created in YYYYMMDD format.

- `-b` Run the globus-scheduler-event-generator program in the background.

- `-p PIDFILE` Write the process-id of globus-scheduler-event-generator to PIDFILE.

Files
/var/lib/globus/globus-seg-LRM/YYYYMMDD  LRM-independent event log generated by globus-scheduler-event-generator

See Also
globus-scheduler-event-generator-admin(8), globus-job-manager(8)
Name
globusrun — Execute and manage jobs via GRAM

Synopsis


        globusrun { -p | -parse }
            { -f RSL_FILENAME | -file RSL_FILENAME | RSL_SPECIFICATION }

        globusrun [-n] [-no-interrupt]
            { -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
            { -a | -authenticate-only }

        globusrun [-n] [-no-interrupt]
            { -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
            { -j | -jobmanager-version }

        globusrun [-n] [-no-interrupt] { -k | -kill } {JOB_ID}


        globusrun { -status } {JOB_ID}

            { -r RESOURCE_CONTACT | -resource RESOURCE_CONTACT }
            { -f RSL_FILENAME | -file RSL_FILENAME | RSL_SPECIFICATION }

Description

The globusrun program for submits and manages jobs run on a local or remote job host. The jobs are controlled by
the globus-job-manager program which interfaces with a local resource manager that schedules and executes the
job.

The globusrun program can be run in a number of different modes chosen by command-line options.

When -help, -usage, -version, or -versions command-line options are used, globusrun will print out di-
agnostic information and then exit.

When the -p or -parse command-line option is present, globusrun will verify the syntax of the RSL specifi-
cation and then terminate. If the syntax is valid, globusrun will print out the string "RSL Parsed Successful-
ly..." and exit with a zero exit code; otherwise, it will print an error message and terminate with a non-zero exit
code.

When the -a or -authenticate-only command-line option is present, globusrun will verify that the service
named by RESOURCE_CONTACT exists and the client's credentials are granted permission to access that service.
If authentication is successful, globusrun will display the string "GRAM Authentication test successful"
and exit with a zero exit code; otherwise it will print an explanation of the problem and will with a non-zero
exit code.

When the -j or -jobmanager-version command-line option is present, globusrun will attempt to determine
the software version that the service named by RESOURCE_CONTACT is running. If successful, it will display both
the Toolkit version and the Job Manager package version and exit with a zero exit code; otherwise, it will print an
explanation of the problem and exit with a non-zero exit code.
When the `-k` or `-kill` command-line option is present, `globusrun` will attempt to terminate the job named by `JOB_ID`. If successful, `globusrun` will exit with zero; otherwise it will display an explanation of the problem and exit with a non-zero exit code.

When the `-y` or `-refresh-proxy` command-line option is present, `globusrun` will attempt to delegate a new X.509 proxy to the job manager which is managing the job named by `JOB_ID`. If successful, `globusrun` will exit with zero; otherwise it will display an explanation of the problem and exit with a non-zero exit code. This behavior can be modified by the `-full-proxy` or `-D` command-line options to enable full proxy delegation. The default is limited proxy delegation.

When the `-status` command-line option is present, `globusrun` will attempt to determine the current state of the job. If successful, the state will be printed to standard output and `globusrun` will exit with a zero exit code; otherwise, a description of the error will be displayed and it will exit with a non-zero exit code.

Otherwise, `globusrun` will submit the job to a GRAM service. By default, `globusrun` waits until the job has terminated or failed before exiting, displaying information about job state changes and at exit time, the job exit code if it is provided by the GRAM service.

The `globusrun` program can also function as a GASS file server to allow the `globus-job-manager` program to stage files to and from the machine on which `globusrun` is executed to the GRAM service node. This behavior is controlled by the `-s`, `-o`, and `-w` command-line options.

Jobs submitted by `globusrun` can be monitored interactively or detached. To have `globusrun` detach from the GRAM service after submitting the job, use the `-b` or `-F` command-line options.

## Options

The full set of options to `globusrun` consist of:

- `-help` Display a help message to standard error and exit.
- `-usage` Display a one-line usage summary to standard error and exit.
- `-version` Display the software version of `globusrun` to standard error and exit.
- `-versions` Display the software version of all modules used by `globusrun` (including DiRT information) to standard error and then exit.
- `-p`, `-parse` Do a parse check on the job specification and print diagnostics. If a parse error occurs, `globusrun` exits with a non-zero exit code.
- `-f RSL_FILENAME`, `-file RSL_FILENAME` Read job specification from the file named by `RSL_FILENAME`.
- `-n`, `-no-interrupt` Disable handling of the SIGINT signal, so that the interrupt character (typically `Control+C`) causes `globusrun` to terminate without canceling the job.
- `-r RESOURCE_CONTACT`, `-resource RESOURCE_CONTACT` Submit the request to the resource specified by `RESOURCE_CONTACT`. A resource may be specified in the following ways:
  - `HOST`
  - `HOST:PORT`
  - `HOST:PORT/SERVICE`
globusrun

- HOST/SERVICE
- HOST/SERVICE
- HOST:SUBJECT
- HOST:PORT:SUBJECT
- HOST/SERVICE:SUBJECT
- HOST/SERVICE:SUBJECT
- HOST:PORT/SERVICE:SUBJECT

If any of PORT, SERVICE, or SUBJECT is omitted, the defaults of 2811, jobmanager, and host@HOST are used respectively.

-j, -jobmanager-version Print the software version being run by the service running at RESOURCE_CONTACT.

-k JOB_ID, -kill JOB_ID Kill the job named by JOB_ID.

-D, -full-proxy Delegate a full impersonation proxy to the service. By default, a limited proxy is delegated when needed.

-y, -refresh-proxy Delegate a new proxy to the service processing JOB_ID.

-status Display the current status of the job named by JOB_ID.

-q, -quiet Do not display job state change or exit code information.

-o, -output-enable Start a GASS server within the globusrun application that allows access to its standard output and standard error streams only. Also, augment the RSL_SPECIFICATION with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add stdout and stderr clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the -q were also specified.

-s, -server Start a GASS server within the globusrun application that allows access to its standard output and standard error streams for writing and any file local to the globusrun invocation for reading. Also, augment the RSL_SPECIFICATION with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add stdout and stderr clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the -q were also specified.

-w, -write-allow Start a GASS server within the globusrun application that allows access to its standard output and standard error streams for writing and any file local to the globusrun invocation for reading or writing. Also, augment the RSL_SPECIFICATION with a definition of the GLOBUSRUN_GASS_URL RSL substitution and add stdout and stderr clauses which redirect the output and error streams of the job to the output and error streams of the interactive globusrun command. If this is specified, then globusrun acts as though the -q were also specified.
-b, --batch

Terminate after submitting the job to the GRAM service. The `globusrun` program will exit after the job hits any of the following states: PENDING, ACTIVE, FAILED, or DONE. The GASS-related options can be used to stage input files, but standard output, standard error, and file staging after the job completes will not be processed.

-F, --fast-batch

Terminate after submitting the job to the GRAM service. The `globusrun` program will exit after it receives a reply from the service. The `JOB_ID` will be displayed to standard output before terminating so that the job can be checked with the `--status` command-line option or modified by the `--refresh-proxy` or `--kill` command-line options.

-d, --dryrun

Submit the job with the `dryrun` attribute set to true. When this is done, the job manager will prepare to start the job but start short of submitting it to the service. This can be used to detect problems with the `RSL_SPECIFICATION`.

Environment

If the following variables affect the execution of `globusrun`

- `X509_USER_PROXY` Path to proxy credential.
- `X509_CERT_DIR` Path to trusted certificate directory.

Bugs

The `globusrun` program assumes any failure to contact the job means the job has terminated. In fact, this may be due to the `globus-job-manager` program exiting after all jobs it is managing have reached the DONE or FAILED states. In order to reliably detect job termination, the `two_phase` RSL attribute should be used.

See Also

globus-job-submit(1), globus-job-run(1), globus-job-clean(1), globus-job-get-output(1), globus-job-cancel(1)
Chapter 4. Configuring GRAM5

GRAM5 is designed to be usable by default without any manual configuration. However, there are many ways to customize a GRAM5 installation to better interact with site policies, filesystem layouts, LRM interactions, logging, and auditing. In addition to GRAM5-specific configuration, see Configuring GSI for information about configuring GSI security.

1. Gatekeeper Configuration

The globus-gatekeeper has many configuration options related to network configuration, security, logging, service path, and nice level. This configuration is located in:

Table 4.1. Gatekeeper Configuration Path

<table>
<thead>
<tr>
<th>Installation Type</th>
<th>Configuration Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>/etc/sysconfig/globus-gatekeeper</td>
</tr>
<tr>
<td>Debian Package</td>
<td>/etc/default/globus-gatekeeper</td>
</tr>
<tr>
<td>Source Installer</td>
<td>PREFIX/etc/globus-gatekeeper.conf</td>
</tr>
</tbody>
</table>

The following configuration variables are available in the globus-gatekeeper configuration file:

GLOBUS_GATEKEEPER_PORT Gatekeeper Service Port. If not set, the globus-gatekeeper uses the default of 2119.

GLOBUS_LOCATION Globus Installation Path. If not set, the globus-gatekeeper uses the paths defined at package compilation time.

GLOBUS_GATEKEEPER_LOG Gatekeeper Log Filename. If not set, the globus-gatekeeper logs to syslog using the GRAM-gatekeeper log identification prefix. The default configuration value is /var/log/globus-gatekeeper.log

GLOBUS_GATEKEEPER_GRID_SERVICES Path to grid service definitions. If not set, the globus-gatekeeper uses the default of /etc/grid-services.

GLOBUS_GATEKEEPER_GRIDMAP Path to grid-mapfile for authorization. If not set, the globus-gatekeeper uses the default of /etc/grid-security/grid-mapfile.

GLOBUS_GATEKEEPER_CERT_DIR Path to a trusted certificate root directory. If not set, the globus-gatekeeper uses the default of /etc/grid-security/certificates.

GLOBUS_GATEKEEPER_CERT_FILE Path to the gatekeeper's certificate. If not set, the globus-gatekeeper uses the default of /etc/grid-security/hostcert.pem.

GLOBUS_GATEKEEPER_KEY_FILE Path to the gatekeeper's private key. If not set, the globus-gatekeeper uses the default of /etc/grid-security/hostkey.pem.
Configuring GRAM5

GLOBUS_GATEKEEPER_KERBEROS_ENABLED Flag indicating whether or not the globus-gatekeeper will use a kerberos GSSAPI implementation instead of the GSI GSSAPI implementation (untested).

GLOBUS_GATEKEEPER_KMAP Path to the KMAP authentication module. (untested).

GLOBUS_GATEKEEPER_PIDFILE Path to a file where the globus-gatekeeper's process ID is written. If not set, globus-gatekeeper uses /var/run/globus-gatekeeper.pid

GLOBUS_GATEKEEPER_NICE_LEVEL Process nice level for globus-gatekeeper and globus-job-manager processes. If not set, the default system process nice level is used.

After modifying the configuration file, restart the globus-gatekeeper using the methods described in Section 2, “Starting and Stopping GRAM5 services”.

2. Scheduler Event Generator Configuration

The globus-scheduler-event-generator has several configuration options related to filesystem paths. This configuration is located in:

<table>
<thead>
<tr>
<th>Installation Type</th>
<th>Configuration Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>/etc/sysconfig/globus-scheduler-event-generator</td>
</tr>
<tr>
<td>Debian Package</td>
<td>/etc/default/globus-scheduler-event-generator</td>
</tr>
<tr>
<td>Source Installer</td>
<td>$PREFIX/etc/globus-scheduler-event-generator.conf</td>
</tr>
</tbody>
</table>

The following configuration variables are available in the globus-scheduler-event-generator configuration file:

GLOBUS_SEG_PIDFMT Scheduler Event Generator PID file path format. Modify this to be the location where the globus-scheduler-event-generator writes its process IDs (one per configured LRM). The format is a printf format string with one %s to be replaced by the LRM name. By default, globus-scheduler-event-generator uses /var/run/globus-scheduler-event-generator-%s.pid.

GLOBUS_SEG_LOGFMT Scheduler Event Generator Log path format. Modify this to be the location where globus-scheduler-event-generator writes its event logs. The format is a printf format string with one %s to be replaced by the LRM name. By default, globus-scheduler-event-generator uses /var/lib/globus/globus-seg-%s. If you modify this value, you'll need to also update the LRM configuration file to look for the log file in the new location.

GLOBUS_SEG_NICE_LEVEL Process nice level for globus-scheduler-event-generator processes. If not set, the default system process nice level is used.

After modifying the configuration file, restart the globus-scheduler-event-generator using the methods described in Section 2, “Starting and Stopping GRAM5 services”.

71
3. Job Manager Configuration

The `globus-job-manager` process is started by the `globus-gatekeeper` and uses the configuration defined in the service entry for the resource name. By default, these service entries use a common configuration file for most job manager features. This configuration is located in:

### Table 4.3. Job Manager Configuration Path

<table>
<thead>
<tr>
<th>Installation Type</th>
<th>Configuration Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>/etc/globus/globus-gram-job-manager.conf</td>
</tr>
<tr>
<td>Debian Package</td>
<td>/etc/globus/globus-gram-job-manager.conf</td>
</tr>
<tr>
<td>Source Installer</td>
<td>PREFIX/etc/globus-gram-job-manager.conf</td>
</tr>
</tbody>
</table>

This configuration file is used to construct the command-line options for the `globus-job-manager` program. Thus, all of the options described in (8) may be used.

#### 3.1. Job Manager Logging

From an administrator's perspective, the most important job manager configuration options are likely the ones related to logging and auditing. The default GRAM5 configuration puts logs in `/var/log/globus/gram_USERNAME.log`, with logging enabled at the FATAL and ERROR levels. To enable more fine-grained logging, add the option `-log-levels LEVELS` to `/etc/globus/globus-gram-job-manager.conf`. The value for `LEVELS` is a set of log levels joined by the `|` character. The available log levels are:

### Table 4.4. GRAM5 Log Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Meaning</th>
<th>Default Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATAL</td>
<td>Problems which cause the job manager to terminate prematurely.</td>
<td>Enabled</td>
</tr>
<tr>
<td>ERROR</td>
<td>Problems which cause a job or operation to fail.</td>
<td>Enabled</td>
</tr>
<tr>
<td>WARN</td>
<td>Problems which cause minor problems with job execution or monitoring.</td>
<td>Disabled</td>
</tr>
<tr>
<td>INFO</td>
<td>Major events in the lifetime of the job manager and its jobs.</td>
<td>Disabled</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Minor events in the lifetime of jobs.</td>
<td>Disabled</td>
</tr>
<tr>
<td>TRACE</td>
<td>Job processing details.</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

In RPM or Debian package installs, these logs will be configured to be rotated via `logrotate`. See `/etc/logrotate.d/globus-job-manager` for details on the default log rotation configuration.

#### 3.2. Firewall Configuration

There are also a few configuration options related to the TCP ports the Job Manager users. This port configuration is useful when dealing with firewalls that restrict incoming or outgoing ports. To restrict incoming ports (those
that the Job Manager listens on), add the command-line option \texttt{-globus-tcp-port-range} to the Job Manager configuration file like this:

\texttt{-globus-tcp-port-range MIN-PORT, MAX-PORT}

Where \textit{MIN-PORT} is the minimum TCP port number the Job Manager will listen on and \textit{MAX-PORT} is the maximum TCP port number the Job Manager will listen on.

Similarly, to restrict the outgoing port numbers that the job manager connects form, use the command-line option \texttt{-globus-tcp-source-range}, like this:

\texttt{-globus-tcp-source-range MIN-PORT, MAX-PORT}

Where \textit{MIN-PORT} is the minimum outgoing TCP port number the Job Manager will use and \textit{MAX-PORT} is the maximum TCP outgoing port number the Job Manager will use.

For more information about Globus and firewalls, see Section 4, “Firewall configuration”.

4. LRM Adapter Configuration

Each LRM adapter has its own configuration file which can help customize the adapter to the site configuration. Some LRMs use non-standard programs to launch parallel or MPI jobs, and some might want to provide queue or project validation to make it easier to translate job failures into problems that can be described by GRAM5. All of the LRM adapter configuration files consist of simple \texttt{variable="value"} pairs, with a leading \# starting a comment until end-of-line.

Generally, the GRAM5 LRM configuration files are located in the globus configuration directory, with each configuration file named by the LRM name (fork, condor, pbs, sge). The following table contains the paths to these configurations:

<table>
<thead>
<tr>
<th>Installation Type</th>
<th>Configuration Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>/etc/globus/globus-LRM.conf</td>
</tr>
<tr>
<td>Debian Package</td>
<td>/etc/globus/globus-LRM.conf</td>
</tr>
<tr>
<td>Source Installer</td>
<td>PREFIX/etc/globus/globus-LRM.conf</td>
</tr>
</tbody>
</table>

4.1. Fork

The \texttt{globus-fork.conf} configuration file can define the following configuration parameters:

- \texttt{log_path} Path to the \texttt{globus-fork.log} file used by the \texttt{globus-fork-starter} and fork SEG module.
- \texttt{mpirun} Path to \texttt{mpirun} for parallel jobs which use MPI. By default, these are not configured. The LRM adapter will use \texttt{mpirun} if both are defined.
- \texttt{softenv_dir} Path to an installation of \texttt{softenv}\footnote{http://www.mcs.anl.gov/hs/software/systems/softenv/softenv-intro.html}, which is used on some systems to manage application environment variables.

4.2. Condor

The \texttt{globus-condor.conf} configuration file can define the following configuration parameters:
Configuring GRAM5

condor_os Custom value for the OpSys requirement for condor jobs. If not specified, the system-wide default will be used.

condor_arch Custom value for the OpSys requirement for condor jobs. If not specified, the system-wide default will be used.

condor_submit, condor_rm Path to the condor commands that the LRM adapter uses. These are usually determined when the LRM adapter is compiled if the commands are in the PATH.

condor_config Value of the CONDOR_CONFIG environment variable, which might be needed to use condor in some cases.

check_vanilla_files Enable checking if executable, standard input, and directory are valid paths for vanilla universe jobs. This can detect some types of errors before submitting jobs to condor, but only if the filesystems between the condor submit host and condor execution hosts are equivalent. In other cases, this may cause unnecessary job failures.

condor_mpi_script Path to a script to launch MPI jobs on condor

4.3. PBS

The globus-pbs.conf configuration file can define the following configuration parameters:

log_path Path to PBS server_logs directory. The PBS SEG module parses these logs to generate LRM events.

pbs_default Name of the PBS server node, if not the same as the GRAM service node.

mpiexec, mpirun Path to mpiexec and mpirun for parallel jobs which use MPI. By default these are not configured. The LRM adapter will use mpiexec over mpirun if both are defined.

qsub, qstat, qdel Path to the LRM-specific command to submit, check, and delete PBS jobs. These are usually determined when the LRM adapter is compiled if they are in the PATH.

cluster If this value is set to yes, then the LRM adapter will attempt to use a remote shell command to launch multiple instances of the executable on different nodes, as defined by the file named by the PBS_NODEFILE environment variable.

remote_shell Remote shell command to launch processes on different nodes when cluster is set to yes.

cpu_per_node Number of instances of the executable to launch per allocated node.

softenv_dir Path to an installation of softenv\(^2\) which is used on some systems to manage application environment variables.

4.4. SGE

The globus-sge.conf configuration file can define the following configuration parameters:

sge_root Root location of the GridEngine installation. If this is set to undefined, then the LRM adapter will try to determine it from the globus-job-manager environment, or if not there, the contents of the file named by the sge_config configuration parameter.

Configuring GRAM5

sge_cell  Name of the GridEngine cell to interact with. If this is set to undefined, then the LRM adapter will try to determine it from the globus-job-manager environment, or if not there, the contents of the file named by the sge_config configuration parameter.

sge_config  Path to a file which defines the SGE_ROOT and the SGE_CELL environment variables.

log_path  Path to GridEngine reporting file. This value is used by the SGE SEG module. If this is used, GridEngine must be configured to write a reporting file and not load reporting data into an ARCo database.

qsub, qstat, qdel, qconf  Path to the LRM-specific command to submit, check, and delete GridEngine jobs. These are usually determined when the LRM adapter is compiled if they are in the PATH.

sun_mprun, mpirun  Path to mprun and mpirun for parallel jobs which use MPI. By default these are not configured. The LRM adapter will use mprun over mpirun if both are defined.

default_pe  Default parallel environment to submit parallel jobs to. If this is not set, then clients must use the parallel_environment RSL attribute to choose one.

validate_pes  If this value is set to yes, then the LRM adapter will verify that the parallel_environment RSL attribute value matches one of the parallel environments supported by this GridEngine service.

available_pes  If this value is defined, use it as a list of parallel environments supported by this GridEngine deployment for validation when validate_pes is set to yes. If validation is being done but this value is not set, then the LRM adapter will query the GridEngine service to determine available parallel environments at startup.

default_queue  Default queue to use if the job description does not name one.

validate_queues  If this value is set to yes, then the LRM adapter will verify that the queue RSL attribute value matches one of the queues supported by this GridEngine service.

available_queues  If this value is defined, use it as a list of queues supported by this GridEngine deployment for validation when validate_queues is set to yes. If validation is being done but this value is not set, then the LRM adapter will query the GridEngine service to determine available queues at startup.

4.4.1. Enabling reporting for the GridEngine Scheduler Event Generator

In order to use the Scheduler Event Generator with GridEngine, the job reporting feature must be enabled, and ARCo database storage must not be enabled. To enable this, use the command `qconf -mconf` and modify the reporting_params parameter so that the options reporting and joblog are set to true.

5. Auditing

The globus-gram-audit configuration defines information about the database to load the GRAM5 audit records into. This configuration is located in:

Table 4.6. GRAM Audit Configuration Path

<table>
<thead>
<tr>
<th>Installation Type</th>
<th>Configuration Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>/etc/globus/gram-audit.conf</td>
</tr>
</tbody>
</table>
This configuration file contains the following attributes. Each attribute is defined by a \texttt{ATTRIBUTE:VALUE} pair.

\textbf{Table 4.7. Audit Configuration Attributes}

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRIVER</td>
<td>The name of the Perl 5 DBI driver for the database to be used. The supported drivers for this program are SQLite, \texttt{Pg} (for PostgreSQL), and mysql.</td>
<td>SQLite</td>
</tr>
<tr>
<td>DATABASE</td>
<td>The DBI data source specification to contact the audit database.</td>
<td>dbname=/var/gram_audit_database/gram_audit.db</td>
</tr>
<tr>
<td>USERNAME</td>
<td>Username to authenticate as to the database</td>
<td></td>
</tr>
<tr>
<td>PASSWORD</td>
<td>Password to use to authenticate with the database</td>
<td></td>
</tr>
<tr>
<td>AUDITVERSION</td>
<td>Version of the audit database table schemas to use. May be 1 or 1TG for this version of the software.</td>
<td>1</td>
</tr>
</tbody>
</table>

\section{6. RSL Attributes}

GRAM5 uses the RSL language to encode job descriptions. The attributes supported by gram are defined in RSL Validation Files. These definitions contain information about when the different RSL attributes are valid and what their default values might be if not present. GRAM5 will look in \texttt{/etc/globus/gram/job-manager.rvf} and \texttt{/etc/globus/gram/LRM.rvf} for site-specific changes to the RSL validation file.
Chapter 5. Job description

Jobs are described in GRAM5's job description language. For detailed schema information see the GRAM5 RSL documentation. For more information and examples please check the Globusrun section in the GRAM5 Users's guide.
Chapter 6. Semantics and syntax of protocols

1. GRAM5 Protocol

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 as used by GRAM5. This is compatible with with the GRAM Protocol parsers in GRAM2 with extensions.

1.1. 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-Line:

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

1.2. 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\" )"

In GRAM5, protocol extensions are supported in the status update messages. These extensions are implemented as extra attribute names after all of the attributes defined in the messages below. Older GRAM protocol parsers will ignore those extensions that occur after the attributes in the messages defined below. In GRAM5, the following extensions are used:

- **exit-code**
  Job exit code. Sent in job state callbacks and in job status replies when the job completes.

- **gt3-failure-type**
  Failure detail type for staging errors. Sent in job state callbacks and in job status replies when a job fails.

- **gt3-failure-message**
  Failure detail message for more context for errors. Sent in job state callbacks and in job status replies when a job fails.

- **gt3-failure-source**
  Failure detail message for the source of a failed file transfer. Sent in job state callbacks and in job status replies when a job fails.

- **gt3-failure-destination**
  Failure detail message for the destination of a failed file transfer. Sent in job state callbacks and in job status replies when a job fails.

- **version**
  Job manager package version. Sent in all messages from the job manager.

- **toolkit-version**
  Toolkit release that the job manager is running. Sent in all messages from the job manager.

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.

### 1.3. Message Types

#### 1.3.1. 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-LRM".

- **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".

1.3.2. 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-LRM`.
- **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 user-name 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.
- **callback-url**: 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.
- **rsl**: A quoted string containing the RSL description of the job request.

1.3.3. 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.

1.3.4. 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 mask callback-contact"

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.

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.
1.3.5. 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 callback-contact"
```

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 @ref globus_gram_protocol_job_state_updates "below".

1.3.6. 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.

1.3.7. 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 protocol library documentation. 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
"signal"

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 its parameters.

1.3.8. 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.
Semantics and syntax of protocols

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.

1.3.9. 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 proceed 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.

1.3.10. 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.

1.4. Job State Model

As the GRAM service processes a job, the job undergoes a series of state transitions. These states and their meanings follow:

Table 6.1. GRAM Job States

<table>
<thead>
<tr>
<th>State</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED</td>
<td>Initial job state</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_IN</td>
<td>Job staging in progress</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING</td>
<td>Job submitted to LRM, awaiting execution</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE</td>
<td>Job executing</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_SUSPENDED</td>
<td>Job made progress executing but is now suspended</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_STAGE_OUT</td>
<td>Job staging in progress after job completed</td>
</tr>
<tr>
<td>GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE</td>
<td>Job completed successfully</td>
</tr>
<tr>
<td>State</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>GLOBUSGRAM_PROTOCOL_JOB_STATE_FAILED</td>
<td>Job was canceled or failed</td>
</tr>
</tbody>
</table>

Figure 6.1. GRAM State Transitions
# Appendix A. Errors

Table A.1. GRAM5 Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Reason</th>
<th>Possible Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>one of the RSL parameters is not supported</td>
<td>Check RSL documentation</td>
</tr>
<tr>
<td>2</td>
<td>the RSL length is greater than the maximum allowed</td>
<td>Use RSL substitutions to reduce length of RSL strings</td>
</tr>
<tr>
<td>3</td>
<td>an I/O operation failed</td>
<td>Enable trace logging and report to <a href="mailto:gram-dev@globus.org">gram-dev@globus.org</a></td>
</tr>
<tr>
<td>4</td>
<td>jobmanager unable to set default to the directory requested</td>
<td>Check that RSL directory attribute refers to a directory that exists on the target system.</td>
</tr>
<tr>
<td>5</td>
<td>the executable does not exist</td>
<td>Check that the RSL executable attribute refers to an executable that exists on the target system.</td>
</tr>
<tr>
<td>6</td>
<td>of an unused INSUFFICIENT_FUNDS</td>
<td>Unimplemented feature.</td>
</tr>
<tr>
<td>7</td>
<td>authentication with the remote server failed</td>
<td>Check that the contact string contains the proper X.509 DN.</td>
</tr>
<tr>
<td>8</td>
<td>the user cancelled the job</td>
<td>Don't cancel jobs you want to complete.</td>
</tr>
<tr>
<td>9</td>
<td>the system cancelled the job</td>
<td>Check RSL requirements such as maximum time and memory are valid for the job.</td>
</tr>
<tr>
<td>10</td>
<td>data transfer to the server failed</td>
<td>Check gatekeeper and/or job manager logs to see why the process failed.</td>
</tr>
<tr>
<td>11</td>
<td>the stdin file does not exist</td>
<td>Check that the RSL stdin attribute refers to a file that exists on the target system or has a valid ftp, gsiftp, http, or https URL.</td>
</tr>
<tr>
<td>12</td>
<td>the connection to the server failed (check host and port)</td>
<td>Check that the service is running on the expected TCP/IP port. Check that no firewall prevents contacting that TCP/IP port. Check $GLOBUS_LOCATION/var/globus-gatekeeper.log for runtime configuration errors.</td>
</tr>
<tr>
<td>13</td>
<td>the provided RSL 'maxtime' value is not an integer</td>
<td>Check that the RSL maxtime value evaluates to an integer.</td>
</tr>
<tr>
<td>14</td>
<td>the provided RSL 'count' value is not an integer</td>
<td>Check that the RSL count value evaluates to an integer.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>15</td>
<td>the job manager received an invalid RSL</td>
<td>Check that the RSL string can be parsed by using <code>globusrun -p RSL</code>.</td>
</tr>
<tr>
<td>16</td>
<td>the job manager failed in allowing others to make contact</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>17</td>
<td>the job failed when the job manager attempted to run it</td>
<td>Verify that the LRM is configured properly.</td>
</tr>
<tr>
<td>18</td>
<td>an invalid paradyn was specified</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>19</td>
<td>the provided RSL 'jobtype' value is invalid</td>
<td>The RSL jobtype attribute is not indicated as supported by the LRM. Valid jobtype values are <code>single</code>, <code>multiple</code>, <code>mpi</code>, and <code>condor</code>.</td>
</tr>
<tr>
<td>20</td>
<td>the provided RSL 'myjob' value is invalid</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>21</td>
<td>the job manager failed to locate an internal script argument file</td>
<td>Check that <code>$GLOBUS_LOCATION/libexec/globus-job-manager-script.pl</code> exists and is executable. Check that the LRM-specific perl module is located in <code>$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/</code> directory and is valid. The command <code>perl -I $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/LRM.pm</code> can be used to check if there are any syntax errors in the script.</td>
</tr>
<tr>
<td>22</td>
<td>the job manager failed to create an internal script argument file</td>
<td>Check that your home directory is writable and not full.</td>
</tr>
<tr>
<td>23</td>
<td>the job manager detected an invalid job state</td>
<td>Check job manager logs.</td>
</tr>
<tr>
<td>24</td>
<td>the job manager detected an invalid script response</td>
<td>Check job manager logs. This is likely a bug in the LRM script.</td>
</tr>
<tr>
<td>25</td>
<td>the job manager detected an invalid script status</td>
<td>Check job manager logs. This is likely a bug in the LRM script.</td>
</tr>
<tr>
<td>26</td>
<td>the provided RSL 'jobtype' value is not supported by this job manager</td>
<td>Check that the RSL jobtype attribute is implemented by the LRM script. Note that some job types require configuration</td>
</tr>
<tr>
<td>27</td>
<td>unused ERROR_UNIMPLEMENTED</td>
<td>LRM does not support some feature included in the job request.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>28</td>
<td>the job manager failed to create an internal script submission file</td>
<td>Check that the user's home filesystem is not full. Check job manager log</td>
</tr>
<tr>
<td>29</td>
<td>the job manager cannot find the user proxy</td>
<td>Check that client is delegating a proxy when authenticating with the gatekeeper. Check that the user's home filesystem and the /tmp file system are not full.</td>
</tr>
<tr>
<td>30</td>
<td>the job manager failed to open the user proxy</td>
<td>Check that the user's home filesystem and the /tmp file system are not full.</td>
</tr>
<tr>
<td>31</td>
<td>the job manager failed to cancel the job as requested</td>
<td>Check that the user's home filesystem and the /tmp file system are not full.</td>
</tr>
<tr>
<td>32</td>
<td>system memory allocation failed</td>
<td>Check job manager log for details.</td>
</tr>
<tr>
<td>33</td>
<td>the interprocess job communication initialization failed</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>34</td>
<td>the interprocess job communication setup failed</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>35</td>
<td>the provided RSL 'host count' value is invalid</td>
<td>Check that the RSL host_count attribute evaluates to an integer.</td>
</tr>
<tr>
<td>36</td>
<td>one of the provided RSL parameters is unsupported</td>
<td>Check job manager log for details about invalid parameter.</td>
</tr>
<tr>
<td>37</td>
<td>the provided RSL 'queue' parameter is invalid</td>
<td>Check that the RSL queue attribute evaluates to a string that corresponds to an LRM-specific queue name.</td>
</tr>
<tr>
<td>38</td>
<td>the provided RSL 'project' parameter is invalid</td>
<td>Check that the RSL project attribute evaluates to a string that corresponds to an LRM-specific project name.</td>
</tr>
<tr>
<td>39</td>
<td>the provided RSL string includes variables that could not be identified</td>
<td>Check that all RSL substitutions are defined before being used in the job description.</td>
</tr>
<tr>
<td>40</td>
<td>the provided RSL 'environment' parameter is invalid</td>
<td>Check that the RSL environment attribute contains a sequence of VARIABLE VALUE pairs.</td>
</tr>
<tr>
<td>41</td>
<td>the provided RSL 'dryrun' parameter is invalid</td>
<td>Remove the RSL dryrun attribute from the job description.</td>
</tr>
<tr>
<td>42</td>
<td>the provided RSL is invalid (an empty string)</td>
<td>Include a non-empty RSL string in your job submission request.</td>
</tr>
<tr>
<td>43</td>
<td>the job manager failed to stage the executable</td>
<td>Check that the file service hosting the executable is reachable from the GRAM5 service node.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>44</td>
<td>the job manager failed to stage the stdin file</td>
<td>Check that the file service hosting the standard input file is reachable from the GRAM5 service node. Check that the standard input file exists on the file service node. Check that there is sufficient disk space in the user's home directory on the service node to store the standard input file.</td>
</tr>
<tr>
<td>45</td>
<td>the requested job manager type is invalid</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>46</td>
<td>the provided RSL 'arguments' parameter is invalid</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>47</td>
<td>the gatekeeper failed to run the job manager</td>
<td>Check the gatekeeper or job manager logs for more information.</td>
</tr>
<tr>
<td>48</td>
<td>the provided RSL could not be properly parsed</td>
<td>Check that the RSL string can be parsed by using <code>globusrun -p RSL</code>.</td>
</tr>
<tr>
<td>49</td>
<td>there is a version mismatch between GRAM components</td>
<td>Ask system administrator to upgrade GRAM service to GRAM2 or GRAM5</td>
</tr>
<tr>
<td>50</td>
<td>the provided RSL 'arguments' parameter is invalid</td>
<td>Check that the RSL arguments attribute evaluates to a sequence of strings.</td>
</tr>
<tr>
<td>51</td>
<td>the provided RSL 'count' parameter is invalid</td>
<td>Check that the RSL count attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>52</td>
<td>the provided RSL 'directory' parameter is invalid</td>
<td>Check that the RSL directory attribute evaluates to a string.</td>
</tr>
<tr>
<td>53</td>
<td>the provided RSL 'dryrun' parameter is invalid</td>
<td>Check that the RSL dryrun attribute evaluates to either yes or no.</td>
</tr>
<tr>
<td>54</td>
<td>the provided RSL 'environment' parameter is invalid</td>
<td>Check that the RSL environment attribute evaluates to a sequence of VARIABLE, VALUE pairs.</td>
</tr>
<tr>
<td>55</td>
<td>the provided RSL 'executable' parameter is invalid</td>
<td>Check that the RSL executable attribute evaluates to a string value.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>56</td>
<td>the provided RSL 'host_count' parameter is invalid</td>
<td>Check that the RSL host_count attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>57</td>
<td>the provided RSL 'jobtype' parameter is invalid</td>
<td>Check that the RSL jobtype attribute evaluates to one of single, multiple, mpi, or condor</td>
</tr>
<tr>
<td>58</td>
<td>the provided RSL 'maxtime' parameter is invalid</td>
<td>Check that the RSL maxtime attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>59</td>
<td>the provided RSL 'myjob' parameter is invalid</td>
<td>OBSOLETE IN GRAM5.</td>
</tr>
<tr>
<td>60</td>
<td>the provided RSL 'paradyn' parameter is invalid</td>
<td>OBSOLETE IN GRAM2.</td>
</tr>
<tr>
<td>61</td>
<td>the provided RSL 'project' parameter is invalid</td>
<td>Check that the RSL project attribute evaluates to a string value.</td>
</tr>
<tr>
<td>62</td>
<td>the provided RSL 'queue' parameter is invalid</td>
<td>Check that the RSL queue attribute evaluates to a string value.</td>
</tr>
<tr>
<td>63</td>
<td>the provided RSL 'stderr' parameter is invalid</td>
<td>Check that the RSL stderr attribute evaluates to a string value or a sequence of DESTINATION URLs with optional CACHE_TAG string parameters.</td>
</tr>
<tr>
<td>64</td>
<td>the provided RSL 'stdin' parameter is invalid</td>
<td>Check that the RSL stdin attribute evaluates to a string value.</td>
</tr>
<tr>
<td>65</td>
<td>the provided RSL 'stdout' parameter is invalid</td>
<td>Check that the RSL stdout attribute evaluates to a string value or a sequence of DESTINATION URLs with optional CACHE_TAG string parameters.</td>
</tr>
<tr>
<td>66</td>
<td>the job manager failed to locate an internal script</td>
<td>Check job manager log for more details.</td>
</tr>
<tr>
<td>67</td>
<td>the job manager failed on the system call pipe()</td>
<td>OBSOLETE IN GRAM5.</td>
</tr>
<tr>
<td>68</td>
<td>the job manager failed on the system call fcntl()</td>
<td>OBSOLETE IN GRAM2.</td>
</tr>
<tr>
<td>69</td>
<td>the job manager failed to create the temporary stdout filename</td>
<td>OBSOLETE IN GRAM5.</td>
</tr>
<tr>
<td>70</td>
<td>the job manager failed to create the temporary stderr filename</td>
<td>OBSOLETE IN GRAM5.</td>
</tr>
<tr>
<td>71</td>
<td>the job manager failed on the system call fork()</td>
<td>OBSOLETE IN GRAM2.</td>
</tr>
<tr>
<td>72</td>
<td>the executable file permissions do not allow execution</td>
<td>Check that the RSL executable attribute refers to an executable program or script.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>73</td>
<td>the job manager failed to open stdout</td>
<td>Check that the RSL <code>stdout</code> attribute refers to one or more valid destination files or URLs.</td>
</tr>
<tr>
<td>74</td>
<td>the job manager failed to open stderr</td>
<td>Check that the RSL <code>stderr</code> attribute refers to one or more valid destination files or URLs.</td>
</tr>
<tr>
<td>75</td>
<td>the cache file could not be opened in order to relocate the user proxy</td>
<td>Check that the user's home directory is writable and not full on the GRAM5 service node.</td>
</tr>
<tr>
<td>76</td>
<td>cannot access cache files in <code>~/.globus/.gass_cache</code>, check permissions, quota, and disk space</td>
<td>Check that the user's home directory is writable and not full on the GRAM5 service node.</td>
</tr>
<tr>
<td>77</td>
<td>the job manager failed to insert the contact in the client contact list</td>
<td>Check job manager log</td>
</tr>
<tr>
<td>78</td>
<td>the contact was not found in the job manager's client contact list</td>
<td>Don't attempt to unregister callback contacts that are not registered</td>
</tr>
<tr>
<td>79</td>
<td>connecting to the job manager failed. Possible reasons: job terminated, invalid job contact, network problems, ...</td>
<td>Check that the job manager process is running. Check that the job manager credential has not expired. Check that the job manager contact refers to the correct TCP/IP host and port. Check that the job manager contact is not blocked by a firewall.</td>
</tr>
<tr>
<td>80</td>
<td>the syntax of the job contact is invalid</td>
<td>Check the syntax of job contact string.</td>
</tr>
<tr>
<td>81</td>
<td>the executable parameter in the RSL is undefined</td>
<td>Include the RSL <code>executable</code> in all job requests.</td>
</tr>
<tr>
<td>82</td>
<td>the job manager service is misconfigured. <code>condor arch</code> undefined</td>
<td>Add the <code>-condor-arch</code> to the command-line or configuration file for a job manager configured to use the <code>condor LRM</code>.</td>
</tr>
<tr>
<td>83</td>
<td>the job manager service is misconfigured. <code>condor os</code> undefined</td>
<td>Add the <code>-condor-os</code> to the command-line or configuration file for a job manager configured to use the <code>condor LRM</code>.</td>
</tr>
<tr>
<td>84</td>
<td>the provided RSL <code>min_memory</code> parameter is invalid</td>
<td>Check that the RSL <code>min_memory</code> attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>85</td>
<td>the provided RSL <code>max_memory</code> parameter is invalid</td>
<td>Check that the RSL <code>max_memory</code> attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>86</td>
<td>the RSL <code>min_memory</code> value is not zero or greater</td>
<td>Check that the RSL <code>min_memory</code> attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>87</td>
<td>the RSL 'max_memory' value is not zero or greater</td>
<td>Check that the RSL max_memory attribute evaluates to a positive integer value.</td>
</tr>
<tr>
<td>88</td>
<td>the creation of a HTTP message failed</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>89</td>
<td>parsing incoming HTTP message failed</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>90</td>
<td>the packing of information into a HTTP message failed</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>91</td>
<td>an incoming HTTP message did not contain the expected information</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>92</td>
<td>the job manager does not support the service that the client requested</td>
<td>Check that the client is talking to the correct service</td>
</tr>
<tr>
<td>93</td>
<td>the gatekeeper failed to find the requested service</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>94</td>
<td>the jobmanager does not accept any new requests (shutting down)</td>
<td>Execute queries before the job has been cleaned up.</td>
</tr>
<tr>
<td>95</td>
<td>the client failed to close the listener associated with the callback URL</td>
<td>Call globus_gram_client_callback_disallow() with a valid the callback contact.</td>
</tr>
<tr>
<td>96</td>
<td>the gatekeeper contact cannot be parsed</td>
<td>Check the syntax of the gatekeeper contact string you are attempting to contact.</td>
</tr>
<tr>
<td>97</td>
<td>the job manager could not find the 'poe' command</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>98</td>
<td>the job manager could not find the 'mpirun' command</td>
<td>Configure the LRM script with mpirun in your path.</td>
</tr>
<tr>
<td>99</td>
<td>the provided RSL 'start_time' parameter is invalid</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>100</td>
<td>the provided RSL 'reservation_handle' parameter is invalid</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>101</td>
<td>the provided RSL 'max_wall_time' parameter is invalid</td>
<td>Check that the RSL max_wall_time attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>102</td>
<td>the RSL 'max_wall_time' value is not zero or greater</td>
<td>Check that the RSL max_wall_time attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>103</td>
<td>the provided RSL 'max_cpu_time' parameter is invalid</td>
<td>Check that the RSL max_cpu_time attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>104</td>
<td>the RSL 'max_cpu_time' value is not zero or greater</td>
<td>Check that the RSL max_cpu_time attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>105</td>
<td>the job manager is misconfigured, a scheduler script is missing</td>
<td>Check that the administrator has configured the LRM by running its setup script.</td>
</tr>
<tr>
<td>106</td>
<td>the job manager is misconfigured, a scheduler script has invalid permissions</td>
<td>Check that the administrator has installed the GLLOBUS_LOCATION/libexec/globus-job-manager-script.pl script. Check that the file system containing that script allows file execution.</td>
</tr>
<tr>
<td>107</td>
<td>the job manager failed to signal the job</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>108</td>
<td>the job manager did not recognize/support the signal type</td>
<td>Check that your signal operation is using the correct signal constant.</td>
</tr>
<tr>
<td>109</td>
<td>the job manager failed to get the job id from the local scheduler</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>110</td>
<td>the job manager is waiting for a commit signal</td>
<td>Send a two-phase commit signal to the job manager to acknowledge receiving the job contact from the job manager.</td>
</tr>
<tr>
<td>111</td>
<td>the job manager timed out while waiting for a commit signal</td>
<td>Send a two-phase commit signal to the job manager to acknowledge receiving the job contact from the job manager. Increase the two-phase commit time out for your job. Check that the job manager contact TCP/IP port is reachable from your client.</td>
</tr>
<tr>
<td>112</td>
<td>the provided RSL 'save_state' parameter is invalid</td>
<td>Check that the RSL save_state attribute is set to yes or no.</td>
</tr>
<tr>
<td>113</td>
<td>the provided RSL 'restart' parameter is invalid</td>
<td>Check that the RSL restart attribute evaluates to a string containing a job contact string.</td>
</tr>
<tr>
<td>114</td>
<td>the provided RSL 'two_phase' parameter is invalid</td>
<td>Check that the RSL two_phase attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>115</td>
<td>the RSL 'two_phase' value is not zero or greater</td>
<td>Check that the RSL two_phase attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>116</td>
<td>the provided RSL 'stdout_position' parameter is invalid</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>117</td>
<td>the RSL 'stdout_position' value is not zero or greater</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>118</td>
<td>the provided RSL 'stderr_position' parameter is invalid</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>119</td>
<td>the RSL 'stderr_position' value is not zero or greater</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>120</td>
<td>the job manager restart attempt failed</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>121</td>
<td>the job state file doesn't exist</td>
<td>Check that the job contact you are trying to restart matches one that the job manager returned to you.</td>
</tr>
<tr>
<td>122</td>
<td>could not read the job state file</td>
<td>Check that the state file directory is not full.</td>
</tr>
<tr>
<td>123</td>
<td>could not write the job state file</td>
<td>Check that the state file directory is not full.</td>
</tr>
<tr>
<td>124</td>
<td>old job manager is still alive</td>
<td>Contact the returned job manager contact to manage the job you are trying to restart.</td>
</tr>
<tr>
<td>125</td>
<td>job manager state file TTL expired</td>
<td>OBSOLETE in GRAM2</td>
</tr>
<tr>
<td>126</td>
<td>it is unknown if the job was submitted</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>127</td>
<td>the provided RSL 'remote_io_url' parameter is invalid</td>
<td>Check that the RSL remote_io_url attribute evaluates to a string value.</td>
</tr>
<tr>
<td>128</td>
<td>could not write the remote io url file</td>
<td>Check that the user's home file system on the job manager service node is writable and not full.</td>
</tr>
<tr>
<td>129</td>
<td>the standard output/error size is different</td>
<td>Send a stdio update signal to redirect the job manager output to a new URL</td>
</tr>
<tr>
<td>130</td>
<td>the job manager was sent a stop signal (job is still running)</td>
<td>Submit a restart request to monitor the job.</td>
</tr>
<tr>
<td>131</td>
<td>the user proxy expired (job is still running)</td>
<td>Generate a new proxy and then submit a restart request to monitor the job.</td>
</tr>
<tr>
<td>132</td>
<td>the job was not submitted by original jobmanager</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>133</td>
<td>the job manager is not waiting for that commit signal</td>
<td>Do not send a commit signal to a job that is not waiting for a commit signal.</td>
</tr>
<tr>
<td>134</td>
<td>the provided RSL scheduler specific parameter is invalid</td>
<td>Check the LRM-specific documentation to determine what values are legal for the RSL extensions implemented by the LRM.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>135</td>
<td>the job manager could not stage in a file</td>
<td>Check that the file service hosting the file to stage is reachable from the GRAM5 service node. Check that the file to stage exists on the file service node. Check that there is sufficient disk space in the user's home directory on the service node to store the file to stage.</td>
</tr>
<tr>
<td>136</td>
<td>the scratch directory could not be created</td>
<td>Check that the directory named by the RSL <code>scratch_dir</code> attribute exists and is writable. Check that the directory named by the RSL <code>scratch_dir</code> attribute is not full.</td>
</tr>
<tr>
<td>137</td>
<td>the provided 'gass_cache' parameter is invalid</td>
<td>Check that the RSL <code>gass_cache</code> attribute evaluates to a string.</td>
</tr>
<tr>
<td>138</td>
<td>the RSL contains attributes which are not valid for job submission</td>
<td>Do not use restart- or signal-only RSL attributes when submitting a job.</td>
</tr>
<tr>
<td>139</td>
<td>the RSL contains attributes which are not valid for stdio update</td>
<td>Do not use submit- or restart-only RSL attributes when sending a stdio update signal to a job.</td>
</tr>
<tr>
<td>140</td>
<td>the RSL contains attributes which are not valid for job restart</td>
<td>Do not use submit- or signal-only RSL attributes when restarting a job.</td>
</tr>
<tr>
<td>141</td>
<td>the provided RSL 'file_stage_in' parameter is invalid</td>
<td>Check that the RSL <code>file_stage_in</code> attribute evaluates to a sequence of SOURCE DESTINATION pairs.</td>
</tr>
<tr>
<td>142</td>
<td>the provided RSL 'file_stage_in_shared' parameter is invalid</td>
<td>Check that the RSL <code>file_stage_in_shared</code> attribute evaluates to a sequence of SOURCE DESTINATION pairs.</td>
</tr>
<tr>
<td>143</td>
<td>the provided RSL 'file_stage_out' parameter is invalid</td>
<td>Check that the RSL <code>file_stage_out</code> attribute evaluates to a sequence of SOURCE DESTINATION pairs.</td>
</tr>
<tr>
<td>144</td>
<td>the provided RSL 'gass_cache' parameter is invalid</td>
<td>Check that the RSL <code>gass_cache</code> attribute evaluates to a string.</td>
</tr>
<tr>
<td>145</td>
<td>the provided RSL 'file_cleanup' parameter is invalid</td>
<td>Check that the RSL <code>file_clean_up</code> attribute evaluates to a sequence of strings.</td>
</tr>
<tr>
<td>146</td>
<td>the provided RSL 'scratch_dir' parameter is invalid</td>
<td>Check that the RSL <code>scratch_dir</code> attribute evaluates to a string.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>147</td>
<td>the provided scheduler-specific RSL parameter is invalid</td>
<td>Check the LRM-specific documentation to determine what values are legal for the RSL extensions implemented by the LRM.</td>
</tr>
<tr>
<td>148</td>
<td>a required RSL attribute was not defined in the RSL spec</td>
<td>Check that the RSL executable attribute is present in your job request RSL. Check that the RSL restart attributes is present in your restart RSL.</td>
</tr>
<tr>
<td>149</td>
<td>the gass_cache attribute points to an invalid cache directory</td>
<td>Check that the RSL gass_cache attributes evaluates to a directory that exists or can be created. Check that the user's home file system is writable and not full.</td>
</tr>
<tr>
<td>150</td>
<td>the provided RSL 'save_state' parameter has an invalid value</td>
<td>Check that the RSL save_state attribute has a value of yes or no.</td>
</tr>
<tr>
<td>151</td>
<td>the job manager could not open the RSL attribute validation file</td>
<td>Check that <code>$GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf</code> is present and readable on the job manager service node. Check that <code>$GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf</code> is readable on the job manager service node if present.</td>
</tr>
<tr>
<td>152</td>
<td>the job manager could not read the RSL attribute validation file</td>
<td>Check that <code>$GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf</code> is valid. Check that <code>$GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf</code> is valid if present.</td>
</tr>
<tr>
<td>153</td>
<td>the provided RSL 'proxy_timeout' is invalid</td>
<td>Check that RSL proxy_timeout attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>154</td>
<td>the RSL 'proxy_timeout' value is not greater than zero</td>
<td>Check that RSL proxy_timeout attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>155</td>
<td>the job manager could not stage out a file</td>
<td>Check that the source file being staged exists on the job manager service node. Check that the directory of the destination file being staged exists on the file service</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td></td>
<td>node. Check that the directory of the destination file being staged is writable by the user. Check that the destination file service is reachable by the job manager service node.</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>the job contact string does not match any which the job manager is handling</td>
<td>Check that the job contact string matches one returned from a job request.</td>
</tr>
<tr>
<td>157</td>
<td>proxy delegation failed</td>
<td>Check that the job manager service node trusts the signer of your credential. Check that you trust the signer of the job manager service node's credential.</td>
</tr>
<tr>
<td>158</td>
<td>the job manager could not lock the state lock file</td>
<td>Check that the file system holding the job state directory supports POSIX advisory locking. Check that the job state directory is writable by the user on the service node. Check that the job state directory is not full.</td>
</tr>
<tr>
<td>159</td>
<td>an invalid globus_io_clientattr_t was used.</td>
<td>Check that you have initialized the globus_io_clientattr_t attribute prior to using it with the GRAM client API.</td>
</tr>
<tr>
<td>160</td>
<td>an null parameter was passed to the gram library</td>
<td>Check that you are passing legal values to all GRAM API calls.</td>
</tr>
<tr>
<td>161</td>
<td>the job manager is still streaming output</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>162</td>
<td>the authorization system denied the request</td>
<td>Check with your GRAM system administrator to allow a particular certificate to be authorized.</td>
</tr>
<tr>
<td>163</td>
<td>the authorization system reported a failure</td>
<td>Check with your system administrator to verify that the authorization system is configured properly.</td>
</tr>
<tr>
<td>164</td>
<td>the authorization system denied the request - invalid job id</td>
<td>Check with your system administrator to verify that the authorization system is configured properly. Use a credential which is authorized to interact with a particular GRAM job.</td>
</tr>
<tr>
<td>165</td>
<td>the authorization system denied the request - not authorized to run the specified executable</td>
<td>Check with your system administrator to verify that the authorization system is configured properly. Use a credential which is authorized to interact with a particular GRAM job.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>166</td>
<td>the provided RSL 'user_name' parameter is invalid.</td>
<td>Check that the RSL <code>user_name</code> attribute evaluates to a string.</td>
</tr>
<tr>
<td>167</td>
<td>the job is not running in the account named by the 'user_name' parameter.</td>
<td>Ask with the GRAM system administrator to add an authorization entry to allow your credential to run jobs as the specified user account.</td>
</tr>
</tbody>
</table>
Glossary

R

Resource Specification Language (RSL)  Term used to describe a GRAM job for GT2 and GT3. (Note: This is not the same as RLS - the Replica Location Service)