GT 5.2.4 GRAM5: User’s Guide
Introduction

GRAM services provide secure, remote job submission to different local resource managers in a Grid environment. This document describes the features of the GRAM service and an overview of tools to use the service.
# Table of Contents

1. GRAM5 Overview ................................................................. 1
   - 1. GRAM Client Tools ....................................................... 1
   - 2. Portals and Science Gateways that use GRAM ........................... 2
2. Using GRAM5 ........................................................................ 3
   - 1. Before Getting Started .................................................... 3
   - 2. Basic Client Interface ....................................................... 4
   - 3. Advanced GRAM Client with the globusrun tool ................. 8
3. Troubleshooting ..................................................................... 15
   - 1. GRAM Client Troubleshooting ........................................... 15
4. Command-line Client Reference Pages ......................................... 19
   - globusrun ............................................................................. 20
   - globus-job-cancel ............................................................... 24
   - globus-job-clean ................................................................... 25
   - globus-job-get-output ........................................................ 26
   - globus-job-run ..................................................................... 28
   - globus-job-status ................................................................ 31
   - globus-job-submit ................................................................. 33
5. GRAM RSL Quick Reference .................................................... 36
6. GRAM Error Message Reference ............................................. 40
   - 1. Errors .............................................................................. 40
7. Known Problems in GRAM5 .................................................... 53
   - 1. Known Problems in GRAM5 ............................................. 53
8. Usage statistics collection by the Globus Alliance ......................... 54
   - 1. GRAM5-specific usage statistics ...................................... 54
Glossary ..................................................................................... 57
Index ......................................................................................... 58
List of Tables

6.1. GRAM5 Errors ................................................................. 40
List of Examples

2.1. Minimal job using globus-job-run ................................................................. 4
2.2. Multiprocess job using globus-job-run ............................................................ 5
2.3. Staging an executable file using globus-job-run ................................................. 5
2.4. Providing an input file to a job using globus-job-run .......................................... 5
2.5. Staging an input file to a job using globus-job-run ............................................. 5
2.6. Canceling an interactive job ............................................................................... 5
2.7. Setting job environment variables with globus-job-run ...................................... 6
2.8. globus-job-submit ......................................................................................... 7
2.9. Using custom RSL clauses with globus-job-run ............................................... 7
2.10. Constructing RSL strings with globus-job-run ............................................... 8
2.11. Checking RSL Syntax ..................................................................................... 8
2.12. GRAM Authentication test .............................................................................. 9
2.13. GRAM version check ..................................................................................... 9
2.14. Basic Interactive Job ..................................................................................... 10
2.15. Basic Batch Job .......................................................................................... 10
2.16. Refreshing a Credential ................................................................................. 10
2.17. Proxy Expiration Example ........................................................................... 11
2.18. File stage in ................................................................................................. 11
2.19. File stage in shared ..................................................................................... 12
2.20. File stage out ............................................................................................... 12
2.21. Staging to scratch directory ........................................................................... 12
2.22. Cleaning up a file ......................................................................................... 12
2.23. Two phase commit example ......................................................................... 13
2.24. Restart example .......................................................................................... 13
2.25. Java example ............................................................................................... 14
Chapter 1. GRAM5 Overview

GRAM provides a uniform, remote interface for executing jobs on compute resources. GRAM jobs consist of file transfers and program execution on one or more compute elements managed by a local resource manager. The GRAM client can submit the job and then later poll for its status, or it can request that the GRAM service notify it when the job changes state or completes. While the job is executing, the client may send control messages to the GRAM service to monitor or modify the job. GRAM provides reliable job submission, job recovery in case of service or client failures, file staging, and asynchronous notification messages.

GRAM achieves its uniform interface by implementing a domain-specific language called the Resource Specification Language (RSL) which provides a simple way to express job requirements, environment, and commands in a specification which is independent of the local resource manager which will actually execute the job.

The GRAM protocol is a two-phase protocol, so that when jobs are submitted to a GRAM service, they will not start until a client has received a contact handle to the job. The GRAM service will not clean up a job until it has received acknowledgment from the client that the job completion state has been received. In the case of transient errors, GRAM clients can reconnect to the GRAM service to determine job state, or to update information the job will need to stage output files.

The GRAM service has been built to work in the presence of client and service failures without losing state information about jobs. If a client exits and is restarted, it can request job state information, update URLs for output files to be staged to, and register a new address to receive job state callbacks. If a service exits and is restarted, it will resume processing all existing GRAM jobs from their previous state, and continue to send state updates to any clients which are registered for them.

GRAM provides file staging before and after a job runs, scratch directory management, and a cache location for common files. File staging is Grid-aware and access remote storage resources via the GridFTP, ftp https, http protocols.

Because the GRAM service implements client callbacks for job state changes, clients can submit a number of jobs and be notified when each completes. This allows clients to be more responsive to changes in state than services which require polling for job completion.

1. GRAM Client Tools

There are a number of GRAM clients which can be used to interact with the GRAM service. The Globus Toolkit includes globusrun, globus-job-submit, and globus-job-run. Other projects provide higher-level tools which can be used to manage large sets of jobs.

1.1. Condor-G

Condor is a high-throughput job scheduler from the University of Wisconsin. It provides a facility called Condor-G to run jobs via GRAM. See the Condor documentation\(^1\), especially the section on Grid Universe, which describes how to write Condor Classified Ads to run jobs using GRAM services. The gt5 Grid type provides the best performance for using GRAM with Condor-G.

1.2. Swift

The Swift\(^2\) system from the University of Chicago is a data-oriented coarse grained scripting language that supports dataset typing and mapping, dataset iteration, conditional branching, and procedural composition. The SwiftScript

---

\(^1\) http://research.cs.wisc.edu/condor/manual/latest/index.html
\(^2\) http://www.ci.uchicago.edu/swift/main/
language can be used to create workflows that are executed on various services, including GRAM. See the Swift User's Guide for information about using Swift.

### 1.3. GridWay

The GridWay Metascheduler enables large-scale, reliable and efficient sharing of computing resources: clusters, supercomputers, stand-alone servers. It supports different LRM systems (PBS, SGE, LSF, Condor) within a single organization or scattered across several administrative domains. The GridWay manual describes how to use GridWay.

### 1.4. GRAM APIs

In addition to the tools above, you can write your own GRAM clients, using the public APIs described in the GRAM Developer's Guide. The client APIs there can be used to write custom applications that interact with GRAM services in C/C++ or Java.

### 2. Portals and Science Gateways that use GRAM

#### 2.1. XSEDE

XSEDE provides a number of domain-specific science gateways and portals, which provide interfaces to various computation and data resources, including some managed by GRAM.

These include CGD’s Atmospheric Modeling & Predictability Section from NCAR, the UltraScan LIMS Portal at the Bioinformatics Core Facility at the University of Texas Health Science Center at San Antonio, the Social Informatics Data (SID) Grid at the University of Chicago, and Southern California Earthquake Center headquartered at the University of Southern California.

---

3 [http://www.ci.uchicago.edu/swift/guides/userguide.php](http://www.ci.uchicago.edu/swift/guides/userguide.php)
6 [https://www.xsede.org](https://www.xsede.org)
7 [https://www.xsede.org/gateways-overview](https://www.xsede.org/gateways-overview)
8 [http://www.cgd.ucar.edu/amp/](http://www.cgd.ucar.edu/amp/)
9 [http://ulims.uthscsa.edu](http://ulims.uthscsa.edu)
10 [https://sidgrid.ci.uchicago.edu](https://sidgrid.ci.uchicago.edu)
11 [http://scec.org](http://scec.org)
Chapter 2. Using GRAM5

1. Before Getting Started

1.1. GRAM and Security

GRAM uses the Grid Security Infrastructure for its security implementation, based on X.509 certificates and the TLSv1 protocol to authenticate user identities with GRAM services. Before using GRAM, you must first obtain a security credential. This is typically done by requesting a certificate from a site-specific CA, or by using a portal to obtain a temporary credential. In typical use, GRAM uses a proxy certificate which is a short-term credential digitally signed by a private key. Please read the Section 1, “Basic procedure for using GSI C” to learn more about how to obtain and use a GSI credential before continuing this guide.

1.2. GRAM Resource Names

Before interacting with a GRAM service, you must know its contact address. GRAM uses a very flexible URL-like syntax to contain information about the service's hostname, TCP port number, service name, and security identity. In the basic case, you will only need to use the service's hostname to contact the service. However, if the service is configured to run on a non-standard port, or with a custom service name, or credential which doesn't match its hostname, you will need to use one of the longer forms.

A fully-qualified resource name looks something like grid.example.org:2120/jobmanager-sge:/C=US/O=Example/OU=Grid/CN=host/grid.example.org. Breaking this down, the resource name includes:

<table>
<thead>
<tr>
<th>Name Component</th>
<th>Example</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Name</td>
<td>grid.example.org</td>
<td>Host which the GRAM service is running on.</td>
<td>None. This is always a required component.</td>
</tr>
<tr>
<td>TCP Port</td>
<td>2120</td>
<td>TCP port which the GRAM service is listening on. If multiple GRAM services are running on the same machine, they may use alternate TCP ports.</td>
<td>2119</td>
</tr>
<tr>
<td>Service Name</td>
<td>jobmanager-sge</td>
<td>The name of the GRAM service on the given host. A host may provide access to multiple resources using different local resource managers. This name is used to distinguish which service to use for a particular job request. Typically, a host will provide a default entry called jobmanager which will interface</td>
<td>jobmanager</td>
</tr>
</tbody>
</table>

1 http://www.ietf.org/rfc/rfc3820.txt
Using GRAM5

<table>
<thead>
<tr>
<th>Name Component</th>
<th>Example</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/C=US/O=Example/OU=Grid/CN=host/grid.example.org</td>
<td>with a batch computing or high-throughput scheduling system, and another called jobmanager-fork for simple non-compute jobs.</td>
<td>host@hostname</td>
</tr>
</tbody>
</table>

Credential Name

<table>
<thead>
<tr>
<th>Credential Name</th>
<th>Example</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/C=US/O=Example/OU=Grid/CN=host/grid.example.org</td>
<td>The name of the credential which the GRAM service is using. This is only needed if the credential's common name does not match the host name.</td>
<td>host@hostname</td>
</tr>
</tbody>
</table>

Any component of the resource name may be omitted, except for the host name, and defaults will be used. The field separator : must be retained when skipping between name components.

2. Basic Client Interface

This section contains the basic command-line interface for interacting with gram services. For these examples, we will use the GRAM resource named grid.example.org:2119/jobmanager-pbs. You will need to change that to resources which you have been granted access.

2.1. Batch and Interactive Use

The tools `globus-job-run` and `globus-job-submit` an both be used to submit jobs to GRAM resources. The difference is that `globus-job-run` will wait until the job terminates before exiting and prints job standard output and stderr after the job completes, while `globus-job-submit` will submit the job and then exit immediately, printing the job contact to its standard output stream. The job can be then polled for status with the `globus-job-status` command, its output can be fetched with the `globus-job-get-output` and cleaned up with the `globus-job-clean` command.

2.2. Running Basic Jobs with `globus-job-run`

The `globus-job-run` provides a simple blocking command-line interface to the GRAM service. The `globus-job-run` program submits a job to a GRAM resource and waits for the job to terminate. After the job terminates, the output and error streams of the job are sent to the output and error streams of `globus-job-run`. Note that truly interactive jobs are not supported with GRAM.

The `globus-job-run` program has command-line options to control most aspects of jobs run by GRAM. However, certain behaviors must be specified by definition of an RSL string containing various job attributes. A more detailed description about the RSL language is included on the section on running jobs with `globusrun` below.

The following examples show some of the common command-line options to `globus-job-run`. Full `globus-job-run` documentation is available in the GRAM5 public interface guide.

Example 2.1. Minimal job using `globus-job-run`

The following command line submits a single instance of the `/bin/hostname` executable to the resource named by `grid.example.org/jobmanager-pbs`.

```bash
% globus-job-run grid.example.org/jobmanager-pbs /bin/hostname
node1.grid.example.org
```
Example 2.2. Multiprocess job using globus-job-run

The following command line submits ten instances of an executable /bin/hostname. The output of the job is the name of the ten hosts that the job ran on. The -np COUNT option causes globus-job-run to run COUNT instances of the executable.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -np 4 /bin/hostname
node1.grid.example.org
node3.grid.example.org
node2.grid.example.org
node10.grid.example.org
```

Example 2.3. Staging an executable file using globus-job-run

The following command line submits an executable which is local to the submit machine to the GRAM resource, then executes it. The executable is removed automatically from the GRAM resource after the job completes. The -s option prior to the executable name causes globus-job-run to stage the executable using GASS (an https-based protocol) from the machine running globus-job-run to the GRAM resource.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -s my-executable
node1.grid.example.org
```

Example 2.4. Providing an input file to a job using globus-job-run

The following command line submits a job to a GRAM resource. When this job runs, its standard input will read from the file $HOME/inputfile.txt, which is located on the GRAM resource. The -stdin command-line option indicates this path.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -stdin inputfile.txt /bin/cat
Hello, Grid
```

Example 2.5. Staging an input file to a job using globus-job-run

The following command line submits a job to a GRAM resource. When this job runs, its standard input will read from the file inputfile.txt, which is located on the submit client machine. The -stdin -s command-line option combination causes the input to be staged in the above executable staging example.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -stdin -s inputfile.txt /bin/cat
Hello, staged input on the Grid
```

Example 2.6. Canceling an interactive job

This example shows how using the Control+C (or other system-specific mechanism for sending the SIGINT signal) can be used to cancel a GRAM job.

```bash
% globus-job-run grid.example.org/jobmanager-pbs /bin/sleep 90
Control+C
GRAM Job failed because the user cancelled the job (error code 8)
```
Example 2.7. Setting job environment variables with globus-job-run

The following command line submits one instances of the executable /usr/bin/env, setting some environment variables in the job environment beyond those set by GRAM5. The -env VARIABLE=VALUE command-line option adds the named variable to the job environment. It may be present multiple times in the command-line to set multiple environment variables.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -env TEST=1 -env GRID=1 /usr/bin/env
```

### 2.3. Submitting Basic Jobs with globus-job-submit

A related tool to globus-job-run is globus-job-submit. This command submits a job to a GRAM5 service then exits without waiting for the job to terminate. Other tools (globus-job-cancel, globus-job-clean, and globus-job-get-output) allow further interaction with the job.

⚠️ **Important**

When using globus-job-submit, the job output and state will remain on disk on the GRAM resource until one of globus-job-clean or globus-job-cancel is run for that job. Be sure to clean up your jobs!

The globus-job-submit program has most of the same command-line options as globus-job-run. When run, instead of displaying the output and error streams of the job, it prints the job contact, which is used with the other globus-job tools to interact with the job.
Example 2.8. `globus-job-submit`  

This example shows the interaction of submitting a job via `globus-job-submit`, checking its status with `globus-job-status`, getting its output with `globus-job-get-output`, and then cleaning the job with `globus-job-clean`. Note that this example uses the `jobmanager-fork` service when retrieving input and cleaning the job. This allows those tasks to be done without waiting in the batch system. Most sites will allow these sorts of administrative jobs to be run on the GRAM node, but consult your system administrator to be sure.

Also, note, that the job contact returned from `globus-job-submit` can be used to get information about the job from any computer, provided you have GRAM tools installed and your security environment set up.

```bash
% globus-job-submit grid.example.org/jobmanager-pbs /bin/hostname
https://grid.example.org:38843/16001600430615223386/5295612977486013582/
% globus-job-status https://grid.example.org:38843/16001600430615223386/5295612977486013582/
PENDING
% globus-job-status https://grid.example.org:38843/16001600430615223386/5295612977486013582/
ACTIVE
% globus-job-status https://grid.example.org:38843/16001600430615223386/5295612977486013582/
DONE
% globus-job-get-output -r grid.example.org/jobmanager-fork \
  https://grid.example.org:38843/16001600430615223386/5295612977486013582/\nnode1.grid.example.org
% globus-job-clean -r grid.example.org/jobmanager-fork \
  https://grid.example.org:38843/16001600430615223386/5295612977486013582/

  WARNING: Cleaning a job means:
  - Kill the job if it still running, and
  - Remove the cached output on the remote resource

  Are you sure you want to cleanup the job now (Y/N) ?

  y

  Cleanup successful.

2.4. Advanced Jobs with `globus-job-run`

Example 2.9. Using custom RSL clauses with `globus-job-run`  

The following command line submits an mpi job using `globus-job-run`, setting the `jobtype` RSL attribute to `mpi`. Any RSL attribute understood by the LRM can be added to a job via this method.

```bash
% globus-job-run grid.example.org/jobmanager-pbs -np 5 -x '{jobtype=mpi}' a.out
Hello, MPI (rank: 0, count: 5)
Hello, MPI (rank: 3, count: 5)
Hello, MPI (rank: 1, count: 5)
Hello, MPI (rank: 4, count: 5)
Hello, MPI (rank: 2, count: 5)
```
Example 2.10. Constructing RSL strings with globus-job-run

The `globus-job-run` program can also generate the RSL language description of a job based on the command-line options given to it. This example combines some of the features above and prints out the resulting RSL. This RSL string can be passed to tools such as `globusrun` to be run later.

```plaintext
% globus-job-run -dumpstrl grid.example.org/jobmanager-pbs -np 5 -x '&(jobtype=mpi)' -env GRID=1 -env TEST=1 a.out
&(jobtype=mpi)
(executable="a.out")
(environment= ("GRID" "1") ("TEST" "1"))
(count=5)
```

3. Advanced GRAM Client with the globusrun tool

The `globusrun` tool provides a more flexible tool for submitting, monitoring, and canceling jobs. With this tool, most of the functionality of the GRAM5 APIs are made available from the command-line.

One major difference between `globusrun` and the other tools described above is that `globusrun` uses the RSL language to provide the job description, instead of multiple command-line options to describe the various aspects of the job. The section on `globus-job-run` contained a brief example RSL in the -dumpstrl example above.

The following sections show examples of the different modes that `globusrun` can run in. Full information about `globusrun` command-line options is available in the public interface guide.

3.1. Checking RSL Syntax

This example shows how to check that an RSL document contains a syntactically correct job description. Note that this mode does not do semantic validation of the RSL, so an RSL document that passes this test may not work when submitted to a GRAM5 service.

Example 2.11. Checking RSL Syntax

```plaintext
% globusrun -p "/(executable=a.out)"
RSL Parsed Successfully...

% globusrun -p "/executable=a.out"
ERROR: cannot parse RSL &/executable=/adfadf/adf /adf /adf)
```

Use -help to display full usage

3.2. Checking Service Contacts

This example shows how to check that a `globus-gatekeeper` is running at a particular contact and that the client and service have mutually-trusted credentials.
Example 2.12. GRAM Authentication test

% `globusrun` -a -r grid.example.org/jobmanager-pbs
GRAM Authentication test successful
% `globusrun` -a -r grid.example.org/jobmanager-lsf
GRAM Authentication test failure: the gatekeeper failed to find the requested service
% `globusrun` -a -r grid.example.org/jobmanager-pbs:host@not.example.org
GRAM Authentication test failure: an authorization operation failed
globus_xio_gsi: gss_init_sec_context failed.
GSS Major Status: Unexpected Gatekeeper or Service Name
globus_gsi_gssapi: Authorization denied: The name of the remote host (host@not.example.org), and the expected name for the remote host (grid.example.org) do not match. This happens when the name in the host certificate does not match the information obtained from DNS and is often a DNS configuration problem.

Note

The DNS configuration problem was a common issue in GRAM2, but GRAM5 will not depend on DNS to resolve names for mutual authentication.

3.3. Checking GRAM service version

This example shows how to determine what software version of GRAM5 is deployed at a particular service contact.

Example 2.13. GRAM version check

% `globusrun` -j -r grid.example.org/jobmanager-pbs:host@not.example.org
Toolkit version: 4.3.0-HEAD
Job Manager version: 10.5 (1256257907-0)

Note

This example shows the version number for an unreleased development version of GRAM5. The actual numbers returned will be different.

Note

This feature is new in GRAM5. When contacting a GRAM2 service, `globusrun` will display the following error message:

GRAM version check failed : an incoming HTTP message did not contain the expected info

3.4. Basic Interactive job with globusrun

This example shows how to submit interactive job with `globusrun`. When the `-s` is used, the output of the job command is returned to the client and displayed as if the command ran locally. This is similar to the behavior of the `globus-job-run` program described above.
Using GRAM5

Example 2.14. Basic Interactive Job

```bash
% globusrun -s -r example.grid.org/jobmanager-pbs "&(executable=/bin/hostname)(count=5)"
node03.grid.example.org
node01.grid.example.org
node02.grid.example.org
node05.grid.example.org
node04.grid.example.org
```

**3.5. Basic batch job with globusrun**

This example shows how to submit, monitor, and cancel a batch job using `globusrun`. This method is useful for the case where the job may run for a long time, the job may be queued for a long time, or when there are network reliability issues between the client and service.

Example 2.15. Basic Batch Job

```bash
% globusrun -b -r grid.example.org/jobmanager-pbs "&(executable=/bin/sleep)(arguments=500)"
globus_gram_client_callback_allow successful
GRAM Job submission successful
https://grid.example.org:38824/16001608125017717261/5295612977486019989/
GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING
% globusrun -status https://grid.example.org:38824/16001608125017717261/5295612977486019989/
% globusrun -k https://grid.example.org:38824/16001608125017717261/5295612977486019989/
```

**3.6. Refreshing a GRAM5 Credential**

The following example shows how to refresh the credential used by a job manager and a job.

Example 2.16. Refreshing a Credential

```bash
% globusrun -refresh-proxy https://grid.example.org:38824/16001608125017717261/5295612977486019989/
% echo $?
0
```

**Note**

In GT 5.2.4, `globusrun` does not print any diagnostics when given the `-refresh-proxy` command-line option. Therefore, check the exit code as above to ensure that the refresh is successful.

**3.7. Dealing with credential expiration**

When the Job Manager's credential is about to expire, it sends a message to all clients registered for `GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED` notifications that the job manager is terminating and that the job will continue to run without the job manager.

Any client which receives such a message can (if necessary) generate a new proxy as described above and then submit a restart request to start a job manager with a new credential. This job manager will resume monitoring the jobs which were started prior to proxy expiration.

In this example, the `globusrun` displays an error message when the job manager's proxy is about to expire. The user creates a new proxy and resumes monitoring the job with `globusrun`. 

---

10
Example 2.17. Proxy Expiration Example

% globusrun -r grid.example.org "&(executable=a.out)"
gram5_client_callback_allow successful
GRAM Job submission successful
GLOBUSGRAM_PROTOCOL_JOB_STATE_ACTIVE
GLOBUSGRAM_PROTOCOL_JOB_STATE_FAILED
GRAM Job failed because the user proxy expired (job is still running) (error code 131)
% grid-proxy-init
Your identity: /DC=org/DC=example/OU=grid/CN=Joe User
Enter GRID pass phrase for this identity:
Creating proxy ................................................................. Done
Your proxy is valid until: Tue Nov 10 04:25:03 2009
% globusrun -r grid.example.org "&(restart="https://grid.example.org:1997/16001700477575114131/5295612977486005428/")"
gram5_client_callback_allow successful
GRAM Job submission successful
GLOBUSGRAM_PROTOCOL_JOB_STATE_ACTIVE
GLOBUSGRAM_PROTOCOL_JOB_STATE_DONE

3.8. File staging

In addition to the standard output and error stream output done by globusrun, GRAM5 can do basic file management tasks to stage files to the GRAM5 service node before submitting a job and to stage files from the GRAM5 service node to a file service after the job completes.

GRAM5 file staging supports four URL schemes: ftp, gsiftp, http, and https. Note, that for the https scheme, GRAM expects the file server to be running with the same identity as the client.

General file staging is controlled by three RSL attributes: file_stage_in, file_stage_in_shared, and file_stage_out. In addition, the files named by the RSL attributes executable, stdin may be staged in and the files named by the RSL attributes stdout and stderr may be staged out.

The file_stage_in_shared RSL attribute instructs GRAM to store a local copy of the resource named by the URL in the GASS cache. This is useful if multiple concurrent jobs will be accessing one or more common files. The GASS cache will manage a reference count for files in the cache and remove them when all jobs that refer to them complete.

The following example shows how to stage a few files from a GridFTP server to the GRAM node. It uses the rsl_substitution mechanism to define a substitution variable to reduce the amount of redundancy in the job description.

Example 2.18. File stage in

% globusrun -s -r grid.example.org/jobmanager-pbs \
"&(rsl_substitution = (GRIDFTP_SERVER gsiftp://gridftp.example.org)) \ 
(exeucutable=/bin/ls) \
(arguments=/tmp/staged_file) \
(file_stage_in = ($(GRIDFTP_SERVER)/staged_file /tmp/staged_file))"
/tmp/staged_file

The next example uses the file_stage_in_shared RSL attribute to stage a file into the cache. The file is transferred from the client using the GASS https server embedded in the globusrun program when the -s option is used.
Example 2.19. File stage in shared

```bash
% globusrun -s -r grid.example.org/jobmanager-pbs \
  "& (executable=/bin/ls) \ 
  (arguments = -l /tmp/staged_file_link1 /tmp/staged_file_link1) \ 
  (file_stage_in_shared = \ 
   (${GLOBUSRUN_GASS_URL}/staged_file1 /tmp/staged_file_link1))"
```

```
lrwxr-xr-x  1 juser   juser  120 Nov 11 20:37 /tmp/staged_file1 -> /home/juser/.globus/.gass_cache/local/md5/ff/771bded8a2c7dacc1a1c0fecafa0ce/md5/39/13ab3db7fc002ed54012083ae6ed1c/data
```

The final staging example uses the `file_stage_out` RSL attribute to transfer a file from the GRAM service to an FTP server using anonymous FTP.

Example 2.20. File stage out

```bash
% globusrun -r grid.example.org/jobmanager-pbs \
  "& (executable=a.out) \ 
  (file_stage_out = (results.txt ftp://anonymous:nopass@ftp.example.org/incoming/results.txt))"
```

**Note**

In all of the above cases, multiple files may be staged using any combination of the supported URL schemes.

### 3.9. Temporary files and cleanup

GRAM5 supports creating a per-job scratch directory which can be used as a place to store files that will be automatically removed by GRAM when the job completes. It also supports an explicit list of files to remove when the job completes.

This example shows how to stage files into a scratch directory. It again uses the embedded GASS https server, stages to the GRAM service, then runs `/bin/ls` in the temporary directory. After the job completes, the contents of `$(SCRATCH_DIRECTORY)` and the directory itself are removed.

**Example 2.21. Staging to scratch directory**

```bash
% globusrun -s grid.example.org/jobmanager-pbs \
  "&(scratch_dir = \$(HOME)) \ 
  (directory = \$(SCRATCH_DIRECTORY)) \ 
  (file_stage_in = \ 
   (${GLOBUSRUN_GASS_URL}/inputfile $(SCRATCH_DIRECTORY)/inputfile)) \ 
  (executable = /bin/ls)"
```

```
inputfile
```

This example shows how to explicitly remove a file that was created by the job.

**Example 2.22. Cleaning up a file**

```bash
% globusrun -s grid.example.org/jobmanager-pbs \
  "&(executable = /bin/touch) \ 
  (arguments = temporary_file) \ 
  (file_clean_up = temporary_file)"
```

%
3.10. Reliable job submit

The `globusrun` command supports a two-phase commit protocol to ensure that the client knows the contact of the job which has been created so that it can be monitored or canceled in the case of a client or service error. The two-phase commit affects both job submission and termination.

The two-phase protocol is enabled by using the `two_phase` RSL attribute, as in the next example. When this is enabled, job submission will fail with the error `GLOBUSGRAM_PROTOCOL_ERROR_WAITING_FOR_COMMIT`. The client must respond to this signal with either the `GLOBUSGRAM_PROTOCOL_JOB_SIGNAL_COMMIT_REQUEST` or `GLOBUSGRAM_PROTOCOL_JOB_SIGNAL_COMMIT_EXTEND` signals to either commit the job to execution or delay the commit timeout. One of these signals must be sent prior to the two phase commit timeout, or the job will be discarded by the GRAM service.

A two phase protocol is also used at job termination if the `save_state` RSL attribute is used along with the `two_phase` attribute. When the job manager sends a callback with the job state set to `GLOBUSGRAM_PROTOCOL_JOB_STATE_DONE` or `GLOBUSGRAM_PROTOCOL_JOB_STATE_DONE`, it will wait to clean up the job until the two phase commit occurs. The client must reply with the `GLOBUSGRAM_PROTOCOL_JOB_SIGNAL_COMMIT_END` signal to cause the job to be cleaned. Otherwise, the job will be unloaded from memory until a client restarts the job and sends the signal.

**Example 2.23. Two phase commit example**

In this example, the user submits a job with a `two_phase` timeout of 30 seconds and the `save_state` attribute. The client must send commit signals to ensure the job runs.

```bash
% globusrun -r grid.example.org/jobmanager-pbs \
  "&(two_phase = 30) \ 
  (save_state = yes) \ 
  (executable = a.out)"

globus_gram_client_callback_allow successful
GRAM Job submission successful
GLOBUSGRAM_PROTOCOL_JOB_STATE_PENDING
GLOBUSGRAM_PROTOCOL_JOB_STATE_ACTIVE
GLOBUSGRAM_PROTOCOL_JOB_STATE_DONE
%
```

3.11. Reconnecting to a job

If a job manager or client exits before a job has completed, the job will continue to run. The client can reconnect to a job manager and receive job state notifications and output using the `restart` RSL attribute.

**Example 2.24. Restart example**

This example uses `globus-job-submit` to submit a batch job and then `globusrun` to reconnect to the job.

```bash
% globus-job-submit grid.example.org/jobmanager-pbs /bin/sleep 90 
https://grid.example.org:38824/16001746665595486521/5295612977486005662/
% globusrun -r grid.example.org/jobmanager-pbs \
  "&(restart = https://grid.example.org:38824/16001746665595486521/5295612977486005662/)

globus_gram_client_callback_allow successful
GRAM Job submission successful
GLOBUSGRAM_PROTOCOL_JOB_STATE_DONE
%
```
3.12. Submitting a Java job

To submit a job that runs a java program, the client must ensure that the job can find the Java interpreter and its classes. This example sets the default PATH and CLASSPATH environment variables and uses the shell to locate the path to the java program.

Example 2.25. Java example

This example uses globus-job-submit to submit a java job, staging a jar file from a remote service.

```bash
% globusrun -r grid.example.org/jobmanager-pbs \
"&(environment = (PATH '/usr/bin:/bin') (CLASSPATH \$(SCRATCH_DIRECTORY)))
(scratch_dir = \$(HOME))
(directory = \$(SCRATCH_DIRECTORY))
(rsl_substitution = (JAVA_SERVER http://java.example.org))
(file_stage_in =
  \$(JAVA_SERVER)/example.jar \$(SCRATCH_DIRECTORY)/example.jar
  \$(JAVA_SERVER)/support.jar \$(SCRATCH_DIRECTORY)/support.jar)
(exeutable=/bin/sh)
(arguments=-c 'java -jar example.jar')"

globus_gram_client_callback_allow successful
GRAM Job submission successful
GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING
GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE
GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE
%
```
Chapter 3. Troubleshooting

1. GRAM Client Troubleshooting

1.1. Credential Problems

GRAM requires a client certificate and private key in order to authenticate with the GRAM service. If these are not available, the GRAM client will fail. In typical use, a user will create a temporary proxy certificate either derived from their identity certificate issued by some certificate authority, or from a service such as myproxy. If a GRAM client command returns any error containing the string `GSS Major Status` you’ve hit a credential problem. Look at the Troubleshooting Section of the GSI manual for details about how to diagnose and correct these errors. The `grid-cert-diagnostics` tool with the `-p` command-line option is especially helpful for diagnosing some of these types of problems.

1.2. Connection Problems

There are a few things which can go wrong when trying to contact a GRAM service. These have slightly different error types which can help diagnose which problem is occurring.

1.2.1. Invalid Resource Name

If the hostname or TCP port you are using for a GRAM resource name is not correct, then the GRAM client will be unable to access the service. Errors of this type will look like this:

```bash
% globus-job-run grid.example.org/jobmanager-fork /bin/hostname
GRAM Job submission failed because the connection to the server failed (check host and port) (error code 12)
```

When this occurs, check with the resource administrator for correct resource naming so that you can contact the service.

1.2.2. Mutual Authentication Failure

GRAM performs mutual authentication, that is, both the client and service provide certificates indicating who they are. The service uses the client’s identity to map the user to a local unix account. The client uses the server’s identity to verify that the service is running with a host credential. The failure of the client to trust the server's certificate will generate an error message that looks like this:

```bash
globus_gsi_gssapi: Authorization denied: The expected name for the remote host (host@alias.example.org) does not match the authenticated name of the remote host (host@grid.example.org). This happens when the name in the host certificate does not match the information obtained from DNS and is often a DNS configuration problem.
```

This mismatch can happen for a number of reasons: a site administrator has multiple hosts sharing a certificate, a host has multiple DNS aliases, and the client is not aware of which name the server is using for its certificate, or a host's name has changed since the certificate was issued. The remedy for the client, after confirming with the GRAM administrator that the name after "authenticated name of the remote host" is the correct certificate name is to use a form of the GRAM resource name which includes this name. For example, explicitly adding a name to the abbreviated GRAM contact so that instead of `alias.example.org`, you would use `alias.example.org::host@grid.example.org`. 

1.2.3. Certificate Trust Issues

Because of the mutual authentication, both GRAM users and services can hit problems if they do not trust their peer's certificate or the Certificate Authority which issued it. If the client doesn't trust the server's certificate, it is easier to diagnose, because the GRAM service doesn't send much information back to the client if it doesn't trust it. However, working with the system administrator to get information from the GRAM logs will usually fix these problems fairly easily.

If the service's certificate is not trusted, the client will receive a message like this:

```
% globus-job-run grid.example.org /bin/hostname
GRAM Job submission failed because an authentication operation failed
OpenSSL Error: s3_clnt.c:915: in library: SSL routines, function SSL3_GET_SERVER_CERTIFICATE
globus_gsi_callback_module: Could not verify credential
globus_gsi_callback_module: Can't get the local trusted CA certificate: Untrusted self-signed certificate
```

This error indicates that certificate chain from the service certificate to the client contained a self-signed certificate (usually an indication that it's a CA certificate), which the client doesn't trust, and includes the hash of the certificate name (bbfccedf in this case). If you hit this particular type of error, you should send the information to the GRAM administrator and determine which CA should be trusted and what its signing policy is, to determine if you want to add it to your local set of trust roots.

Note

Different versions of OpenSSL produce different hashes for the same certificate names. If you upgrade a system (or transfer CA certificates between systems) to a different version of OpenSSL, you may hit this problem even if you think you have the CA certificate in your trusted certificate directory. If so, run the `globus-update-certificate-dir` program to update your hashes.

There are other reasons why a certificate might not be trusted (it's in a revoked list, it has expired or was issued in the future, etc). For more details look at the troubleshooting information in the GSI user's guide.

If for some reason the service does not trust your certificate, you'll get a rather cryptic message from GRAM that looks like this:

```
% globus-job-run grid.example.org /bin/hostname
GRAM Job submission failed because an authentication operation failed
globus_gsi_gssapi: Unable to verify remote side's credentials
globus_gsi_gssapi: Unable to verify remote side's credentials: Couldn't verify the remote certificate
OpenSSL Error: s3_pkt.c:1086: in library: SSL routines, function SSL3_READ_BYTES: sslv3 alert bad certificate
```

To remedy this, consult the GRAM administrator to get information from the `/var/log/globus-gatekeeper.log` file to determine the reason why the gatekeeper didn't like your certificate. Again it could be CA trust issues, clock skew, or a revoked certificate. The error in the gatekeeper log would typically look like the client-side trust issue above.

1.2.4. Authentication with the Remote Server Failed

Once the GRAM service has authenticated the client, it maps the client's identity to a local user account using a gridmapfile or other mapping service. If this fails, the client will receive a message that looks like this:

```
% globus-job-run grid.example.org /bin/hostname
GRAM Job submission failed because authentication with the remote server failed
```
To remedy this, consult the system administrator of the GRAM resource to be added to the authorized user's list. Be sure to send your credential subject name to make it easier for them. To get that information, run the command `grid-cert-info -s`.

### 1.2.5. Unable to Find the Requested Service

Recall that a GRAM resource name includes a component called the service name. The default if not specified is `jobmanager`, but some sites may not use that name, or have a different LRM name than you expect. If you specify an incorrect service name, or the default is not present, you'll get an error that looks like this:

```
% grid-job-run grid.example.org /bin/hostname
GRAM Job submission failed because the gatekeeper failed to find the requested service (error code 93)
```

If you get this error, you'll need to determine which services are available on that GRAM resource, either by asking the admin or by looking at the entries in `/etc/grid-services`

### 1.2.6. Failed to Run the Job Manager

The GRAM service is split between a privileged process called the `globus-gatekeeper` and a non-privileged process called the `globus-job-manager` which runs as a user process. If the `globus-gatekeeper` is unable to locate the `globus-job-manager` process, then this misconfiguration will show up like this:

```
% grid-job-run grid.example.org /bin/hostname
GRAM Job submission failed because the gatekeeper failed to run the job manager (error code 47)
```

This is an installation mistake, and the administrator of the GRAM resource must fix this.

### 1.3. Jobs are Hanging

One problem GRAM users sometimes encounter is that it looks like jobs submitted to GRAM are not making any progress, even though the local resource manager thinks they've run. There are a couple of reasons why this might occur: GRAM is not getting the information it needs from the local resource manager or the GRAM client is not getting the information it needs. We'll cover diagnosing and handling the latter case in this document, as the other is an system administrator issue.

The way `globus-job-run` and `globusrun` determine that jobs have completed is via GRAM job state callbacks. These are messages sent by the GRAM service to the client node indicating that something significant has happened in the lifecycle of the job. If for some reason the GRAM service can not get those messages to the client, the client will not be able to detect job state changes.

In order to determine if this is the case, submit a job using `globus-job-submit`, and then use the `globus-job-status` command to see if the job state changes. If it does not, then consult the GRAM administrator---there might be some problem with the installation. If it does, then for some reason the callbacks are not happening. This might be firewall issues or host naming issues.

The GRAM client sends a "callback contact" to the GRAM service when it submits a job, in order that it can receive notifications. This contact is a reference to a https server embedded in the GRAM client which only handles GRAM state callbacks. As with all web servers, it has a URL which defines how to contact it, which in this case consists of the client host name and the service port number. If the host name that is used is not resolvable (such as a for a laptop with a dynamic address), then the GRAM service will not be able to contact it. If that's the case, you can set the `GLOBUS_HOSTNAME` environment variable to the IP address that your client can be reached at, and then submit your jobs. This will cause GRAM to publish that address instead of what it thinks the client's host name is.

Another way that the GRAM service would be unable to send job state updates to a client would be if there's a firewall between the service and the client. If that's the case, you might need to set the `GLOBUS_TCP_PORT_RANGE`
Troubleshooting

environment variable to a comma-separated list of numbers which represent a range of minimum and maximum TCP port numbers to listen on. You might have to contact your site administrator to determine what TCP ports are allowed. If there are none, you can still use `globus-job-submit` and `globus-job-status` to track your job's state changes, or use another tool like those mentioned in the section about client tools.

1.4. Logs and Debugging

The GRAM service has a log file which contains information about the job as it is processed. These logs are located by default in `/var/log/globus/gram_$USERNAME.log`. There are some different logging levels available, as described in the GRAM Administrator's Guide. These can be controlled on a per-job basis by adding the `loglevel` RSL attribute to your job description. The default is to log only `FATAL` and `ERROR` messages, but other levels can sometimes help understand what is going on.

1.5. Diagnosing LRM Errors

Sometimes, bugs creep into the LRM adapter scripts. When that occurs, the GRAM job will usually fail with an error like this:

```
GRAM Job failed because the job manager detected an invalid script status (error code 25)
```

If this occurs, you may have to work with a GRAM administrator to help debug this problem. One helpful thing you can do when reporting it is to save the GRAM internal script data so that it can be used outside of the GRAM service to see what the low-level error looks like. To do this, add the RSL fragment `savejobdescription = yes` to your job request. This will cause GRAM to leave a file called something like `~/.globus/gram_[0-9]*.pl` in your home directory. You can use this with the internal tool `/usr/share/globus/globus-job-manager-script.pl` to try to submit the job to the LRM without using the GRAM service. The command line `/usr/share/globus/globus-job-manager-script.pl -m LRM -c submit -f GRAM-PL-FILE` will attempt to submit the job to the LRM. It will show all the information the LRM script sends to the GRAM service, which might include some perl-language error or badly formatted output from the script (which must only output lines which begin with `GRAM_SCRIPT_`.

In some extreme cases, the `savejobdescription` option will not generate a file. If that's the case, pass `/dev/null` as the argument to the `-f` command-line option. The problem is likely a perl syntax error which will be reached before the job description is loaded.

1.6. Email Support

If all else fails, please send information about your problem to `<gram-user@globus.org>`. You'll have to subscribe to a list before you can send an e-mail to it. See here¹ for general e-mail lists and information on how to subscribe to a list and here² for GRAM specific lists. Depending on the problem, you may be requested to file a bug report to the globus project's Issue Tracker³.

---

¹ http://dev.globus.org/wiki/Mailing_Lists
² http://dev.globus.org/wiki/GRAM#Mailing_Lists
³ http://jira.globus.org
Chapter 4. Command-line Client Reference Pages

This section contains reference pages for all of the tools described in the previous section. These pages contain all the command-line options for these tools. These are available as manpages in the documentation subpackages for the globus-gram-client-tools package.
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 diagnostic information and then exit.

When the -p or -parse command-line option is present, globusrun will verify the syntax of the RSL specification and then terminate. If the syntax is valid, globusrun will print out the string "RSL Parsed Successfully..." 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`
• 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 the 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 the 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)
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 successful 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 succesful 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-run — Execute a job using GRAM

Synopsis

[-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] ...
[stdin [-l | -s ] STDIN_FILE] [stdout [-l | -s ] STDOUT_FILE] [stderr [-l | -s ] STDERR_FILE]
[-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 -dumpsl, -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.
- **-dumpsl** 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.
-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-run 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-run 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-run 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-run.

-1
When included outside the context of -stdin, -stdout, or -stderr command-line options, -1 option alters the interpretation of the executable path. If the -1 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-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 cannot 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

globus-job-submit [-dumpsl] [-dryrun] [-verify]
[-file ARGUMENT_FILE]
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]...
[stdin [-l|-s] STDIN_FILE] [stdout [-l|-s] STDOUT_FILE] [stderr [-l|-s] STDERR_FILE]
[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 pro-
vided on the command-line after all other options. Note that the -dumpsl, -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 sub-
mitted 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.
-dumpsl Translate the command-line options to globus-job-submit into an RSL ex-
pression that can be used with tools such as globusrun.
-dryrun Submit the job request to the GRAM service with the dryrun option en-
abled. 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 diag-
nose 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.

-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] 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.

-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)
Chapter 5. GRAM RSL Quick Reference

The GRAM RSL language is described in detail in the GRAM Developer's Guide. For basic use, job description RSLs consist of a set of RSL attributes preceded by the & character. The basic job description looks like:

```
&
  (attribute = value )
  (attribute = value )
  ...
```

The following list contains the RSL attributes which are available in the core job manager. Other LRM-specific RSL attributes may also be available in some situations.

- **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.

- **directory**  
  Specifies the path of the directory the jobmanager will use as the default directory for the requested job.

- **dry_run**  
  If dryrun = yes then the jobmanager will not submit the job for execution and will return success.

- **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.

- **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.

- **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.)
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>library_path</td>
<td>Specifies a list of paths to be appended to the system-specific library path environment variables.</td>
</tr>
<tr>
<td>loglevel</td>
<td>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</td>
</tr>
<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 cpetime 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 cpetime, 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. Walle-time 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>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>save_state</td>
<td>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.</td>
</tr>
<tr>
<td>savejobdescription</td>
<td>Save a copy of the job description to $HOME</td>
</tr>
<tr>
<td>scratch_dir</td>
<td>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.</td>
</tr>
<tr>
<td>stderr</td>
<td>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.</td>
</tr>
<tr>
<td>stderr_position</td>
<td>Specifies where in the file remote standard error streaming should be restarted from. Must be 0.</td>
</tr>
<tr>
<td>stdin</td>
<td>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.</td>
</tr>
<tr>
<td>stdout</td>
<td>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.</td>
</tr>
<tr>
<td>stdout_position</td>
<td>Specifies where in the file remote output streaming should be restarted from. Must be 0.</td>
</tr>
<tr>
<td>two_phase</td>
<td>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 timeout value 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...</td>
</tr>
</tbody>
</table>
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 cannot 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.
## Chapter 6. GRAM Error Message Reference

### 1. 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 $GLOBUS_LOCATION/var/globus-gatekeeper.log for runtime configuration errors.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td>13</td>
<td>the provided RSL 'maxtime' value is not an integer</td>
<td>Check that the RSL.maxtime value evaluates to an integer.</td>
</tr>
<tr>
<td>14</td>
<td>the provided RSL 'count' value is not an integer</td>
<td>Check that the RSL.count value evaluates to an integer.</td>
</tr>
<tr>
<td>15</td>
<td>the job manager received an invalid RSL</td>
<td>Check that the RSL string can be parsed by using <code>globusrun -p RSL</code>.</td>
</tr>
<tr>
<td>16</td>
<td>the job manager failed in allowing others to make contact</td>
<td>Check job manager log.</td>
</tr>
<tr>
<td>17</td>
<td>the job failed when the job manager attempted to run it</td>
<td>Verify that the LRM is configured properly.</td>
</tr>
<tr>
<td>18</td>
<td>an invalid paradigm was specified</td>
<td>OBSOLETE IN GRAM2</td>
</tr>
<tr>
<td>19</td>
<td>the provided RSL 'jobtype' value is invalid</td>
<td>The RSL jobtype attribute is not indicated as supported by the LRM. Valid jobtype values are <code>single</code>, <code>multiple</code>, <code>mpi</code>, and <code>condor</code>.</td>
</tr>
<tr>
<td>20</td>
<td>the provided RSL 'myjob' value is invalid</td>
<td>OBSOLETE IN GRAM5</td>
</tr>
<tr>
<td>21</td>
<td>the job manager failed to locate an internal script argument file</td>
<td>Check that <code>$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_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>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------</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>28</td>
<td>the job manager failed to create an internal script submission file</td>
<td>Check that the user's home file system 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</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. Check that the executable 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 executable.</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>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</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>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>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>72</td>
<td>the executable file permissions do not allow execution</td>
<td>Check that the RSL <code>executable</code> attribute refers to an executable program or script.</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 'min_memory' 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 'max_memory' parameter is invalid</td>
<td>Check that the RSL <code>max_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>86</td>
<td>the RSL 'min_memory' value is not zero or greater</td>
<td>Check that the RSL min_memory attribute evaluates to a positive integer value.</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>Error Code</td>
<td>Reason</td>
<td>Possible Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>103</td>
<td>the provided RSL 'max_cpu_time' parameter is invalid</td>
<td>Check that the RSL <code>max_cpu_time</code> attribute evaluates to a positive integer.</td>
</tr>
<tr>
<td>104</td>
<td>the RSL 'max_cpu_time' value is not zero or greater</td>
<td>Check that the RSL <code>max_cpu_time</code> 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 <code>GLLOBUS_LOCATION/libexec/globus-job-manager-script.pl</code> 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 <code>save_state</code> 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 <code>restart</code> 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 <code>two_phase</code> 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 <code>two_phase</code> 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>116</td>
<td>the provided RSL 'stdout_position' parameter is invalid</td>
<td>OBSOLETE IN GRAM5</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 val-</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 scratch_dir attribute exists and is writable. Check that the directory named by the RSL scratch_dir attribute is not full.</td>
</tr>
<tr>
<td>137</td>
<td>the provided 'gass_cache' parameter is invalid</td>
<td>Check that the RSL gass_cache 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 file_stage_in 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 file_stage_in_shared 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 file_stage_out 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 gass_cache 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 file_clean_up attribute evaluates to a sequence of strings.</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>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>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 <code>executable</code> attribute is present in your job request RSL. Check that the RSL <code>restart</code> 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 <code>gass_cache</code> attribute 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 <code>save_state</code> 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 $GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf is present and readable on the job manager service node. Check that $GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf 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 $GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf is present and readable on the job manager service node. Check that $GLOBUS_LOCATION/share/globus_gram_job_manager/LRM.rvf is valid if present.</td>
</tr>
<tr>
<td>153</td>
<td>the provided RSL 'proxy_timeout' attribute is invalid</td>
<td>Check that RSL <code>proxy_timeout</code> 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 <code>proxy_timeout</code> 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.</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>service node. Check that the directory of the destination file being staged exists on the file service 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 proper-</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>
Chapter 7. Known Problems in GRAM5

1. Known Problems in GRAM5

- **GT-45**: Manager lock double-locked
- **GT-47**: globus-job-manager null pointer dereference for some call paths
- **GT-52**: SEG may deadlock with threads
- **GT-56**: Tear-down of object requires multiple threads
- **GT-103**: GRAM refresh credentials test sometimes fails because job terminates
- **GT-292**: Service tags may not isolate services completely
- **GT-311**: globus job manager is leaking memory
- **GT-324**: Behaviour of globus-job-status
- **GT-359**: SGE SEG hangs when log_path points to directory

1. http://jira.globus.org/browse/GT-45
5. http://jira.globus.org/browse/GT-103
7. http://jira.globus.org/browse/GT-311
Chapter 8. Usage statistics collection by the Globus Alliance

1. GRAM5-specific usage statistics

The following usage statistics are sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at the end of each job.

- Job Manager Session ID
- dryrun used
- RSL Host Count
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_FILE_STAGE_IN
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_FILE_STAGE_OUT
- Timestamp when job hit GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE
- Job Failure Code
- Number of times status is called
- Number of times register is called
- Number of times signal is called
- Number of times refresh is called
- Number of files named in file_clean_up RSL
- Number of files being staged in (including executable, stdin) from http servers
- Number of files being staged in (including executable, stdin) from https servers
- Number of files being staged in (including executable, stdin) from ftp servers
- Number of files being staged in (including executable, stdin) from gsiftp servers
- Number of files being staged into the GASS cache from http servers
- Number of files being staged into the GASS cache from https servers
- Number of files being staged into the GASS cache from ftp servers
- Number of files being staged into the GASS cache from gsiftp servers
Usage statistics collection by the Globus Alliance

- Number of files being staged out (including stdout and stderr) to http servers
- Number of files being staged out (including stdout and stderr) to https servers
- Number of files being staged out (including stdout and stderr) to ftp servers
- Number of files being staged out (including stdout and stderr) to gsiftp servers
- Bitmask of used RSL attributes (values are $2^i$ from the gram5_rsl_attributes table)
- Number of times unregister is called
- Value of the `count` RSL attribute
- Comma-separated list of string names of other RSL attributes not in the set defined in `globus-gram-job-manager.rvf`
- Job type string
- Number of times the job was restarted
- Total number of state callbacks sent to all clients for this job

The following information can be sent as well in a job status packet but it is not sent unless explicitly enabled by the system administrator:

- Value of the executable RSL attribute
- Value of the arguments RSL attribute
- IP address and port of the client that submitted the job
- User DN of the client that submitted the job

In addition to job-related status, the job manager sends information periodically about its execution status. The following information is sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at job manager start and every 1 hour during the job manager lifetime:

- Job Manager Start Time
- Job Manager Session ID
- Job Manager Status Time
- Job Manager Version
- LRM
- Poll used
- Audit used
- Number of restarted jobs
- Total number of jobs
- Total number of failed jobs
- Total number of canceled jobs
Usage statistics collection by the Globus Alliance

- Total number of completed jobs
- Total number of dry-run jobs
- Peak number of concurrently managed jobs
- Number of jobs currently being managed
- Number of jobs currently in the UNSUBMITTED state
- Number of jobs currently in the STAGE_IN state
- Number of jobs currently in the PENDING state
- Number of jobs currently in the ACTIVE state
- Number of jobs currently in the STAGE_OUT state
- Number of jobs currently in the FAILED state
- Number of jobs currently in the DONE state

Also, please see our policy statement\(^1\) on the collection of usage statistics.

\(^1\)/toolkit/docs/latest-stable/Usage_STATS.html
Glossary

C

Condor

A Local Resource Manager mechanism supported by GRAM. See the Condor Project Website\(^1\) for more information.

See Also Local Resource Manager.

L

Local Resource Manager (LRM)

A system which controls access to a compute resource, such as a compute cluster or parallel computer. Such systems provide batch execution interfaces, which GRAM uses to execute jobs. Condor, Portable Batch System, GridEngine are examples of local resource managers.

See Also Condor, Portable Batch System, Oracle GridEngine.

O

Oracle GridEngine

A Local Resource Manager supported by GRAM. See Oracle's Web Site\(^{10}\) for more information.

See Also Local Resource Manager.

P

Portable Batch System (PBS)

A Local Resource Manager mechanism supported by GRAM. Multiple implementations of PBS exist: GRAM currently supports TORQUE. See also TORQUE.

See Also Local Resource Manager, TORQUE.

T

TORQUE

An implementation of the Portable Batch System interface by Adaptive Computing\(^{11}\).

See Also Portable Batch System.

---

\(^1\) http://www.cs.wisc.edu/condor/


\(^{11}\) http://www.adaptivecomputing.com/products/torque.php
Index

E
errors, 40

J
jobs
  preparing
generate valid proxy, 3, 3

T
troubleshooting, 15
  errors, 15

U
usage statistics, 54