

GT 4.2.1 Migrating Guide for GRAM4

GT 4.2.1 Migrating Guide for GRAM4

Abstract

The following provides available information about migrating from previous versions of the Globus Toolkit.

Table of Contents

1. Migrating GRAM from GT4.0	1
1. Admin - Migration Guide	1
2. User - Migration Guide	1
3. Developer - Migration Guide	3
2. Migrating GRAM from GT3	7
3. Migrating GRAM from GT2	9
1. Admin - Migration Guide	9
2. User - Migration Guide	11
3. Developer - API and RSL Migration Guide	12
Glossary	21

List of Tables

3.1. Command Line Option Comparison	12
3.2. C API Migration Table	13
3.3. RSL Migration Table	15
3.4. RSL Migration Examples	17

Chapter 1. Migrating GRAM from GT4.0

The 4.2.1 protocol has been changed to be WSRF 1.2, WSN 1.3 and WS Addressing 1.0 compliant. There is no backward compatibility between 4.2.1 and 4.0.

1. Admin - Migration Guide

1.1. Default local resource manager

In GRAM4 an administrator can configure a default local resource manager being used for job submission if the client did not specify any in a job submission request. Check section [Defining a default local resource manager](#) in the [System Administrator's Guide](#) for more information.

1.2. Audit Logging

The log4j configuration for audit logging in GRAM4 in 4.2 is slightly different compared to the 4.0 series. This involves log4j and database configuration changes. For the log4j configuration check section [Configure Log4j](#) in the [System Administrator's Guide](#) for more information. For the database related changes check [Configure the Database in JNDI](#) in the [System Administrator's Guide](#).

1.3. Job lifetime

An administrator can configure lifetime parameters for all jobs in GRAM4's JNDI configuration. In short these parameters restrict the maximum lifetime a client can set on a job, and the maximum time a job is kept in the container and thus in the persistence directory after the job had been fully processed. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job lifetime configuration](#) in the [System Administrator's Guide](#) for more information.

1.4. Local Java calls from GRAM4 to RFT

By default GRAM4 in the 4.0 series does Web Service invocations when calling RFT to perform staging and file cleanup. The default in GRAM4 in 4.2 is local java calls to significantly improve performance in jobs with staging and file cleanup. If GRAM4 is configured to use RFT in a separate machine local invocations are disabled. For more information check the section [GRAM4 - RFT interaction](#) in the [System Administrator's Guide](#)

1.5. Threads for SEG event processing

An administrator can configure the number of threads in the thread-pool that is responsible for handling scheduler events. The default size should be fine for all Scheduler Event Generators (SEGs) provided by Globus. It might be interesting for custom written SEGs. Check section [Configuring thread-pools](#) in the [System Administrator's Guide](#) for more information.

2. User - Migration Guide

2.1. Specification upgrade related changes

Due to the WSRF, WSN and WS-Addressing specification upgrade some namespaces changed. All occurrences of the following old 4.0 namespaces must be updated to the new 4.2 namespaces.

4.0	4.2
http://schemas.xmlsoap.org/ws/2004/03/addressing	http://www.w3.org/2005/08/addressing
http://www.globus.org/namespaces/2004/10/gram/job	http://www.globus.org/namespaces/2008/03/gram/job
http://www.globus.org/namespaces/2004/10/rft	http://www.globus.org/namespaces/2008/04/rft

Additionally, in a job description the ReferenceProperties element must be renamed to ReferenceParameters.

The following shows a factoryEndpoint element of a 4.0 job description and a 4.2 job description:

```
# Endpoint in 4.0
<factoryEndpoint
  xmlns:gram="http://www.globus.org/namespaces/2004/10/gram/job"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <wsa:Address>
    https://grid-w.ncsa.teragrid.org:8444/wsrf/services/ManagedJobFactoryService
  </wsa:Address>
  <wsa:ReferenceProperties>
    <gram:ResourceID>Fork</gram:ResourceID>
  </wsa:ReferenceProperties>
</factoryEndpoint>

# Endpoint in 4.2
<factoryEndpoint
  xmlns:gram="http://www.globus.org/namespaces/2008/03/gram/job"
  xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Address>
    https://grid-w.ncsa.teragrid.org:8444/wsrf/services/ManagedJobFactoryService
  </wsa:Address>
  <wsa:ReferenceParameters>
    <gram:ResourceID>Fork</gram:ResourceID>
  </wsa:ReferenceParameters>
</factoryEndpoint>
```

2.2. Local resource manager

GRAM4 in 4.2 defines a default local resource manager which is used for job execution if the user did not explicitly specify one. A user however still can specify a local resource manager (LRM), e.g. if it's required that a job runs in a certain LRM. Check

- [Finding available local resource managers](#) to get information about how to get a list of available LRM's of a GRAM4 instance.
- [Finding the default local resource manager](#) to find out which LRM is the default LRM of a GRAM4 instance.
- [Specifying a local resource manager](#) to get information about how to submit to the default vs to a non-default LRM.

2.3. Job Lifetime

globusrun-ws in 4.0 sets a default lifetime of 24h on a job if a user does not explicitly set it. In 4.2 a job will run to completion in any event if no lifetime is specified, i.e. it will not be terminated after 24h.

However, there are server-side settings define limits on an explicitly requested lifetime, and that determine what will happen to a completed job if no lifetime had been set. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job Lifetime](#) in the [User's Guide](#) for more information.

3. Developer - Migration Guide

3.1. Specification upgrade related changes

Due to the WSRF, WSN and WS-Adressing specification upgrade namespaces of many components have changed. Check section [Specification upgrade related changes](#) and the [Java WS Core Migration guide](#) for details.

3.2. Job termination

GRAM4 does not extend from the ImmediateResourceTermination port type anymore, i.e. the Destroy() operation is no longer available to synchronously destroy job resources. Instead GRAM4 offers an asynchronous Terminate() method that lets clients terminate jobs and destroy a job resource. Check the TerminateManagedJobPortType PortType in the section 'WSDL and Schema Definition' of the [Developer's guide](#) for details and section [Job termination](#) for explanations of the new termination interface.

3.3. Job lifetime

The Java API GramJob in 4.0 sets a default lifetime of 24h on a job if a developer does not explicitly set it. In 4.2 it will not be set and a job will run to completion in any event if no lifetime is specified, i.e. it will not be terminated after 24h.

However, there are server-side settings define limits on an explicitly requested lifetime, and that determine what will happen to a completed job if no lifetime had been set. Check section [Job Lifetime](#) in the [Execution key concepts](#) and section [Job Lifetime](#) in the [User's Guide](#) for more information.

3.4. Subscription resources

In GRAM4 in the 4.0 series a client had to destroy subscription resources before destroying a job if it subscribed for notifications in order to cleanup state on the server-side. In GRAM4 in 4.2 all subscription resources of a job resource are destroyed on the server-side automatically if a job resource is destroyed. There's no need to keep track of subscription resources anymore.

3.5. Default local resource manager

In GRAM4 an administrator can configure a default local resource manager (LRM) being used for job submission if the client did not specify any in a job submission request. To submit a job to the default local resource manager just don't put a ReferenceParameters element in the EndpointReference used to contact GRAM4's job factory service. By querying resource properties of GRAM4's factory service a client can find out which local resource manager is the default, or which LRM's are available on the server-side.

3.6. Client-side generated UUIDS for MultiJobs

In GRAM4 in GT 4.0 the client-side generated submission being set in CreateManagedJobInputType.JobID had been used as server-side job id for multi-jobs. In GRAM4 in 4.2 a UUID is created on the server-side to manage the job and the client-side generated ID will only be stored to enable reliable job submission. A client cannot use the self-generated

id for any job operation besides a second submission call for multi-jobs anymore. See section [Client-Side Generated Submission ID](#) for more information about this.

3.7. API changes

3.7.1. Service

3.7.1.1. Namespace

All occurrences of `http://www.globus.org/namespaces/2004/10/gram/job` had been replaced by `http://www.globus.org/namespaces/2008/03/gram/job`.

3.7.1.2. ResourceProperties

- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}fault` from the `ManagedJobPortType` is an array instead of a single element now. All faults that happen during job processing will be added to that array.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}availableLocalResourceManager` All local resource managers that are configured in this GRAM4 instance.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}jobTTLAfterProcessingTime` in seconds a job resource will stay alive after a job finished processing in GRAM4 (including `fileStageOut`, `fileCleanUp`). When this time elapsed the job resource is destroyed and no longer be available for a client. A negative values means that the job resource will never be destroyed.
- New `{http://www.globus.org/namespaces/2008/03/gram/job}maxJobLifetime` Max time in seconds a user can set as initial lifetime in job submission or in subsequent `setTerminationTime` calls. A negative value means that there is no limit.

3.7.1.3. State change notification topic

The QName of the topic that clients use to subscribe for job state change information changed to `{http://www.globus.org/namespaces/2008/03/gram/job}stateChangeNotificationMessage`. (In GRAM4 in 4.0 it was `{http://www.globus.org/namespaces/2004/10/gram/job}state`)

3.7.1.4. Job states

New values of the resource property `{http://www.globus.org/namespaces/2008/03/gram/job/types}state`: `UserTerminateDone` and `UserTerminateFailed`.

3.7.1.5. Fault types

- `{http://www.globus.org/namespaces/2008/03/gram/job/release}ResourceNotReleasedFaultType`: thrown when a resource cannot not be released but it still exists.
- `{http://www.globus.org/namespaces/2008/03/gram/job/terminate}ResourceNotTerminatedFaultType`: Thrown when a resource cannot not be terminated but still exists.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}DelegatedCredentialDestroyFaultType`: Thrown when a job resource is terminated with request to destroy delegated credentials, and the delegated credential can't be destroyed. This fault is also added to the job RP `fault` and part of a notification message if a delegated credential cannot be destroyed in asynchronous termination.

- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}JobResourceExpired-FaultType`: Added to the job RP fault and part of a notification message if a resource expired.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}StagingTerminate-FaultType`: Added to the job RP fault and part of a notification message if termination of a running transfer at RFT as part of job termination failed.
- `{http://www.globus.org/namespaces/2008/03/gram/job/faults}LocalResourceManagerJobTerminateFaultType`: Added to the job RP fault and part of a notification message if termination of a job at the local resource manager as part of job termination failed.

3.7.1.6. Job destruction

GRAM4 does no longer extend from the `ImmediateResourceTermination` port type, i.e. does no longer offer the `Destroy` method. Instead jobs must be terminated. See above description.

3.7.2. Client

3.7.2.1. Java client API: `GramJob`

- There is a new method for job termination:

```
boolean terminate(  
    boolean destroyAfterCleanUp,  
    boolean continueNotifying,  
    boolean destroyDelegatedCredentials)  
    throws ResourceUnknownFaultType, DelegatedCredentialDestroyFaultType,  
           ResourceNotTerminatedFaultType
```

- The methods `destroy()`, `cancel()` are kept for convenience and implemented to do calls to `terminate()` with following settings: Resource destruction after cleanup, no notifications about success or failure of termination, destruction of delegated credentials if `GramJob` did the delegation. These methods are marked as deprecated and will be removed in 4.4.
- The method to set the duration of a job changed from

```
public void setDuration(Date duration)  
  
to  
  
public void setDuration(int hours, int minutes)
```

Check [bug 5806](http://bugzilla.globus.org/show_bug.cgi?id=5806)¹ for details.

- The default Axis stub timeout was extended from from 2min to 5min.
- No more default lifetime of a job: If a client does not explicitly set a lifetime `GramJob` does not set it. The job will run to completion then. If `GramJob` delegates credentials and no lifetime of a job has been specified, the lifetime of the delegated credentials will not explicitly be set but the `timeleft`-value of the proxy will be used. If a lifetime is specified for a job the lifetime of the delegated credentials will be the same like the lifetime of the job.
- The method `GramJob.getFault()` returns an array of `FaultType` objects instead of a single `FaultType` object.

¹ http://bugzilla.globus.org/show_bug.cgi?id=5806

- The method `isLocallyDestroyed()` was removed because it didn't make sense with the new job termination anymore.

3.7.2.2. C client API

Chapter 2. Migrating GRAM from GT3

Migrating to GT 4.2.1 from GT version 3.2:

- The 4.2.1 protocol has been changed to be WSRF compliant. There is no backward compatibility between 4.2.1 and 3.2.

API changes since GT 3.2:

- The *MJFS* `create` operation has become `createManagedJob` and, now provides the option to send a *uuid*. A client can use this uuid to recover a job EPR in the event that the reply message is not received. Given this new scheme, the `start` operation was removed. The `createManagedJob()` operation also allows a notification subscription request to be specified. This is the only way to reliably get all job state notifications.
- The *MJS* `start` operation has been removed. Its purpose was to ensure that the client had received the job EPR prior to the job being executed (and thus consuming resources), and is redundant with the `uuid` functionality.

New GRAM Client Submission Tool:

- *globusrun-ws* has replaced `managed-job-globusrun` as the GRAM4 client submission program. The main reason was performance. The cost of JVM startup for each job submission through `managed-job-globusrun` was too much. `globusrun-ws` is written in C and thus avoids the JVM startup cost. `globusrun-ws` is very similar in functionality to `managed-job-globusrun`, but you will need to become familiar with the arguments and options.

RSL Schema Changes Since GT 3.2:

- RSL Substitutions RSL substitution syntax has changed to allow for a simpler RSL schema that can be parsed by standard tools. In GT 3.2, applications could define arbitrary RSL substitutions within an RSL document and rely on the GRAM service to resolve them. In GT4 GRAM4, this feature is no longer present. In GT 4.2.1 there are 5 RSL variables: `${GLOBUS_USER_HOME}`, `${GLOBUS_USER_NAME}`, `${GLOBUS_SCRATCH_DIR}`, and `${GLOBUS_LOCATION}`.
- `executable` is now a single local file path. Remote URLs are no longer allowed. If executable staging is desired, it should be added to the `fileStageIn` directive.
- `stdin` is now a single local file path. Remote URLs are no longer allowed. If `stdin` staging is desired, it should be added to the `fileStageIn` directive.
- `stdout` is now a single local file path, instead of a list of remote URLs. If `stdout` staging is desired, it should be added to the `fileStageOut` directive.
- `stderr` is now a single local file path, instead of a list of remote URLs. If `stderr` staging is desired, it should be added to the `fileStageOut` directive.
- `scratchDirectory` has been removed.
- `gramMyJobType` has been removed. "Collective" functionality is always available if a job chooses to use it.
- `dryRun` has been removed. This is obsolete given the addition of the `holdState` attribute. setting `holdState` to "StageIn" should prevent the job from being submitted to the local *scheduler*. It can then be canceled once the StageIn-Hold state notification is received.
- `remoteIoUrl` has been removed. This was a hack for GRAM2 involved with staging via GASS, and has no relevancy in the current implementation.

- File Staging related RSL attributes have been replaced with RFT file transfer attributes/syntax.
- RSL substitution definitions and substitution references have been removed in order to be able to use standard XML parsing/serialization tools.
- RSL variables have been added. These are keywords denoted in the form of `${variable name}` that can be found in certain RSL attributes.
- Explicit credential references have been added, which, along with use of the new `DelegationFactory` service, replace the old implicit delegation model.

Fault changes since GT version 3.2:

- `CacheFaultType` was removed since there is no longer a GASS cache.
- `RepeatedlyStartedFaultType` was removed since there is no longer a `start` operation. Repeat creates with the same submission ID simply return the job EPR.
- `SLAFaultType` was changed to `ServiceLevelAgreementFaultType` for clarification.
- `StreamServiceCreationFaultType` was removed since there is no longer a stream service.
- `UnresolvedSubstitutionReferencesFaultType` was removed since there is no longer support for substitution definitions and references in the RSL.
- `DatabaseAccessFaultType` was removed since a database is no longer used to save job data.

Chapter 3. Migrating GRAM from GT2

1. Admin - Migration Guide

1.1. Installation / Deployment Differences

In GRAM2, jobs are submitted to a job manager process started by a Gatekeeper process. The Gatekeeper process is typically started out by an inetd server, which forks a new gatekeeper per job. In GRAM4, jobs are started by the ManagedExecutionJobService, which is a Java service implementation running within the globus service container.

The gatekeeper searches the \$GLOBUS_LOCATION/etc/grid-services directory to determine which services it will start. Typically there is one job manager service entry file in that directory per scheduler type.

1.2. Security Differences

1.2.1. Proxies and Delegation

In GRAM2, the GRAM client is required to delegate a proxy credential to the Gatekeeper so that the job manager can send authenticated job state change messages.

In GRAM4, delegation is done as needed using the DelegationFactoryService. Jobs may be passed references to delegated credentials as part of the job description.

1.2.2. Network Communication

In GRAM2, communication between the client and gatekeeper is done using GSI-wrapped messages. Communication between the the client and job manager are sent using SSL. The job manager uses the delegated credential for file streaming or staging as well. Mutual authentication is done on all connections. All communications consist of a single request-response pattern. Network connections and security contexts are never cached between messages.

In GRAM4, communication may be secured using TLS, ws secure messaging, or ws secure conversation, depending on service configuration. When doing authentication, the service will use the credentials of the container, or for secure message or conversation, a service-specific credential. It will not use a delegated credential when communicating with the client.

1.2.3. Root / Local Account Access

The gatekeeper process is started as a root service out of inetd. It then uses the grid-mapfile decide which local user it should setuid() to before starting the job manager process, based on the credential used to submit the job request. The user may optionally propose a non-default user-id by specifying it in the gatekeeper contact string. The job manager process runs entirely under the local user account.

In GRAM4, the job management service runs within a container shared with other services. The container is run under a non-privileged account. All commands which need to be run as a particular user (such as interactions with the *scheduler*) are started via *sudo*. Authorization is done via the globus-gridmap-and-execute program.

1.3. Scheduler Interaction Differences

In GRAM2, all file system and scheduler interactions occur within a perl module called by the globus-job-manager-script.pl program. Scheduler-specific perl modules implement a number of methods which are used by the job manager:

- submit
- poll
- cancel
- signal
- make_scratchdir
- remove_scratchdir
- stage_in
- stage_out
- cache_cleanup
- remote_io_file_create
- proxy_relocate
- proxy_update

Only a small set of these script methods are used in the GRAM4 implementation. The subset used is:

- submit
- poll (called only once per job and only for fork/condor jobs to merge output)
- cancel
- cache_cleanup

Some of the functionality has been moved into other services for reliability or performance reasons. Other functions have been removed altogether.

- poll: SEG
- signal: dropped
- make_scratchdir: rft
- remove_scratchdir: rft
- stage_in: rft
- stage_out: rft
- remote_io_file_create: rft or resource property queries
- proxy_relocate: delegation service
- proxy_update: delegation service

1.4. Local Node Impact

In GRAM2, each job submitted would cause the following processes to be created:

- gatekeeper (short lived)
- job manager (lives the duration of the job)
- perl script (short lived 4 or more instances depending on job type)
- perl script poll called periodically

In GRAM4, each job causes the following processes to be created

- sudo + perl script--(typically 2 times: submit, cache_cleanup)
- for fork jobs, one fork-starter process (blocked waiting for a signal) for the duration of the job

Additionally, there will be a per-scheduler instance of the *SEG* program, monitoring a log file for job state changes.

2. User - Migration Guide

2.1. Command Line Tools

Typical interactions with the GRAM2 service were done with either the *globusrun* or *command* or the *globus-job* suite of scripts (*globus-job-submit*, *globus-job-run*, *globus-job-get-output*, *globus-job-status*, *globus-job-clean*). The main difference between these sets of commands is that *globusrun* required a *job description* in *RSL* format, and the *globus-job-submit* and *globus-job-run* scripts would construct that based on command line options.

In GRAM4, the *globusrun-ws* command implements the functionality of *globusrun* using the XML Job Description language in place of the RSL format job description of GRAM2. It also allows specifying parts of the Job Description with simple command line arguments (for executable and arguments), similar to what one would do with *globus-job-run*. Like *globusrun*, the *globusrun-ws* program supports both the interactive and batch submission of GRAM jobs.

Table 3.1. Command Line Option Comparison

Description	GRAM2 globusrun option	GRAM4 globusrun-ws option
Interactive Multirequest Control	-i	NO EQUIVALENT
Job Description File Path	-f <rsl filename> -file <rsl filename>	-f <filename> -job-description-file <filename>
Quiet operation	-q -quiet	-q -quiet
File streaming of stdout and stderr *see note 1*	-o (Implies -q)	-s -streaming (Implies -q, sometimes -staging-delegate)
Enable embedded GASS Server	-s -server (Implies -o and -q)	NO EQUIVALENT
Enable writing to the embedded GASS Server	-w -write-allow (Implies -s and -q)	NO EQUIVALENT
Specify Service Contact	-r <resource-manager> -resource <resource-manager> (Specifies Gatekeeper contact)	-F, -Ft, or -Ff; Use either factory service contact (-F), Factory Type (-Ft) or Factory EPR file (-Ff)
Do not terminate job when SIGINT is received.	-n -no-interrupt	-n -no-cleanup
Destroy a job based on a job - contact	-k <job contact> -kill <job contact>	-kill -j <filename> -kill -job-epr-file <filename>
Get current job status	-status <job contact>	-status -j <filename> -status -job-epr-file <filename>
Batch mode job submission	-b -batch or -F -fast-batch	-batch -b
Refresh proxy	-refresh-proxy <job contact> -y <job contact>	NO EQUIVALENT
Stop a job manager process, saving state	-stop-manager <job contact>	NO EQUIVALENT
Validate job description without submitting job	-p -parse	-validate
Ping job manager	-a -authenticate-only	NO EQUIVALENT
Dryrun	-d -dryrun	NO EQUIVALENT

Note 1: In GRAM2, streaming is done using https connections from the job manager to a GASS server embedded in the globusrun program. In GRAM4, streaming is implemented by accessing a gridftp server configured to run along with the service container.

globusrun-ws has additional options to deal with file streaming, monitoring an existing job, authentication and authorization, http timeouts, default termination time, encryption, etc.

3. Developer - API and RSL Migration Guide

This table describes the migration path for applications which use the C language interface to GRAM2. This table covers the globus_gram_client API.

Table 3.2. C API Migration Table

GT2 API Command	GT4 API Command
globus_gram_client_callback_allow()	globus_notification_create_consumer()
globus_gram_client_register_job_request()	ManagedJobFactoryPortType_GetResourceProperty_epr_register()
globus_gram_client_job_request()	ManagedJobFactoryPortType_GetResourceProperty_epr()
globus_gram_client_register_job_cancel()	ManagedExecutableJobPortType_Destroy_epr_register()
globus_gram_client_job_cancel()	ManagedExecutableJobPortType_Destroy_epr()
globus_gram_client_job_status()	ManagedExecutableJobPortType_GetResourceProperty_epr() with the property name {http://www.globus.org/namespaces/2008/03/gram/job/types}state
globus_gram_client_register_job_status()	ManagedExecutableJobPortType_GetResourceProperty_epr_register() with the property name {http://www.globus.org/namespaces/2008/03/gram/job/types}state
globus_gram_client_job_refresh_credentials()	globus_delegation_client_util_delegate_epr
globus_gram_client_register_job_refresh_credentials()	globus_delegation_client_util_delegate_epr_register()
globus_gram_client_register_job_signal()	ManagedExecutableJobPortType_release_epr_register()
globus_gram_client_job_signal()	ManagedExecutableJobPortType_release_epr()
globus_gram_client_register_job_callback_registration()	ManagedExecutableJobPortType_Subscribe_epr_register()
globus_gram_client_job_callback_register()	ManagedExecutableJobPortType_Subscribe_epr()
globus_gram_client_register_job_callback_unregister()	SubscriptionManager_Destroy_epr_register()
globus_gram_client_job_callback_unregister()	SubscriptionManager_Destroy_epr()
globus_gram_client_callback_disallow()	globus_notification_destroy_consumer()
globus_gram_client_job_contact_free()	wsa_EndpointReferenceType_destroy()
globus_gram_client_error_string()	globus_error_get(result)
globus_gram_client_set_credentials()	globus_soap_message_handle_set_attr() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY and the value the gss_cred_id_t
globus_gram_client_ping()	XXX? Maybe factory get resource properties?
globus_gram_client_register_ping()	XXX? Maybe factory get resource properties?
globus_gram_client_debug()	set GLOBUS_SOAP_MESSAGE_DEBUG environment variable to MESSAGES to see XML messages sent/received
globus_gram_client_version()	NO EQUIVALENT
globus_gram_client_attr_init()	globus_soap_message_attr_init()
globus_gram_client_attr_destroy()	globus_soap_message_attr_destroy()
globus_gram_client_attr_set_credential()	globus_soap_message_handle_set_attr() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY and the value the gss_cred_id_t

GT2 API Command	GT4 API Command
globus_gram_client_attr_get_credential()	globus_soap_message_attr_get() with the property name GLOBUS_SOAP_MESSAGE_USER_CREDENTIAL_KEY. Migration from GRAM2 to GRAM4

GRAM2 uses a custom language for specifying a job description. GRAM4 uses an xml based language for this same purpose. In GRAM2, relations (such as count=5) can occur in any order within the RSL; in GRAM4, the relations must be in the order in the XML schema definition. The RSL attribute description below is in the order defined by the XML schema

Table 3.3. RSL Migration Table

GT2 RSL Attribute	GT4 job description element
(username = NAME)	<localUserId>NAME</localUserId>
(two_phase = TWO_PHASE_TIMEOUT) *See Note 1*	<holdState>Pending</holdState>
(executable = EXE)	<executable>EXE</executable>
(directory = DIR)	<directory>DIR</directory>
(arguments=ARG1 ... ARGN)	<argument>ARG1</argument> ... <argument>ARGN</argument>
(environment = (ENV_VAR_1 ENV_VAL_1) ... (ENV_VAR_N ENV_VAL_N))	<environment> <name>ENV_VAR_1</name> <value>ENV_VAL_1</value> ... <name>ENV_VAR_N</name> <value>ENV_VAL_N</value> </environment>
(stdin = LOCAL_FILE_PATH) *See Note 2*	<stdin>file:///LOCAL_FILE_PATH</stdin>
(stdout = LOCAL_FILE_PATH) *See Note 2*	<stdout>file:///LOCAL_FILE_PATH</stdout>
(stderr = LOCAL_FILE_PATH) *See Note 2*	<stderr>file:///LOCAL_FILE_PATH</stderr>
(count = NUMBER)	<count>NUMBER</count>
(library_path = PATH_ELEMENT_1 ... PATH_ELEMENT_N)	<libraryPath>PATH_ELEMENT_1</libraryPath> ... <libraryPath>PATH_ELEMENT_N</libraryPath>
(host_count = NUMBER)	<hostCount>NUMBER</hostCount>
(project = PROJECT)	<project>PROJECT</project>
(queue = QUEUE)	<queue>QUEUE</queue>
(max_time = MINUTES)	<maxTime>MINUTES</maxTime>
(max_wall_time = MINUTES)	<maxWallTime>MINUTES</maxWallTime>
(max_cpu_time = MINUTES)	<maxCpuTime>MINUTES</maxCpuTime>
(max_memory = MEGABYTES)	<maxMemory>MEGABYTES</maxMemory>
(min_memory = MEGABYTES)	<minMemory>MEGABYTES</minMemory>
(job_type = JOBTYP)	<jobType>JOBTYP</jobType>
(file_stage_in = (REMOTE_GRIDFTP_URL_1 LOCAL_FILE_PATH_1) ... (REMOTE_GRIDFTP_URL_N LOCAL_FILE_PATH_N)) *See Note 4*	<fileStageIn> <transfer> <sourceUrl>REMOTE_GRIDFTP_URL_1</sourceUrl> <destinationUrl>file:///LOCAL_FILE_PATH_1</destinationUrl> </transfer> <transfer> <sourceUrl>REMOTE_GRIDFTP_URL_N</sourceUrl> <destinationUrl>file:///LOCAL_FILE_PATH_N</destinationUrl> </transfer> </fileStageIn>
(file_stage_out = (LOCAL_FILE_PATH_1 REMOTE_GRIDFTP_URL_1) ... (LOCAL_FILE_PATH_N REMOTE_GRIDFTP_URL_N)) *See Note 4*	<fileStageOut> <transfer> <sourceUrl>file:///LOCAL_FILE_PATH_1</sourceUrl> <destinationUrl>REMOTE_GRIDFTP_URL_1</destinationUrl> </transfer> <transfer> <sourceUrl>file:///LOCAL_FILE_PATH_N</sourceUrl> <destinationUrl>REMOTE_GRIDFTP_URL_N</destinationUrl> </transfer> </fileStageOut>

Note 1: The globusrun-ws program will automatically release the hold after receiving the indicated hold state. To simulate the two-phase submit timeout, an application could set the initial termination time of the resource. A hold

state may be set for fileCleanUp state for two-phase commit end, but it is not possible to submit a job with both hold states.

Note 2: stdin, stdout, and stderr must only be a local file URL. Ftp and gridftp URLs can be handled by using a fileStageIn and fileStageOut elements (described below).

Note 3: Value job types for GRAM4 are multiple (the default), single, mpi, and condor.

Note 4: The GRAM4 service uses RFT to transfer files. This only supports gridftp and ftp file transfers. The local file path must be a mappable by an entry in the file system mapping file.

The following RSL attributes have no direct equivalent in GRAM4:

- `dry_run`: Similar behavior can be obtained by using a job hold state of Pending and then destroying the job resource without releasing the hold.
- `file_stage_in_shared`: No support for the GASS cache, hence this is gone. Applications may use RFT to transfer files before submitting a batch of jobs.
- `gass_cache`: GASS cache is not used by GRAM4, so there is no need for setting the cache path.
- `gram_my_job`: collective operations are enabled for every managed execution job service via rendezvous registration
- `proxy_timeout`: Delegated security proxies are handled via the DelegationFactory Service. Resource lifetime is controlled by the `wsrl:SetTerminationTime` operation
- `remote_io_url`: The GRAM4 service does not use GASS, so there is no equivalent to this.
- `restart`: There is no equivalent.
- `rsl_substitution`: The GRAM4 service does not support user-defined substitutions. Certain values may be referenced in some RSL values by a similar technique, but these are for system configuration parameters only. See the GRAM4 job description document for description of RSL variable syntax, values, and attributes where they may be used.
- `save_state`: All GRAM4 jobs are persistent, so there is no elements related to this.
- `scratch_dir`: This is now a deployment configuration option.
- `stderr_position`: Standard error streaming is now a feature of the globusrun-ws program instead of part of the GRAM4 service, so there is no equivalent element for restarting error streaming at a specific point.
- `stdout_position`: Standard output streaming is now a feature of the globusrun-ws program instead of part of the GRAM4 service, so there is no equivalent element for restarting output streaming at a specific point.

Here are some examples of converting some GRAM2 RSLs to GRAM4.

Table 3.4. RSL Migration Examples

GRAM2 RSL	GRAM4 Job Description
<pre>(* Simple Job Request With Arguments *) &(executable = /bin/echo) (arguments = Hello, Grid)</pre>	<pre><?xml version="1.0"?> <!-- Simple Job Request With Ar <job xmlns:ns1="http://www.glob <ns1:executable>/bin/echo</ns <ns1:argument>Hello,</ns1:arg <ns1:argument>Grid</ns1:argum </job></pre>

GRAM2 RSL	GRAM4 Job Description
<pre>(* Multijob Request *) +(&(executable = /bin/echo) (arguments = Hello, Grid From Subjob 1) (resource_manager_name = resource-manager-1.globus.org) (count = 1)) (&(executable = mpi-hello) (arguments = Hello, Grid From Subjob 2) (resource_manager_name = resource-manager-2.globus.org) (count = 2) (jobtype = mpi))</pre>	

GRAM2 RSL	GRAM4 Job Description
	<pre> <?xml version="1.0" <!-- Multijob Reque <multiJob <!-- namespace xmlns:gram="htt <!-- namespace xmlns:wsa="http <factoryEndpo <wsa:Address> <!-- URL fo https://res </wsa:Address <!-- Referenc <wsa:Referenc <!-- ID for <gram:Resou </wsa:Referen </factoryEndpoi <job> <factoryEndpo <wsa:Address <!-- UR https:/ </wsa:Addre <!-- Refere <wsa:Refere <!-- ID f <gram:Res </wsa:Refer </factoryEndp <executable>/ <argument>Hel <argument>Gri <argument>Fro <argument>Sub <argument>1</ <count>1</cou </job> <job> <factoryEndpo <wsa:Address <!-- UR https:/ </wsa:Addre <!-- Refere <wsa:Refere <!-- ID f <gram:Res </wsa:Refer </factoryEndp </pre>

GRAM2 RSL	GRAM4 Job Description
	<pre data-bbox="1321 243 1624 554"><executable>m <argument>Hel <argument>Gri <argument>Fro <argument>Sub <argument>2</ <count>2</cou <jobType>mpi< </job> </multiJob></pre>

Glossary

G

globusrun-ws A command line program used to submit jobs to a GRAM4 service. See the the GRAM4 Commandline page.

J

job description Term used to describe a GRAM4 job for GT4.

M

Managed Job Factory Service (MJFS) [FIXME]

R

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

S

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

Scheduler Event Generator (SEG) The Scheduler Event Generator (SEG) is a program which uses scheduler-specific monitoring modules to generate job state change events. Depending on scheduler-specific requirements, the SEG may need to run with privileges to enable it to obtain scheduler event notifications. As such, one SEG runs per scheduler resource. For example, on a host which provides access to both PBS and fork jobs, two SEGs, running at (potentially) different privilege levels will be running. One SEG instance exists for any particular scheduled resource instance (one for all homogeneous PBS queues, one for all fork jobs, etc). The SEG is implemented in an executable called the globus-scheduler-event-generator, located in the Globus Toolkit's libexec directory.

superuser do (sudo) Allows a system administrator to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while logging the commands and arguments. See <http://www.courtesan.com/sudo/> for more information.

U

Universally Unique Identifier (UUID) Identifier that is immutable and unique across time and space.