

# **GT 5.0.0 Migrating Guide for GRAM5**

---

## **GT 5.0.0 Migrating Guide for GRAM5**

### **Abstract**

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

---

---

# Table of Contents

1. Migrating GRAM from GT4.2 .....	1
1. Admin - Migration Guide .....	1
2. User - Migration Guide .....	1
3. Developer - Migration Guide .....	1
2. Migrating GRAM from GT4.0 .....	2
1. Admin - Migration Guide .....	2
2. User - Migration Guide .....	2
3. Developer - Migration Guide .....	2
3. Migrating GRAM from GT3 .....	3
4. Migrating GRAM from GT2 .....	5
1. Admin - Migration Guide .....	5
2. User - Migration Guide .....	6
3. Developer - API and RSL Migration Guide .....	7
Glossary .....	8

---

# Chapter 1. Migrating GRAM from GT4.2

The GRAM5 protocol has been designed to be backward compatible with GRAM2 protocol from GT 4.2.x. There is no compatibility between GRAM5 and the GRAM4 protocol.

## 1. Admin - Migration Guide

### 1.1. Audit Logging

GRAM5 supports generating audit records the same as GRAM2. It also adds support for a Teragrid-specific Gateway User field in the audit records. The `globus_gram_job_manager_auditing` package contains the audit database interface code. The `globus_gram_job_manager_auditing_setup` setup package configures this package. GRAM5 auditing is enabled by using the `-audit-directory` command-line option in the job manager configuration file. For more information about GRAM5 audit support, see the [GRAM5 admin guide](#).

## 2. User - Migration Guide

### 2.1. Command-line Tools

GRAM5 provides the `globusrun` program to submit jobs to GRAM5 services. It no longer supports multi-request (duroc or MPI) jobs. GRAM5 does not provide the `globusrun-ws` as it does not support the WSRF protocols.

## 3. Developer - Migration Guide

### 3.1. API Changes

The GRAM5 version of the GRAM Client API adds support for receiving protocol extension information in callbacks and responses. All GRAM Client API functions from GRAM2 are provided in the GRAM5 API. The DUROC, DUCT, and Nexus APIs are no longer provided in GRAM5.

---

# Chapter 2. Migrating GRAM from GT4.0

The GRAM5 protocol has been designed to be backward compatible with GRAM2 protocol from GT 4.0.x. There is no compatibility between GRAM5 and the GRAM4 protocol.

## 1. Admin - Migration Guide

### 1.1. Audit Logging

GRAM5 supports generating audit records the same as GRAM2. It also adds support for a Teragrid-specific Gateway User field in the audit records. The `globus_gram_job_manager_auditing` package contains the audit database interface code. The `globus_gram_job_manager_auditing_setup` setup package configures this package. GRAM5 auditing is enabled by using the `-audit-directory` command-line option in the job manager configuration file. For more information about GRAM5 audit support, see the [GRAM5 admin guide](#).

## 2. User - Migration Guide

### 2.1. Command-line Tools

GRAM5 provides the `globusrun` program to submit jobs to GRAM5 services. It no longer supports multi-request (duroc or MPI) jobs. GRAM5 does not provide the `globusrun-ws` as it does not support the WSRF protocols.

## 3. Developer - Migration Guide

### 3.1. API Changes

The GRAM5 version of the GRAM Client API adds support for receiving protocol extension information in callbacks and responses. All GRAM Client API functions from GRAM2 are provided in the GRAM5 API. The DUROC, DUCT, and Nexus APIs are no longer provided in GRAM5.

---

# Chapter 3. Migrating GRAM from GT3

Migrating to GT 5.0.0 from GT version 3.2:

- The 5.0.0 protocol has been changed to be WSRF compliant. There is no backward compatibility between 5.0.0 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 GRAM5 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 GT5 GRAM5, this feature is no longer present. In GT 5.0.0 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 4. 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. In GRAM5, the job submit protocol is the same; however, all jobs for a particular user and LRM run in the same job manager process.

### 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. Because each job is monitored by a separate job manager in GRAM2, each job manager has access to a different delegated credential. In GRAM5, the shared job manager uses the delegated credential with the latest expiration time as its credential.

In GRAM2, the client can control the job manager proxy timeout by setting the value of `proxy_timeout` RSL attribute to a time interval in seconds indicating when the job manager will exit if the proxy is about to expire. In GRAM5, this is a job manager-wide setting and the `proxy_timeout` RSL attribute is ignored.

### 1.3. Network Communication

In GRAM2, the standard output and standard error streams of a job may be sent in near real time to a file server such as the GASS server embedded in a **globusrun** execution. In GRAM5 the same RSL syntax is used to name output and error stream destinations, but those are not sent until after the job execution is complete.

In GRAM2, the job manager implements intra-job communication via DUCT and task synchronization via DUROC. These features have been dropped in GRAM5.

GRAM5 adds various protocol extensions to the GRAM2 protocol. This is done in a way such that the existing GRAM2 protocol processors will ignore the extensions to the messages. Details about the protocol and its extensions can be found in the [GRAM5 public interface document](#).

### 1.4. LRM Interaction Differences

In GRAM2, all file system and LRM 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 GRAM5 implementation. The subset used is:

- submit
- signal (called when not using SEG)
- poll (called when not using SEG)
- cancel

Some of the functionality has been moved into the job manager or other services for performance reasons.

## 1.5. 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 GRAM5, each job causes the following processes to be created

- **globus-gatekeeper** (short lived)
- **globus-job-manager** (short lived for all but one concurrent instance)
- **globus-job-manager-script** (up to 5 per lifetime of the job manager)
- **globus-fork-starter** (up to 5 per job manager when using the fork LRM and the SEG).

Additionally, there will be a per-scheduler instance of the *SEG*-related program, **globus-job-manager-event-generator**.

## 2. User - Migration Guide

### 2.1. Command Line Tools

The **globusrun** tool in GRAM2 supports DUROC and MPICH-G jobs. This feature has been removed in GRAM5, as well as the command-line options related to it.

## **3. Developer - API and RSL Migration Guide**

The DUROC and DUCT APIs have been removed in GRAM5.

---

# Glossary

## R

Resource Specification Language (RSL)

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

## 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.

## U

Universally Unique Identifier (UUID)

Identifier that is immutable and unique across time and space.