GT 4.0 WS_GRAM

GT 4.0 WS_GRAM

Table of Contents

List of Tables

Chapter 1. Execution Management: Key Concepts

1. Overview

The Globus Toolkit provides both a suite of web services and a "pre-web services" Unix server suite to submit, monitor, and cancel jobs on Grid computing resources. Both systems are known under the moniker "GRAM", while "WS GRAM" refers only to the web service implementation. Jobs are computational tasks that may perform input/output operations while running which affect the state of the computational resource and its associated file systems. In practice, such jobs may require coordinated staging of data into the resource prior to job execution and out of the resource following execution. Some users, particularly interactive ones, benefit from accessing output data files as the job is running. Monitoring consists of querying and subscribing for status information such as job state changes.

Grid computing resources are typically operated under the control of a scheduler which implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance. GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format.

For more detailed information about the concepts behind the software implementation, see WS GRAM Approach[1].

2. Conceptual details

A number of concepts underly the purpose and motivation for GRAM. These concepts are divided into broad categories below.

2.1. Targeted job types

GRAM is meant to address a range of jobs where arbitrary programs, reliable operation, stateful monitoring, credential management, and file staging are important. GRAM is not meant to serve as a "remote procedure call" (RPC) interface for applications not requiring many of these features, and furthermore its interface model and implementation may be too costly for such uses. The WS GRAM service will become cheaper over time as the underlying web service technologies improve, but as with the older pre-web service GRAM, its protocols will always involve multiple round-trips to support these advanced features that are not required for simple RPC applications.

The underlying WSRF core environment used for WS GRAM should also be considered as a direct application-hosting solution for lightweight, shared applications. In other words, if an application requires only modest input and output data transport in a stateless manner (all that matters is the result data or fault signal) and will be invoked many times, it may be a good candidate for exposure as an application-specific Web service.

2.2. Component architecture

Rather than consisting of a monolithic solution, GRAM is based on a component architecture at both the protocol and software implementation levels. This component approach serves as an ideal which shapes the implementation as well as the abstract design and features.

[1] WS_GRAM_Approach.html

GRAM service suite	Job management with GRAM makes use of multiple types of service: • Job management services represent, monitor, and control the overall job life cycle. These services are the job-management specific software provided by the GRAM solution. • File transfer services support staging of files into and out of compute resources. GRAM makes use of these existing services rather than providing redundant solutions; WS GRAM has further refactored some file transfer mechanisms present in pre-web service GRAM. • Credential management services are used to control the delegation of rights among distributed elements of the GRAM architecture based on users' application requirements. Again, GRAM makes use of more general infrastructure rather than providing a redundant solution, and WS GRAM has continued this refactoring to better separate credential management at the protocol level.
Service model	For WS GRAM, the Globus Toolkit software development environment, and particularly WSRF core, is used to implement distributed communications and service state. For pre-web service GRAM, the "gatekeeper" daemon and GSI library are used for communications and service dispatch.
Scheduler adapters	A scripted plug-in architecture is provided by GRAM to enable extension with adapters to control a variety of local schedulers.

2.3. Security

Secure operation	WS GRAM utilizes WSRF functionality to provide for authentication of job management requests as well as to protect job requests from malicious interference, while pre-web service GRAM uses GSI and secure sockets directly. The use of GRAM does not reduce the ability for system administrators to control access to their computing resources. The use of GRAM also does not reduce the safety of jobs users run on a given computing resource.
Local system protection domains	To protect users from each other, jobs are executed in appropriate local security contexts, e.g. under specific Unix user IDs based on details of the job request and authorization policies. Additionally, GRAM mechanisms used to interact with the local resource are design to minimize the privileges required and to minimize the risks of service malfunction or compromise.
Credential delegation and management	A client may delegate some of its rights to GRAM services in order to facilitate the above functions, e.g. rights for GRAM to access data on a remote storage element as part of the job execution. Additionally, the client may delegate rights for use by the job process itself. With pre-web service GRAM, these two uses of rights are inseparable, while WS GRAM provides separate control for each purpose (while still allowing rights to be shared if that is desired).
Audit	To assist with normal accounting functions as well as to further mitigate risks from abuse or malfunction, GRAM uses a range of audit and logging techniques to record a history of job submissions and critical system operations.

2.4. Job Management

Reliable job submission	WS GRAM provides an "at most once" job submission semantics. A client is able to check for and possibly resubmit jobs, in order to account for transient communication errors without risk of running more than one copy of the job. Similarly, pre-web service GRAM provides a two-phase submission mechanism to submit and then commit a job to be run.
Job cancellation	While many jobs are allowed to run to their natural completion, GRAM provides a mechanism for clients to cancel (abort) their jobs at any point in the job life cycle.

2.5. Data Management

Reliable data staging	WS GRAM provides for reliable, high-performance transfers of files between the compute resource and external (gridftp) data storage elements before and after the job execution. Pre-web service GRAM can also stage with gridftp systems but with less flexible reliable-transfer logic driving its requests.
Output monitoring	GRAM supports a mechanism for incrementally transferring output file contents from the computation resource while the job is running. WS GRAM uses a new mechanism to allow arbitrary numbers of files to be transferred in this fashion, while pre-web service GRAM only supports incremental transfer of the job's standard output and error streams.

2.6. Task coordination

Parallel jobs	Some jobs are parallel, meaning that they consist of more than one simultaneous tasks. These tasks are often hosted on parallel computing hardware to provide increased computational throughput.
Task rendezvous	Some parallel programming environments, such as MPI, provide mechanisms for all tasks in a parallel computation to find out about each other: to know the number of peer tasks as well as possibly to exchange some information between tasks. Native parallel programming models often support this within the local job start mechanism. GRAM provides a mechanism for task rendezvous which job applications may use if they do not have another more appropriate solution. This mechanism may be used directly by application code or by intervening middleware libraries, e.g. libraries that are designed to present a simplified programming model to applications run via GRAM. GRAM does not require tasks to be coordinated, but addresses this requirement in order to facilitate a wider range of applications. The protocols and APIs used to support task rendezvous are different for WS GRAM and pre-web service GRAM.

3. Related documents

The following links include internal or external documents that expand on some of these key concepts:

- Pre WS GRAM Approach[2]
- WS GRAM Approach[3]

[2] ../prewsgram/Pre_WS_GRAM_Approach.html
[3] ../wsgram/WS_GRAM_Approach.html

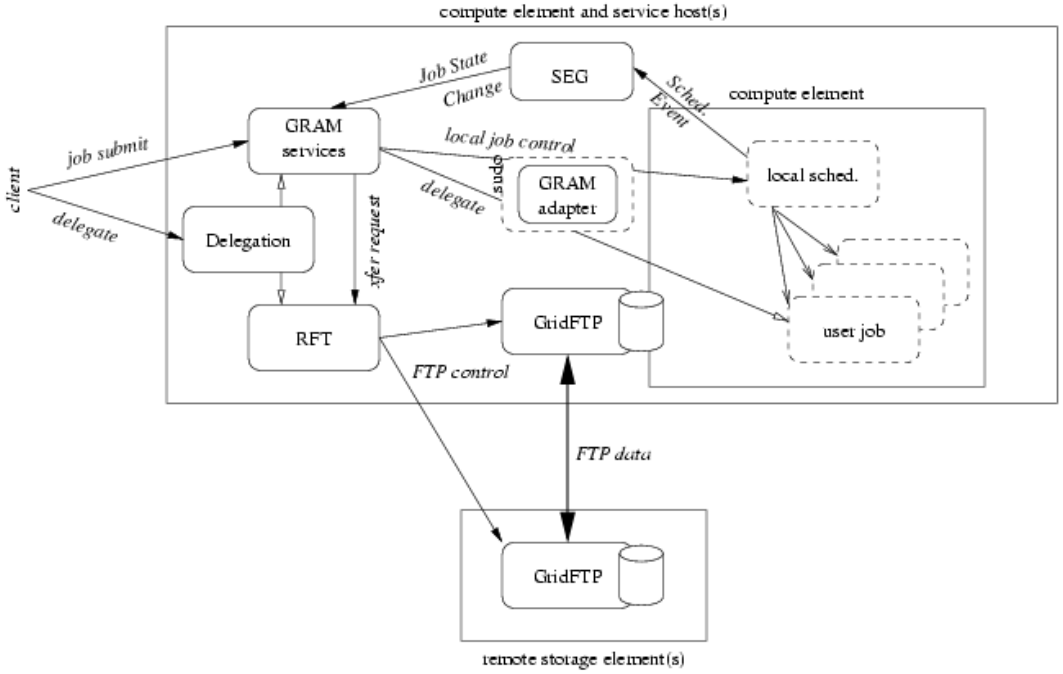
Chapter 2. GT 4.0 WS GRAM Approach

1. Introduction

The WS GRAM software implements a solution to the job-management problem described in the GT 4.0 WS GRAM Key Concepts[1] document, providing Web services interfaces consistent with the WSRF model. This solution is specific to operating systems following the Unix programming and security model.

2. Component architecture approach

WS GRAM combines job-management services and local system adapters with other service components of GT 4.0 in order to support job execution with coordinated file staging.



2.1. WS GRAM

The heart of the WS GRAM service architecture is a set of Web services designed to be hosted in the Globus Toolkit's WSRF core hosting environment. Note, these services, described below, make use of platform-specific callouts to other software components described in the next section.

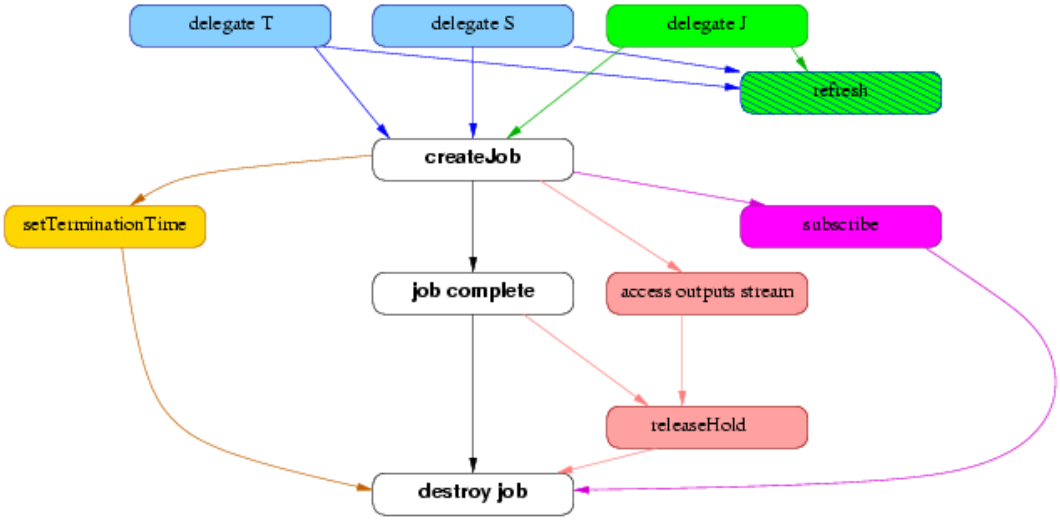
[1] index.html

Table 2.1. WS GRAM Webservices

ManagedJob	Each submitted job is exposed as a distinct resource qualifying the generic ManagedJob service. The service provides an interface to monitor the status of the job or to terminate the job (by terminating the ManagedJob resource). The behavior of the service, i.e. the local scheduler adapter implementation, is selected by the specialized *type* of the resource.
ManagedJobFactory	Each compute element, as accessed through a local scheduler, is exposed as a distinct resource qualifying the generic ManagedJobFactory service. The service provides an interface to create ManagedJob resources of the appropriate type in order to perform a job in that local scheduler.

2.2. Major Protocol Steps

The components in the WS GRAM solution are organized to support a range of optional features that together address different usage scenarios. These scenarios are explored in depth in terms of protocol exchanges in the Protocol Variations section. However, at a high level we can consider the main client activities around a WS GRAM job to be a partially ordered sequence.



2.3. Creation of Job

The main component of the WS GRAM service model is the ManagedJob resource created by a ManagedJobFactory::createManagedJob invocation. A meaningful WS GRAM client MUST create a job that will then go through a life cycle where it eventually completes execution and the resource is eventually destroyed (the core black-and-white nodes in the high-level picture).

2.4. Optional Staging Credentials

Optionally, the client MAY request staging activities to occur before or after the job. If these are requested in the create call, suitable delegated credential EPRs MUST be passed in as part of the creation input, meaning that delegation operations MUST be performed sometime before createManagedJob when staging is enabled (the light-blue delegation nodes in the high-level picture). Two credential fields must be initialized: the staging and transfer credentials, which may refer to distinct credentials or may both refer to the same credential. The staging credential gives WS GRAM the right to interact with the RFT service, while the transfer credential gives RFT the right to interact with GridFTP servers.

2.5. Optional Job Credential

Optionally, the client MAY request that a credential be stored into the user account for use by the job process. When this is requested in the create call, a suitable delegated credential EPR is passed as part of the creation input. As for staging, the credential MUST have been delegated before the job is created (the green nodes in the picture).

2.6. Optional Credential Refresh

Optionally, credentials delegated for use with staging, transfer, or job processes may be refreshed using the Delegation service interface. This operation may be performed on any valid Delegation EPR (the blue/green striped node in the picture).

2.7. Optional Hold of Cleanup for Streaming Output

If the client wishes to directly access output files written by the job (as opposed to waiting for the stage-out step to transfer files from the job host), the client should request that the file cleanup process be held until released. This gives the client an opportunity to fetch all remaining/buffered data after the job completes but before the output files are deleted. (See the pink nodes in the high-level picture).

The cleanup hold and release are not necessary if the client will not be accessing files that are scheduled for cleanup in the job request, either because the client is not accessing any files or because the files it is accessing will remain on the job host after ManagedJob termination.

2.8. ManagedJob Destruction

Under nearly all circumstances, ManagedJob resources will be eventually destroyed after job cleanup has completed. Clients may hasten this step via an explicit destroy request or by manipulation of the scheduled termination time. Most system administrators will set a default and maximum ManagedJob linger time after which automatic purging of completed ManagedJob resources will occur.

2.9. Globus Toolkit Components used by WS GRAM

Table 2.2. Globus Toolkit Components used by WS GRAM

Reliable-FileTrans-fer	The ReliableFileTransfer (RFT) service of GT 4.0 is invoked by the WS GRAM services to effect file staging before and after job computations. The integration with RFT provides a much more robust file staging manager than the ad-hoc solution present in previous versions of the GRAM job manager logic. RFT has better support for retry, restart, and fine-grained control of these capabilities. WS GRAM exposes the full flexibility of the RFT request language in the job staging clauses of the job submission language.
GridFTP	GridFTP servers are required to access remote storage elements as well as file systems accessible to the local compute elements that may host the job. The ReliableFileTransfer Web service acts as a so-called third-party client to the GridFTP servers in order to manage transfers of data between remote storage elements and the compute element file systems. It is not necessary that GridFTP be deployed on the same host/node as the WS GRAM services, but staging will only be possible to the subset of file systems that are shared by the GridFTP server that is registered with the WS GRAM service. (REF TO DEPLOY/CONFIG HERE) If no such server or shared file systems are registered, staging is disallowed to that WS GRAM compute element. GridFTP is also used to monitor the contents of files written by the job during job execution. The standard GridFTP protocol is used by a slightly unusual client to efficiently and reliably check the status of files and incrementally fetch new content as the file grows. This method supports "streaming" of file content from any file accessible by GridFTP, rather than only the standard output and error files named in the job request--the limitation of previous GRAM solutions. This approach also simplifies failover and restart of streaming to multiple clients. The integration with GridFTP replaces the legacy GASS (Globus Access to Secondary Storage) data transfer protocol. This changeover is advantageous both for greater performance and reliability of data staging as well as to remove redundant software from the GRAM codebase.
Delega-tion	The Delegation service of GT 4.0 is used by clients to delegate credentials into the correct hosting environment for use by WS GRAM or RFT services. The integration of the Delegation service replaces the implicit, binding-level delegation of previous GRAM solutions with explicit service operations. This change not only reduces the requirements on client tooling for interoperability purposes, but also allows a new separation of the life cycle of jobs and delegated credentials. Credentials can now be shared between multiple short-lived jobs or eliminated entirely on an application-by-application basis to make desired performance and security tradeoffs. Meanwhile, for unique situations WS GRAM retains the ability to refresh credentials for long-lived jobs and gains an ability to designate different delegated credentials for different parts of the job's life cycle.

2.10. External Components used by WS GRAM

Table 2.3. External Components used by WS GRAM

Local job scheduler	An optional local job scheduler is required in order to manage the resources of the compute element. WS GRAM has the ability to spawn simple time-sharing jobs using standard Unix `fork()` methods, but most large-scale compute elements will be under the control of a schedule such as PBS, LSF, Loadleveler, etc.
sudo	The de facto standard Unix `sudo` utility is used by WS GRAM to gain access to target user accounts without requiring WS GRAM to have general super-user privilege on the system. The sudo command is used to execute WS GRAM adapter tools in the user account context; these adapters perform the local system-specific operations needed to initialize and run user jobs. The `sudo` utility not only provides a simple way for WS GRAM to run programs as other users without "root" privilege, but it provides fine-grained controls for the system administrator to restrict which user accounts are valid WS GRAM targets as well as secure auditing of all operations requested by WS GRAM. This mechanism replaces the root-privileged Gatekeeper component of the Pre-WS GRAM service in order to avoid running an entire WSRF hosting environment as root. This change provides enhanced security, at the expense of slightly more complicated deployment effort, and is motivated by the relative increase in the size of the WS GRAM and WSRF codebase as compared to the traditional Gatekeeper.

2.11. Internal Components used by WS GRAM

Table 2.4. Internal Components used by WS GRAM

Scheduler Event Generator[2]	The Scheduler Event Generator component program provides the job monitoring capability for the WS-GRAM service. Plugin modules provide an interface between the Scheduler Event Generator and local schedulers.
Fork Starter[3]	The Fork Starter program starts and monitors job processes for WS-GRAM services which do not a local scheduler. The starter executes the user application and then waits for it to terminate. It records the start time, termination time, and exit status of each process it starts. This information is used by a Scheduler Event Generator plugin for triggering job state changes.

3. Security model

3.1. Secure operations

WS GRAM utilizes secure Web service invocation, as provided by the WSRF core of the Globus Toolkit, for all job-management and file-management messages. This security provides for authentication of clients, tamper-resistant messaging, and optional privacy of message content.

3.2. Local system protection domains

User jobs are executed within Unix user accounts. WS GRAM authentication mechanisms allow the administrator to control to which local accounts a Grid-based client may submit jobs. WS GRAM uses the Unix `sudo` command to

[2] ../wsgram/developer/internal-components.html#seg
[3] ../wsgram/developer/internal-components.html#forkstarter

access user accounts after determining that the client has the right to access the account. Additionally, the administrator may use access and allocation policy controls in the local scheduler to further restrict the access of specific clients and Unix users to local computing resources.

3.3. Credential delegation and management

A client may optionally delegate some of its rights to WS GRAM and related services in order to facilitate file staging. Additionally, the client may delegate rights for use by the job process itself. If no delegation is performed, staging is disallowed and the job will have no ability to request privileged Grid operations.

3.4. Audit

WS GRAM provides three types of logging or auditing support:

Table 2.5. Audit Logging Support

WSRF core message logging	Detailed logging of the underlying client messages may be logged if such logging is enabled in the container configuration. See WS Core debugging doc[4]
WS GRAM custom logging	WS GRAM generates domain-specific logging information about job requests and exceptional conditions. See WS GRAM debugging doc[5]
WS GRAM job auditing direct to DB	WS GRAM can be configured to write a job audit record directly to a Database. This can be useful for exposing and integrating GRAM job information with a Grid's existing accounting infrastructure. A case study for TeraGrid can be read here[6]
Local scheduler logging	For systems using a local batch scheduler, all of the accounting and logging facilities of that scheduler remain available for the administrator to track jobs whether submitted through WS GRAM or directly to the scheduler by local users.
Local system logging	The use of `sudo` for all operations against target user accounts allows the administrator to log the low-level system operations requested by WS GRAM using `sudo`'s native auditing support.

3.5. Protocol Overview

As depicted above, the WS GRAM protocol is centered around the creation of a stateful ManagedJob resource using the ManagedJobFactory createManagedJob() operation. A simple batch job may involve nothing more than this initial client creation step, with all other job life cycle steps occurring automatically in the server. A number of optional protocol elements are available for more complex scenarios.

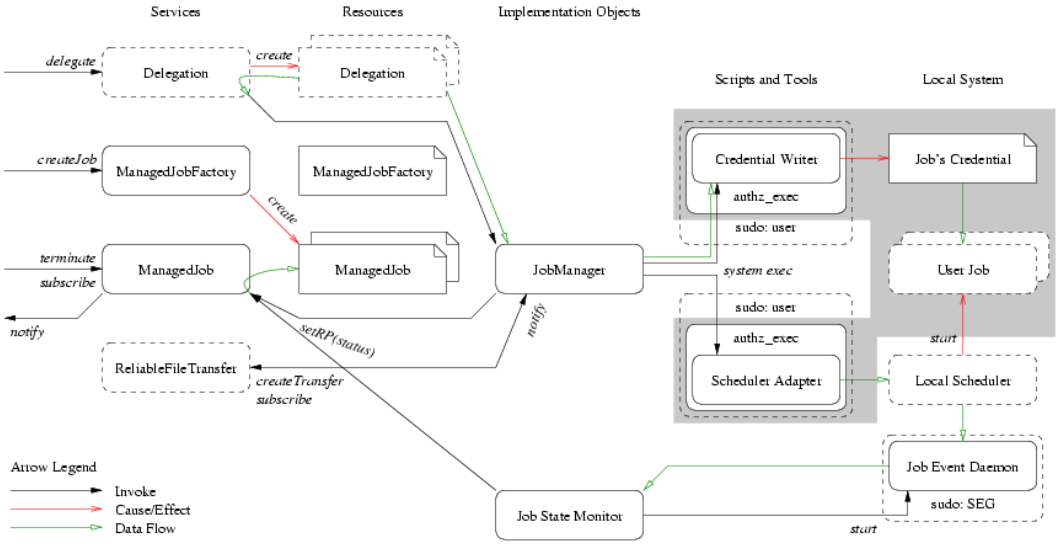
[4] ../../common/javawscore/developer-index.html#debugging
[5] ../wsgram/developer-index.html#debugging
[6] http://www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit

Table 2.6. Protocol Overview

DelegationFactory::requestSecurityToken	This (optional) step allows a client to delegate credentials that will be required for correct operation of WS GRAM, RFT, or the user's job process. Such credentials are only used when referenced in the subsequent job request and under the condition that WS GRAM or RFT is configured to make use of the DelegationFactory, respectively.
Delegation::refresh	This (optional) step allows a client to update the credentials already established for use with the previous requestSecurityToken step.
ManagedJobFactory::getResourceProperty and getMultipleResourceProperties	These (optional) steps allow a client to retrieve information about the scheduler and the jobs associated with a particular factory resource before or after job creation. The delegationFactoryEndpoint and stagingDelegationFactoryEndpoint resource properties are two examples of information that may need to be obtained before job creation.
ManagedJobFactory::createManagedJob	This required step establishes the stateful ManagedJob resource which implements the job processing described in the input request.
ManagedJob::release	This (optional) step allows the ManagedJob to continue through a state in its life cycle where it was previously held or scheduled to be held according to details of the original job request.
ManagedJob::setTerminationTime	This (optional) step allows the client to reschedule automatic termination to be different than was originally set during creation of the ManagedJob resource.
ManagedJob::destroy	This (optional) step allows the client to explicitly abort a job and destroy the ManagedJob resource in the event that the scheduled automatic termination time is not adequate. If the job has already completed (i.e. is in the Done or Failed state), this will simply destroy the resource associated with the job. If the job has not completed, appropriate steps will be taken to purge the job process from the scheduler and perform clean up operations before setting the job state to Failed.
ManagedJob::subscribe	This (optional) step allows a client to subscribe for notifications of status (and particularly life cycle status) of the ManagedJob resource. For responsiveness, it is possible to establish an initial subscription in the createManagedJob() operation without an additional round-trip communication to the newly created job.
ManagedJob::getResourceProperty and getMultipleResourceProperties	These (optional) steps allow a client to query the status (and particularly life cycle status) of the ManagedJob resource.

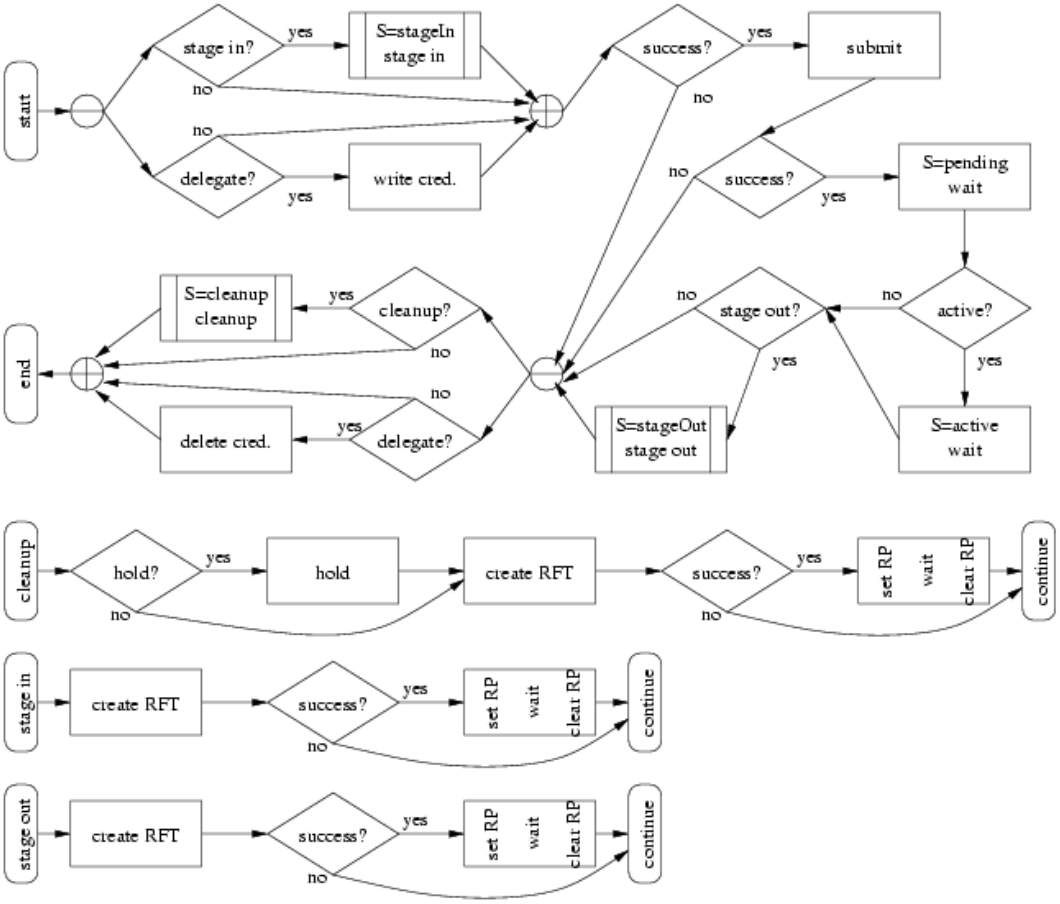
4. WS GRAM software architecture

4.1. Overview



4.2. ManagedJob Resource Life Cycle Logic

The ManagedJob resource has a complex life cycle. The generic behavior is depicted in the following flowchart as a partially ordered sequence of processes and decision points. The status of the ManagedJob resource, including its "job state" is set as a side-effect of this control flow. The processes in the flowchart do not all directly correspond to client-visible job states.



4.3. Software for local system interaction

Table 2.7. Software for local system interaction

Scheduler adapters	Support to control each local scheduler is provided in the form of adapter scripts in the Perl programming language, following the proprietary GRAM adapter plugin API. These adapters implement the system-specific submission, job exit detection, job cancellation, and (optionally) job exit status determination processes.
gridmap_authorize_and_exec	The `gridmap_authorize_and_exec` tool is a default, but optional, program that is invoked in the target user account as a wrapper around WS GRAM operations in order to make a final safety check for whether WS GRAM should be allowed to operate in that account. This tool provides reasonable privilege limits to guard against service compromise without requiring additional system administrator efforts to manage user policies.

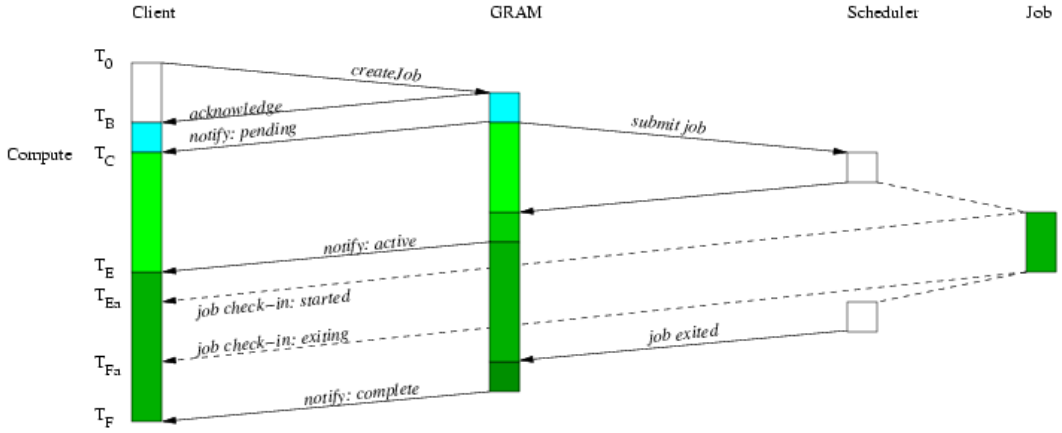
5. Protocol Variations

From a protocol perspective, the longest latency WS GRAM submission scenario involves credential delegation, staging before and after the job, and an explicit hold handshake on file cleanup after the job. The credential refresh feature of WS GRAM can be repeated any number of times, so by longest sequence we mean the longest fixed sequence with at most one credential delegation. Explicit termination is not necessary with WS GRAM so we will not consider that case further.

To understand the following figures which illustrate the protocol sequence: the arrows show communication, signalling, or causal links between tiers in the architecture and the vertical span indicates elapsed time (with the start time at the top of the diagram). Due to unpredictable implementation delays, client and job-observed times are not necessarily ordered with respect to the WS GRAM observed times and the WS GRAM generated state notification messages. The diagrams show one possible ordering but applications (and our measurement methods) must tolerate other orderings as well.

5.1. Minimal Protocol Sequence

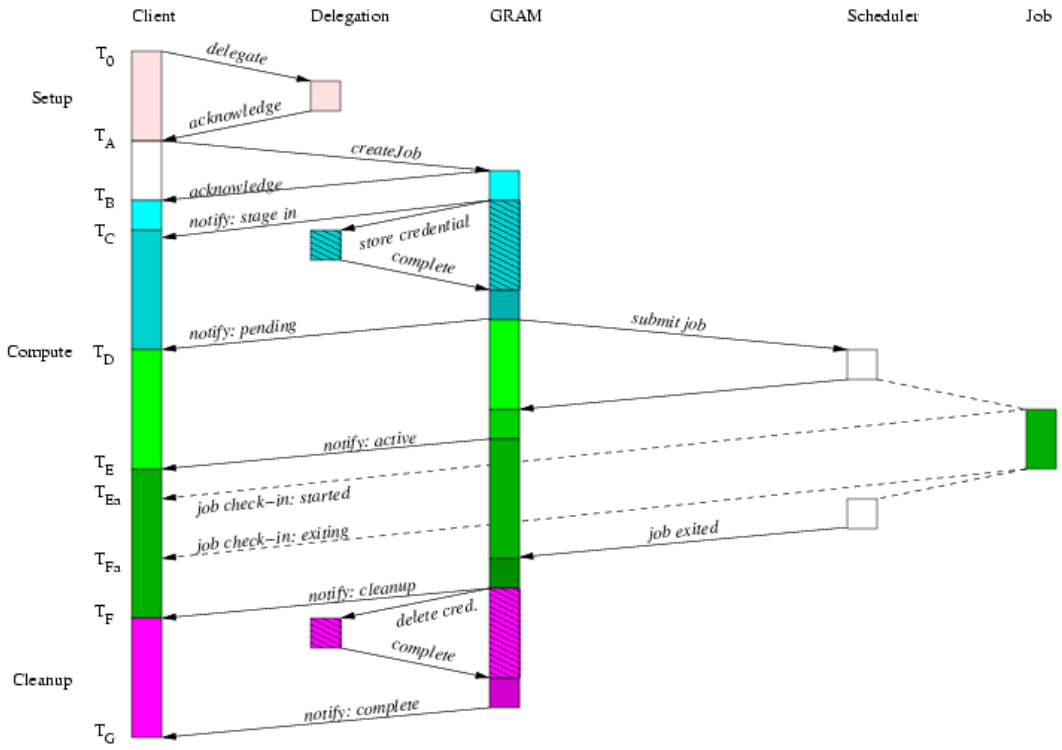
The simplest WS GRAM scenario involves a job that requires neither delegated credentials nor staging and that makes use of the automatic termination of resources to avoid an explicit termination request. In this case, we can measure the latency and throughput for job submission and notification alone.



Note: Any difference between this case and the same measurement points in the full scenario must be due to the additional overhead of the delegation and staging services on the front-end node?

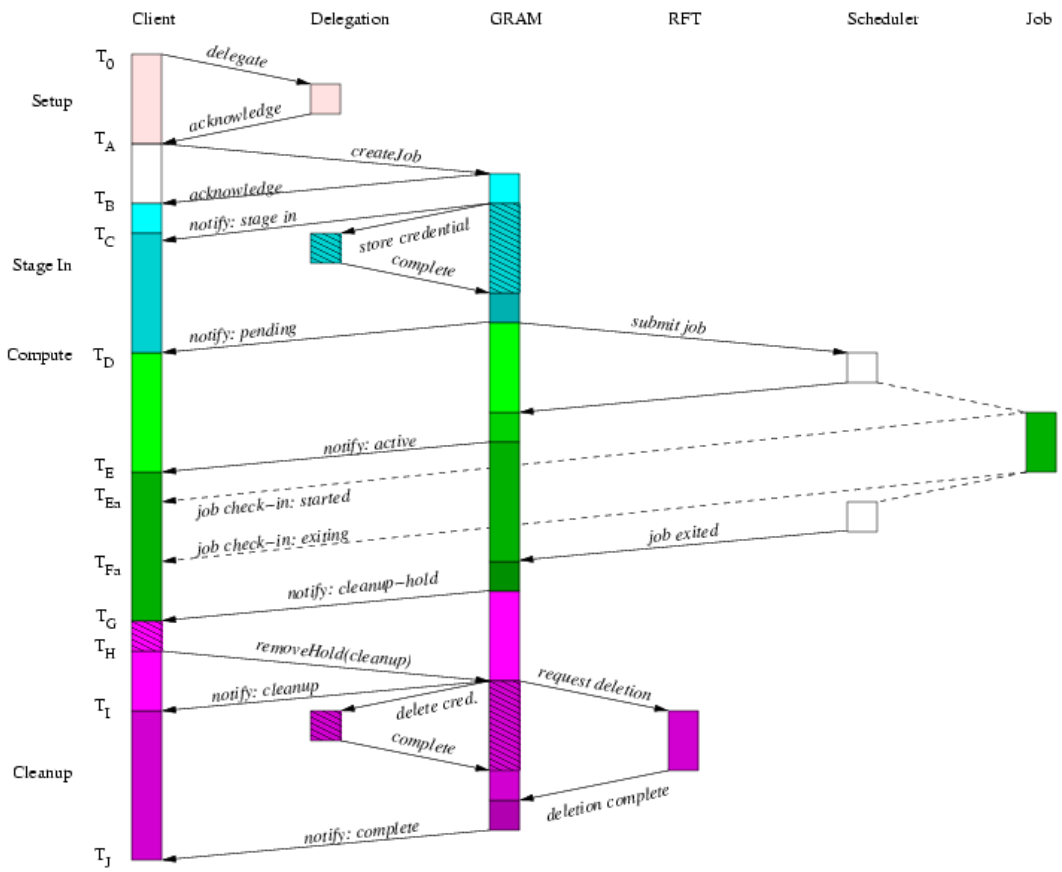
5.2. Non-staging Delegation Sequence

A slightly longer form of job than the minimal sequence is to include credential delegation for use by the job itself, without any staging directives. This sequence is comparable in functionality to previous GRAM releases where delegation was mandatory but staging could be omitted as per the client's request.



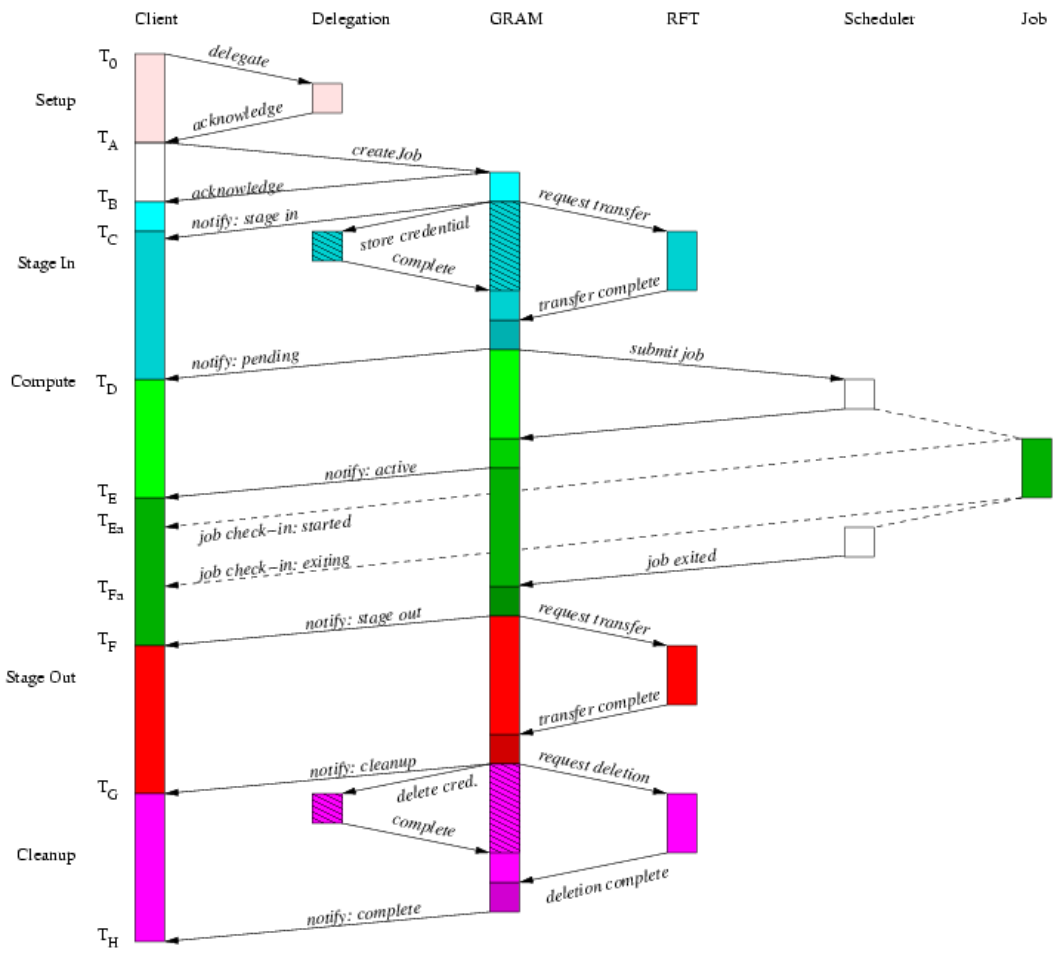
5.3. Non-staging Delegation Sequence with Hold

An optional protocol sequence allows the cleanup state to be held in order to allow a client to safely access output files via the GridFTP server after the job has finished writing them and before the cleanup step deletes them. This variant adds the cleanup hold handshake to the previous scenario.



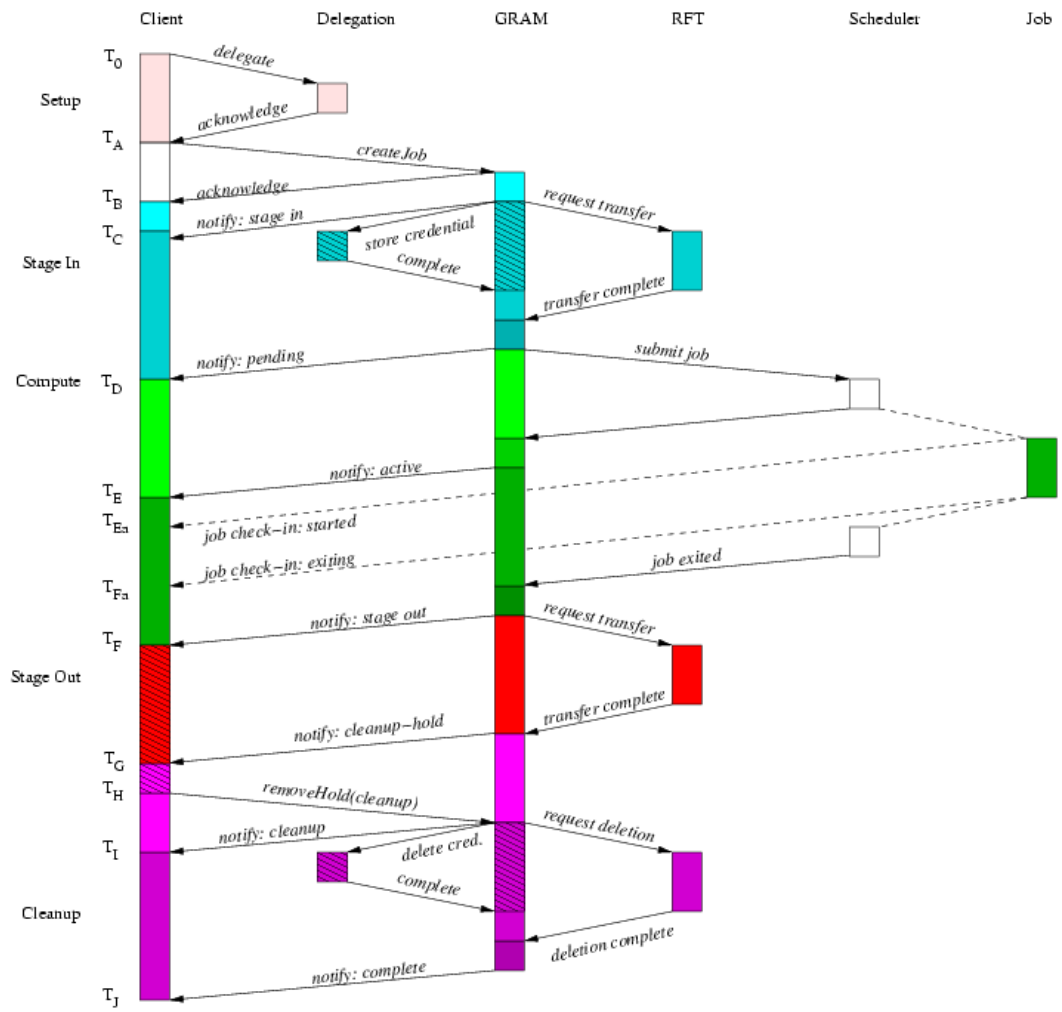
5.4. Staging Sequence

This staging sequence uses almost all of the protocol elements.



5.5. Staging Sequence with Hold

This staging sequence adds the cleanup hold handshake to the staging example to represent a job that has staged files as well as "streamed" output.



6. Performance and scalability

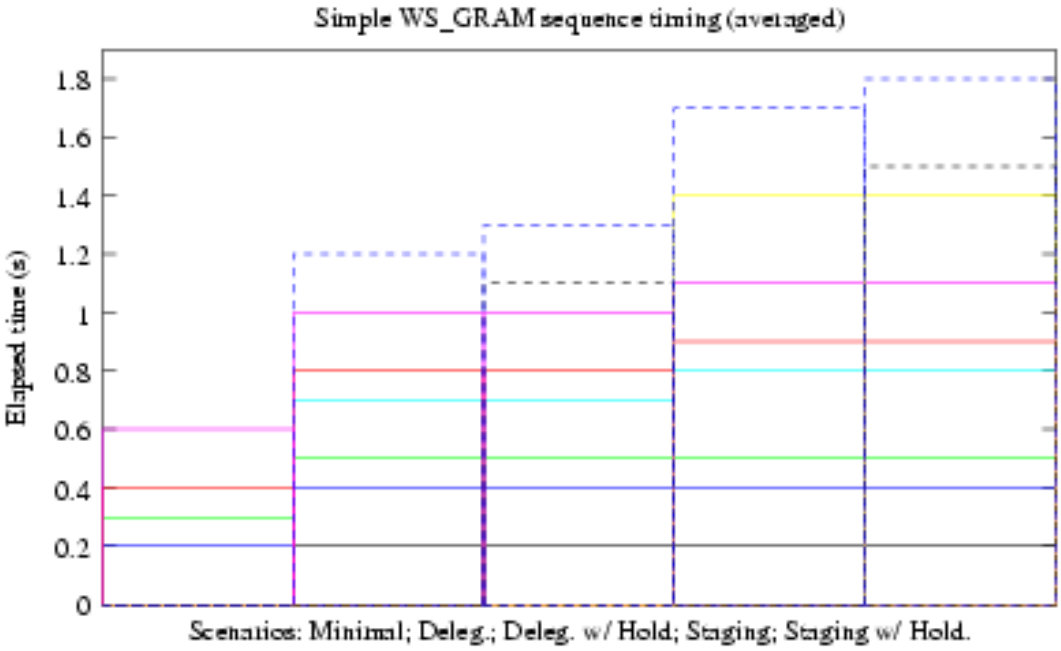
There are several optional parts to the WS GRAM job workflow and protocol. To understand the performance and scalability characteristics of WS GRAM, we must consider variations where different parts of the protocol are used or omitted.

6.1. Basic Client-visible Latency Measurements

We report average performance for #TRIALS submissions of each of the three variant sequences described above, using an instrumented version of the `globusrun-ws` client command-line tool. Each trial is an independent run with no other client load on the test WS GRAM server.

The columns in the figure correspond to the three scenarios described above, and the horizontal bands correspond to the distinct phases of the protocol.

NOTE: these charts currently just illustrate the visualization technique using fictional measurements and do not represent measured data!!



The chart on the left is automatically generated from data files and therefore up to date with any experiments. The chart on the right has been hand-retouched from a snapshot, to color the phases with the same coloring as in the sequence diagrams. Final release documents will include only the retouched chart with consistent coloring.

NOTE: these charts currently just illustrate the visualization technique using fictional measurements and do not represent measured data!!

6.2. Methodology

We use an instrumented version of our command-line client tool to submit one dummy job at a time and log each relevant protocol event with a timestamp. For each of the above scenarios, the protocol events are mapped to the appropriate life cycle boundary for comparison, taking into account the presence or absence of optional protocol sequences in each scenario. The instrumented client measures the initial time by generating a timestamp *before* issuing the first operation request.

The times from one thousand (1000) independent trials are converted to intervals relative to the initiating protocol event and then these intervals are averaged for all trials. The raw timestamped event logs are preserved in case other analysis methods are preferred in the future.

Table 2.8. Mapping of protocol events to life cycle boundaries for each testing scenario

Boundary	Scenario				
	Minimal	**Delegating**	**Delegating w/ Hold**	**Staging**	**Staging w/ Hold**
0. Sequence Initiated	createManaged-Job() invoked	requestSecurityToken() invoked			
1. Delegation complete	requestSecurityToken() returns				
2. Creation complete	createManagedJob() returns				
3. Submission complete	Pending state notified	StageIn state notified			
4. StageIn/Setup complete	Pending state notified				
5. Pending/Start complete	Active state notified				
6. Execution complete	Done state notified	Cleanup state notified	CleanupHold state notified	StageOut state notified	
7. StageOut complete				Cleanup state notified	CleanupHold state notified
8. Hold complete			Cleanup state notified		Cleanup state notified
9. Cleanup complete		Done state notified			

6.3. Pipelined Measurements

For these measurements, we repeat the trials with differing numbers of concurrent (pipelined) submissions from a single client, using a customized test client tool. This leads to a relatively steady-state measurement condition with a parameterized amount of concurrent load on the WS GRAM server.

6.4. Throughput

The submission rate is reported for varying levels of concurrency.

6.5. Concurrency Limits

[TODO]

Chapter 3. GT 4.0 Release Notes: WS GRAM

1. Component Overview

Web Services Grid Resource Allocation and Management (WS GRAM) component comprises a set of WSRF-compliant Web services to locate, submit, monitor, and cancel jobs on Grid computing resources. WS GRAM is not a *job scheduler*, but rather a set of services and clients for communicating with a range of different batch/cluster job schedulers using a common protocol. WS GRAM is meant to address a range of jobs where reliable operation, stateful monitoring, credential management, and file staging are important.

2. Feature Summary

New Features new since 3.2

- Support for mpich-g2 jobs:
  - multi-job submission capabilities
  - ability to coordinate processes in a job
  - ability to coordinate subjobs in a multi-job
- Publishing of the job's exit code
- The ability to select the account under which the remote job will be run. If a user's grid credential is mapped to multiple accounts, then the user can specify, in the *RSL*, under which account the job should be run.
- Optional client-specified hold on a state. Released with the new "release" operation.

Other Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- File / directory clean up after job execution (after file stage out)

Deprecated Features

- managed-job-globusrun has been replaced by *globusrun-ws*.
- Service managed data streaming of job's `stdout/err` during execution.
- File staging using the GASS protocol
- File caching of stages files, e.g. GASS Cache

3. Bug Fixes

- The following link lists all of the bugs resolved for WS GRAM since GT 3.2: Resolved Bugs[1]

4. Known Problems

- The following link lists all of the bugs or enhancements known at the time for the 4.0.0 release: Known Bugs[2]
- Recoverability for jobs which employ any staging directives (i.e. fileStageIn, fileStageOut, and fileCleanUp) is not working. Fixes have been committed to a branch and will be included with the 4.0.1 point release in a month or so.

5. Technology Dependencies

GRAM depends on the following GT components:

- Java WS Core
- Transport-Level Security
- Delegation Service
- RFT
- GridFTP
- MDS - internal libraries

GRAM depends on the following 3rd party software. The dependency exists only for the *batch scheduler*s configured, thus making job submissions possible to the batch scheduling service:

Scheduler adapters are included in the GT 4.0.x releases for these schedulers:

- *PBS*
- *Condor*
- *LSF*

[1] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=GRAM&component=wsrf+discovery+interface&component=wsrf+gram+clients&component=wsrf+managed+execution+job+service&component=wsrf+managed+job+factory+service&component=wsrf+managed+multi+job+service&component=wsrf+rendezvous&component=wsrf+scheduler+interface&component=wsrf+tests&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=RESOLVED&bug_status=VERIFIED&bug_status=CLOSED&resolution=FIXED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfield=resolution&chfieldfrom=2004-09-01&chfieldto=2005-04-22&chfieldvalue=&namedcmd=4.0+gram&newqueryname=&field0-0-0=noop&type0-0-0=noop&value0-0-0=

[2] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=GRAM&component=general&component=wsrf+discovery+interface&component=wsrf+gram+clients&component=wsrf+managed+discovery+execution+job+service&component=wsrf+managed+job+factory+service&component=wsrf+managed+multi+job+service&component=wsrf+rendezvous&component=wsrf+scheduler+interface&component=wsrf+tests&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&bug_severity=blocker&bug_severity=critical&bug_severity=major&bug_severity=normal&bug_severity=minor&bug_severity=trivial&bug_severity=enhancement&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfield=%5BBug+creation%5D&chfieldfrom=2004-01-01&chfieldto=2005-04-30&chfieldvalue=&cmdtype=doit&namedcmd=4.0+gram&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=

Other scheduler adapters available for GT 4.0.x releases:

- SGE scheduler adapter interface[3]
- Loadleveler - as of release 3.3.1 IBM LoadLeveler includes a GRAM Scheduler Adapter. For more information see "What's new" in the LoadLeveler product documentation[4]
- other batch schedulers... (where the GRAM scheduler interface has been implemented)

6. Tested Platforms

Tested platforms for WS GRAM:

- Linux
  - Fedora Core 1 i686
  - Fedora Core 3 i686
  - Fedora Core 3 yup xeon
  - RedHat 7.3 i686
  - RedHat 9 x86
  - Debian Sarge x86
  - Debian 3.1 i686

Tested containers for WS GRAM:

- Java WS Core container
- Tomcat 4.1.31

7. Backward Compatibility Summary

Protocol changes since GT version 3.2:

- The protocol has been changed to be WSRF compliant. There is no backward compatibility between this version and any previous versions.

8. For More Information

Click here[5] for more information about this component.

[3] http://www.lesc.ic.ac.uk/projects/SGE-GT4.html
[4] http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html
[5] index.html

Chapter 4. GT 4.0.1 Incremental Release Notes: WS GRAM

1. Introduction

These release notes are for the incremental release 4.0.1. It includes a summary of changes since 4.0.0, bug fixes since 4.0.0 and any known problems that still exist at the time of the 4.0.1 release. This page is in addition to the top-level 4.0.1 release notes at http://www.globus.org/toolkit/releasenotes/4.0.1.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS GRAM 4.0 Release Notes[1].

2. Changes Summary

The following change has occurred for WS-GRAM:

- The behavior of relative paths in job description elements was fixed. See Bug 3602[2].

3. Bug Fixes

The following bugs were fixed for WS GRAM:

- Bug #3190:[3] Staging jobs are not recoverable
- Bug #3205:[4] globus-scheduler-provider-fork script does not get the correct value for totalnodes
- Bug #3235:[5] Condor SEG seg faults on x86_64
- Bug #3174:[6] globusrun-ws -debug gives seg fault on sles9 x86_64
- Bug #3256:[7] coverability fails in WaitingForStateChanges
- Bug #3114:[8] globusrun-ws should give hints on where the RSL parsing failed
- Bug #3504:[9] Submission without default credential
- Bug #3531:[10] globusrun-ws core dumps with -monitor
- Bug #3544:[11] globusrun-ws using null service path

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Release_Notes.html
[2] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3602
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=3190
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=3205
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=3235
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=3174
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=3256
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=3114
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=3504
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=3531
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=3544

- Bug #3556:[12] globusrun-ws header deserialization error on destroy
- Bug #3554:[13] SEG not relocatable
- Bug #3586:[14] globusrun-ws can't load secure message handler
- Bug #3590:[15] Arguments with '$' in them cause StringIndexOutOfBoundsException
- Bug #3596:[16] PBS missing check for relative directories
- Bug #3599:[17] Job can restart with internal state "Restart"
- Bug #3598:[18] globusrun-ws -self doesn't work for multi-jobs
- Bug #3602:[19] Relative stdout/err paths should be relative to directory not ${GLOBUS_USER_HOME}
- Bug #3230:[20] test security requires a grid-mapfile entry now
- Bug #3252:[21] PBS scheduler event generator doesn't build on Solaris
- Bug #3292:[22] occasional SEG write failure
- Bug #3301:[23] Default ${GLOBUS_SCRATCH_DIR} is wrong when using ant 1.6.2
- Bug #3309:[24] globusrun-ws usage message incorrect
- Bug #3319:[25] GRAM scheduler tests fail with condor
- Bug #3320:[26] relative directory test and stdin test assume files or directories
- Bug #3442:[27] GRAM should use 'mpiexec' before 'mpirun'
- Bug #3447:[28] PBS SEG generates bad dates
- Bug #3410:[29] globus_xio writev() error
- Bug #3532:[30] Recovery thread dies in Tomcat
- Bug #2730:[31] Job description variable fault lost

[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=3556
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=3554
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=3586
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3590
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3596
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=3599
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=3598
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=3602
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=3230
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=3252
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=3292
[23] http://bugzilla.globus.org/globus/show_bug.cgi?id=3301
[24] http://bugzilla.globus.org/globus/show_bug.cgi?id=3309
[25] http://bugzilla.globus.org/globus/show_bug.cgi?id=3319
[26] http://bugzilla.globus.org/globus/show_bug.cgi?id=3320
[27] http://bugzilla.globus.org/globus/show_bug.cgi?id=3442
[28] http://bugzilla.globus.org/globus/show_bug.cgi?id=3447
[29] http://bugzilla.globus.org/globus/show_bug.cgi?id=3410
[30] http://bugzilla.globus.org/globus/show_bug.cgi?id=3532
[31] http://bugzilla.globus.org/globus/show_bug.cgi?id=2730

- Bug #3132:[32] Staging parameters should be automatically determined if running co located
- Bug #3333:[33] Multijob destruction failure
- Bug #3342:[34] Reliable Job Create isn't entirely reliable
- Bug #3498:[35] simple multi-jobs periodically hangs
- Bug #3524:[36] stdout/err RPs aren't being prefixed with user home
- Bug #3533:[37] notification not fired in Tomcat
- Bug #3526:[38] secManager.credentialToSubject problem
- Bug #3075:[39] Issues with globusrun-ws and -Sf option
- Bug #3397:[40] Misconfiguration of GCC "-Wno-unused" flag for Tru64 native compiler
- Bug #3474:[41] tmp subdir is not world-writable
- Bug #3517:[42] WS-GRAM build failure in 4_0_branch
- Bug #3426:[43] globus-job-manager-script.pl broken on !LD_LIBRARY_PATH platforms

4. Known Problems

The following problems are known to exist for WS GRAM at the time of the 4.0.1 release:

- High throughput can cause an OutOfMemoryError exception. Fixes will appear in the next development release.
- Bug #2974:[44] Inconsistent Arguments
- Bug #1562:[45] Condor jobs fail when running globus-sh-exec job
- Bug #2527:[46] Add a failure case for non-existant queue to scheduler test suite
- Bug #2049:[47] Batch providers need a namespace
- Bug #2967:[48] Job submission inconsistent behaviour with delegated credential

[32] http://bugzilla.globus.org/globus/show_bug.cgi?id=3132
[33] http://bugzilla.globus.org/globus/show_bug.cgi?id=3333
[34] http://bugzilla.globus.org/globus/show_bug.cgi?id=3342
[35] http://bugzilla.globus.org/globus/show_bug.cgi?id=3498
[36] http://bugzilla.globus.org/globus/show_bug.cgi?id=3524
[37] http://bugzilla.globus.org/globus/show_bug.cgi?id=3533
[38] http://bugzilla.globus.org/globus/show_bug.cgi?id=3526
[39] http://bugzilla.globus.org/globus/show_bug.cgi?id=3075
[40] http://bugzilla.globus.org/globus/show_bug.cgi?id=3397
[41] http://bugzilla.globus.org/globus/show_bug.cgi?id=3474
[42] http://bugzilla.globus.org/globus/show_bug.cgi?id=3517
[43] http://bugzilla.globus.org/globus/show_bug.cgi?id=3426
[44] http://bugzilla.globus.org/globus/show_bug.cgi?id=2974
[45] http://bugzilla.globus.org/globus/show_bug.cgi?id=1562
[46] http://bugzilla.globus.org/globus/show_bug.cgi?id=2527
[47] http://bugzilla.globus.org/globus/show_bug.cgi?id=2049
[48] http://bugzilla.globus.org/globus/show_bug.cgi?id=2967

- Bug #3204:[49] InternalStateEnumeration schema cleanup
- Bug #3238:[50] Not implemented option
- Bug #3185:[51] pbs job manager exit code
- Bug #2951:[52] strange security output from multiob
- Bug #3242:[53] Software selection thru WS GRAM RSL
- Bug #3002:[54] sudo path isn't set correctly in libexec/globus-sh-tools-vars.sh
- Bug #3495:[55] queue information, job count not reported to MDS
- Bug #3529:[56] setup/postinstall fatal errors should be warnings
- Bug #3458:[57] globusrun-ws isn't automatically releasing holds
- Bug #3534:[58] globusrun-ws needs a -release option
- Bug #3384:[59] Inconsistent jobType/count parameter semantics
- Bug #3571:[60] ant not found during install
- Bug #3416:[61] hardcoded jobtypes
- Bug #3575:[62] SEG dependent on GLOBUS_LOCATION env var
- Bug #3579:[63] takes over 4 seconds to start a single job with 4.0.1 branch container
- Bug #3611:[64] get job state and queue information from SEG

5. For More Information

Click here[65] for more information about this component.

[49] http://bugzilla.globus.org/globus/show_bug.cgi?id=3204
[50] http://bugzilla.globus.org/globus/show_bug.cgi?id=3238
[51] http://bugzilla.globus.org/globus/show_bug.cgi?id=3185
[52] http://bugzilla.globus.org/globus/show_bug.cgi?id=2951
[53] http://bugzilla.globus.org/globus/show_bug.cgi?id=3242
[54] http://bugzilla.globus.org/globus/show_bug.cgi?id=3002
[55] http://bugzilla.globus.org/globus/show_bug.cgi?id=3495
[56] http://bugzilla.globus.org/globus/show_bug.cgi?id=3529
[57] http://bugzilla.globus.org/globus/show_bug.cgi?id=3458
[58] http://bugzilla.globus.org/globus/show_bug.cgi?id=3534
[59] http://bugzilla.globus.org/globus/show_bug.cgi?id=3384
[60] http://bugzilla.globus.org/globus/show_bug.cgi?id=3571
[61] http://bugzilla.globus.org/globus/show_bug.cgi?id=3416
[62] http://bugzilla.globus.org/globus/show_bug.cgi?id=3575
[63] http://bugzilla.globus.org/globus/show_bug.cgi?id=3579
[64] http://bugzilla.globus.org/globus/show_bug.cgi?id=3611
[65] index.html

Chapter 5. GT 4.0.2 Incremental Release Notes: WS GRAM

1. Introduction

These release notes are for the incremental release 4.0.2. It includes a summary of changes since 4.0.1, bug fixes since 4.0.1 and any known problems that still exist at the time of the 4.0.2 release. This page is in addition to the top-level 4.0.2 release notes at http://www.globus.org/toolkit/releasenotes/4.0.2.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS GRAM 4.0 Release Notes[1].

2. Changes Summary

Overall, WS GRAM in 4.0.2 is much improved over 4.0.1.

A focused (and ongoing) effort for large job loads to complete reliably has helped to identify a number of improvements. See Campaign 4197[2] for the complete details and note the 6 bugs marked as dependencies to this bug. One of the 6 improvements made (Enh 4330[3]) was implementing a better algorithm for processing jobs from the internal job run queue. This improved responsiveness significantly. During a large throughput run to a 4.0.1 WS GRAM service, a separate simple /bin/date Fork job took > 10 minutes to return. In 4.0.2, the same Fork job was processed in ~2 minutes.

The WS GRAM testing infrastructure has been improved with automated throughput testing - http://skynet-login.isi.edu/gram-testing/. Nightly, the WS GRAM code is checked out of various CVS branches, built, and throughput tests are run. This has helped identify bugs more quickly and ease the effort of resolving bugs that can be difficult to reproduce. For a good example, see Bug 4235[4]. The WS GRAM service is not released until these tests are passing consistently from the CVS release branch. This has helped us provide an overall better WS GRAM service.

There were 38 bug fixes since 4.0.1; here are a few worth highlighting:

- Improvements were made to help identify service installation/configuration problems between WS GRAM and the local Resource Manager by improving the SEG reporting on fatal errors. Bug 4229[5]
- The variable "SCRATCH_DIRECTORY" was not being set in the job's environment. Bug 4192[6]
- WS GRAM was not failing the job on some fatal error conditions. Bug 4247[7] Bug 4279[8] Bug 3631[9]
- Job error reporting was improved. Bug 4273[10] Bug 4241[11]
- For the (default) INFO logging, WS GRAM will produce entry and exit logging for a job. Bug 3742[12]

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Release_Notes.html
[2] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4197
[3] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4330
[4] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4253
[5] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4229
[6] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4192
[7] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4247
[8] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4279
[9] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3631
[10] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3742
[11] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4241
[12] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3742

3. Bug Fixes

The following bugs were fixed for WS GRAM:

- Bug #3190:[13] Staging jobs are not recoverable
- Bug #3631:[14] Delegation fetching errors should cause the job to fail i...
- Bug #3642:[15] Incorrect canceling of condor jobs
- Bug #3738:[16] self auth is applied to globusrun-ws, but host auth is st...
- Bug #3698:[17] user not allowed to override GLOBUS_LOCATION
- Bug #3002:[18] sudo path isn't set correctly in libexec/globus-sh-tools-...
- Bug #3812:[19] Absolute Windows paths not registering as absolute paths.
- Bug #3772:[20] Problem parsing certain PBS scheduler logs
- Bug #3777:[21] PBS SEG module can stop prematurely on PBS restart
- Bug #3458:[22] globusrun-ws isn't automatically releasing holds
- Bug #3966:[23] PBS SEG not refreshing
- Bug #3931:[24] Failed to retrieve Resource Properties from ManagedJobSer...
- Bug #3935:[25] GRAM setup scripts must be run from $GLOBUS_LOCATION/setu...
- Bug #4187:[26] AuthorizationHelper broken on container restart
- Bug #3185:[27] pbs job manager exit code
- Bug #4229:[28] Improve SEG fatal Error logging
- Bug #2266:[29] SEG needs to communicate errors better
- Bug #3757:[30] Remote exit status 0 ambiguity

[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=3190
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=3631
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3642
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3738
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=3698
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=3002
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=3812
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=3772
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=3777
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=3458
[23] http://bugzilla.globus.org/globus/show_bug.cgi?id=3966
[24] http://bugzilla.globus.org/globus/show_bug.cgi?id=3931
[25] http://bugzilla.globus.org/globus/show_bug.cgi?id=3935
[26] http://bugzilla.globus.org/globus/show_bug.cgi?id=4187
[27] http://bugzilla.globus.org/globus/show_bug.cgi?id=3185
[28] http://bugzilla.globus.org/globus/show_bug.cgi?id=4229
[29] http://bugzilla.globus.org/globus/show_bug.cgi?id=2266
[30] http://bugzilla.globus.org/globus/show_bug.cgi?id=3757

- Bug #4247:[31] JSM registration error not failing job
- Bug #4192:[32] Missing SCRATCH_DIRECTORY env var.
- Bug #4279:[33] File mapping errors should fail the job.
- Bug #4273:[34] globusrun-ws error reports are ugly
- Bug #4253:[35] unable to monitor job for state changes
- Bug #4297:[36] PBS SEG module does not follow torque log rotation
- Bug #4198:[37] Bad reg-ex in condor poll script
- Bug #4327:[38] Script crashing with "Terminated"
- Bug #4329:[39] Missing context upon job recovery.
- Bug #4330:[40] Job Run Queue Processing Algorithm Inefficient
- Bug #4331:[41] Jobs disappear from state machine under heavy loads.
- Bug #2397:[42] optimize rft request if running in same container
- Bug #4241:[43] Allow for multi-line error from scheduler commands
- Bug #4170:[44] LoadLeveler 3.3.1 includes a GT4 GRAM Scheduler Adapter -...
- Bug #3770:[45] Incorrect GPT metadata breaks VDT builds
- Bug #3687:[46] %ENV mispelled %env in globus-job-manager-script.pl
- Bug #3699:[47] pbs scheduler event generator must be started after logs ...
- Bug #1039:[48] jobmanager-condor doesn't support the java universe
- Bug #4159:[49] Improve container error message when SEG execution fails
- Bug #4111:[50] locking / synch problems

[31] http://bugzilla.globus.org/globus/show_bug.cgi?id=4247
[32] http://bugzilla.globus.org/globus/show_bug.cgi?id=4192
[33] http://bugzilla.globus.org/globus/show_bug.cgi?id=4279
[34] http://bugzilla.globus.org/globus/show_bug.cgi?id=4273
[35] http://bugzilla.globus.org/globus/show_bug.cgi?id=4253
[36] http://bugzilla.globus.org/globus/show_bug.cgi?id=4297
[37] http://bugzilla.globus.org/globus/show_bug.cgi?id=4198
[38] http://bugzilla.globus.org/globus/show_bug.cgi?id=4327
[39] http://bugzilla.globus.org/globus/show_bug.cgi?id=4329
[40] http://bugzilla.globus.org/globus/show_bug.cgi?id=4330
[41] http://bugzilla.globus.org/globus/show_bug.cgi?id=4331
[42] http://bugzilla.globus.org/globus/show_bug.cgi?id=2397
[43] http://bugzilla.globus.org/globus/show_bug.cgi?id=4241
[44] http://bugzilla.globus.org/globus/show_bug.cgi?id=4170
[45] http://bugzilla.globus.org/globus/show_bug.cgi?id=3770
[46] http://bugzilla.globus.org/globus/show_bug.cgi?id=3687
[47] http://bugzilla.globus.org/globus/show_bug.cgi?id=3699
[48] http://bugzilla.globus.org/globus/show_bug.cgi?id=1039
[49] http://bugzilla.globus.org/globus/show_bug.cgi?id=4159
[50] http://bugzilla.globus.org/globus/show_bug.cgi?id=4111

- Bug #3742:[51] GRAM should log entry/exit of jobs to INFO level
- Bug #4339:[52] Condor adapter's poll subroutine not working for count > 1

4. Known Problems

The following problems are known to exist for WS GRAM at the time of the 4.0.1 release:

- Bug #1562:[53]Condor jobs fail when running globus-sh-exec job
- Bug #2049:[54]Batch providers need a namespace
- Bug #3571:[55]ant not found during install
- Bug #3575:[56]SEG dependent on GLOBUS_LOCATION env var
- Bug #3579:[57]takes over 4 seconds to start a single job with 4.0.1 bra...
- Bug #3778:[58]WS-GRAM dependant on GT2 job manager
- Bug #2527:[59]Add a failure case for non-existant queue to scheduler te...
- Bug #2286:[60]internationalization
- Bug #3495:[61]queue information, job count not reported to MDS
- Bug #3726:[62]GlobusRun error message typo
- Bug #3844:[63]Bad GPT metadata for globus_wsrf_gram_client_tools-1.0
- Bug #3803:[64]Default scratchDirectory doesn't exist
- Bug #3384:[65]Inconsistent jobType/count parameter semantics
- Bug #3892:[66]Out of date performance data?
- Bug #3672:[67]Streaming with PBS fails
- Bug #3897:[68]Must modify Globus in order to use authorization callouts

[51] http://bugzilla.globus.org/globus/show_bug.cgi?id=3742
[52] http://bugzilla.globus.org/globus/show_bug.cgi?id=4339
[53] http://bugzilla.globus.org/globus/show_bug.cgi?id=1562
[54] http://bugzilla.globus.org/globus/show_bug.cgi?id=2049
[55] http://bugzilla.globus.org/globus/show_bug.cgi?id=3571
[56] http://bugzilla.globus.org/globus/show_bug.cgi?id=3575
[57] http://bugzilla.globus.org/globus/show_bug.cgi?id=3579
[58] http://bugzilla.globus.org/globus/show_bug.cgi?id=3778
[59] http://bugzilla.globus.org/globus/show_bug.cgi?id=2527
[60] http://bugzilla.globus.org/globus/show_bug.cgi?id=2286
[61] http://bugzilla.globus.org/globus/show_bug.cgi?id=3495
[62] http://bugzilla.globus.org/globus/show_bug.cgi?id=3726
[63] http://bugzilla.globus.org/globus/show_bug.cgi?id=3844
[64] http://bugzilla.globus.org/globus/show_bug.cgi?id=3803
[65] http://bugzilla.globus.org/globus/show_bug.cgi?id=3384
[66] http://bugzilla.globus.org/globus/show_bug.cgi?id=3892
[67] http://bugzilla.globus.org/globus/show_bug.cgi?id=3672
[68] http://bugzilla.globus.org/globus/show_bug.cgi?id=3897

- Bug #3911:[69]Bug in GRAM documentation for Condor log file
- Bug #3746:[70]bad link on scheduler tutorial
- Bug #4116:[71]globus_module_activate(GLOBUS_GRAM_CLIENT_MODULE) behav...
- Bug #3675:[72]wsrf gram client file streaming error
- Bug #4153:[73]gram scheduler test failing - submit202
- Bug #4161:[74]Managed Job Types, holding, state, userSubject and exitCo...
- Bug #4162:[75]fill in LINK TO SEG API Doc
- Bug #4181:[76]Allow File Staging To/From globusrun-ws application witho...
- Bug #4182:[77]Improve Condor/Fork Job Monitoring for reliability and se...
- Bug #4178:[78]no job output when streaming with globusrun-ws
- Bug #4191:[79]globusrun-ws job submission hangs
- Bug #3910:[80]Bad permissions on condor log file prevents job submissions
- Bug #3529:[81]setup/postinstall fatal errors should be warnings
- Bug #4216:[82]Empty submit_test/submitxxx.err causes FAILure in Local t...
- Bug #4275:[83]globus-gram-local-proxy-tool fails on Solaris
- Bug #4388:[84]globusrun-ws handles -self differently than java clients
- Bug #3748:[85]WS-GRAM Plugable Resource Manager Backend
- Bug #3751:[86]convert persistence data store from files to a database
- Bug #4207:[87]Enabling dynamic job description variables using softenv
- Bug #4319:[88]globus_scheduler_event_generator doesn't build with -stat...

[69] http://bugzilla.globus.org/globus/show_bug.cgi?id=3911
[70] http://bugzilla.globus.org/globus/show_bug.cgi?id=3746
[71] http://bugzilla.globus.org/globus/show_bug.cgi?id=4116
[72] http://bugzilla.globus.org/globus/show_bug.cgi?id=3675
[73] http://bugzilla.globus.org/globus/show_bug.cgi?id=4153
[74] http://bugzilla.globus.org/globus/show_bug.cgi?id=4161
[75] http://bugzilla.globus.org/globus/show_bug.cgi?id=4162
[76] http://bugzilla.globus.org/globus/show_bug.cgi?id=4181
[77] http://bugzilla.globus.org/globus/show_bug.cgi?id=4182
[78] http://bugzilla.globus.org/globus/show_bug.cgi?id=4178
[79] http://bugzilla.globus.org/globus/show_bug.cgi?id=4191
[80] http://bugzilla.globus.org/globus/show_bug.cgi?id=3910
[81] http://bugzilla.globus.org/globus/show_bug.cgi?id=3529
[82] http://bugzilla.globus.org/globus/show_bug.cgi?id=4216
[83] http://bugzilla.globus.org/globus/show_bug.cgi?id=4275
[84] http://bugzilla.globus.org/globus/show_bug.cgi?id=4308
[85] http://bugzilla.globus.org/globus/show_bug.cgi?id=3748
[86] http://bugzilla.globus.org/globus/show_bug.cgi?id=3751
[87] http://bugzilla.globus.org/globus/show_bug.cgi?id=4207
[88] http://bugzilla.globus.org/globus/show_bug.cgi?id=4319

- Bug #3948:[89]Service must release all of its resources on deactivation
- Bug #4321:[90]globusrun-ws freezes with "-job-delegate" option

5. For More Information

Click here[91] for more information about this component.

[89] http://bugzilla.globus.org/globus/show_bug.cgi?id=3948
[90] http://bugzilla.globus.org/globus/show_bug.cgi?id=4321
[91] index.html

Chapter 6. GT 4.0.3 Incremental Release Notes: WS GRAM

1. Introduction

These release notes are for the incremental release 4.0.3. It includes a summary of changes since 4.0.2, bug fixes since 4.0.2 and any known problems that still exist at the time of the 4.0.3 release. This page is in addition to the top-level 4.0.3 release notes at http://www.globus.org/toolkit/releasenotes/4.0.3.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS GRAM 4.0 Release Notes[1].

2. Changes Summary

The following changes were made to WS GRAM since GT 4.0.2:

- The MEJS now caches the user home directory values.
- Error reporting for multijobs was improved.
- Run queue threads now have more unique name (helps with debugging thread issues).

3. Bug Fixes

The following bugs have been fixed for WS GRAM since GT 4.0.2:

- Bug 4376:[2] Multijob shouldn't be distributing credential endpoints to subjobs
- Bug 4577:[3] GramJob not receiving notifications

4. Known Problems

The following problems are known to exist for WS GRAM at the time of the 4.0.3 release:

- Bug 2734:[4]non-shared FS scheduler file list
- Bug 3675:[5]wsrf gram client file streaming error
- Bug 3726:[6]GlobusRun error message typo
- Bug 2974:[7]Inconsistent Arguments

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=4376
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=4577
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=2734
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=3675
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=3726
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=2974

- Bug 4356:[8]globusrun-ws is overriding manual subjob credential deleg...
- Bug 4513:[9]LD_LIBRARY_PATH should not be set if no library_path is s...
- Bug 4584:[10]security descriptor uses operation field name instead of ...
- Bug 2527:[11]Add a failure case for non-existant queue to scheduler te...
- Bug 2623:[12]service summary/diagnostics
- Bug 2680:[13]Can't change source url for gram job staged files
- Bug 2776:[14]add directive to set the priority in the job description ...
- Bug 3801:[15]condor commands need CONDOR_CONFIG
- Bug 3829:[16]Separating the factory type from the resource name
- Bug 3865:[17]Enhance RSL with Job Name
- Bug 3866:[18]Support for parametric or array job types
- Bug 3876:[19]Automatic transfer of all files modified by a job in GRAM
- Bug 3948:[20]Service must release all of its resources on deactivation
- Bug 4009:[21]Use pbsdsh if available
- Bug 4153:[22]gram scheduler test failing - submit202
- Bug 4191:[23]globusrun-ws job submission hangs
- Bug 4216:[24]Empty submit_test/submitxxx.err causes FAILure in Local t...
- Bug 4275:[25]globus-gram-local-proxy-tool fails on Solaris
- Bug 4452:[26]job submission response is effected by java 1.5 thread pr...
- Bug 4461:[27]Configuring GRAM RM interface to use SSH or RSH

[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=4356
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=4513
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=4584
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=2527
[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=2623
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=2680
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=2776
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3801
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3829
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=3865
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=3866
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=3876
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=3948
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=4009
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=4153
[23] http://bugzilla.globus.org/globus/show_bug.cgi?id=4191
[24] http://bugzilla.globus.org/globus/show_bug.cgi?id=4216
[25] http://bugzilla.globus.org/globus/show_bug.cgi?id=4275
[26] http://bugzilla.globus.org/globus/show_bug.cgi?id=4452
[27] http://bugzilla.globus.org/globus/show_bug.cgi?id=4461

- Bug 4533:[28]make-test-script.pl bug
- Bug 4539:[29]Add script error #21 troubleshooting information.
- Bug 4560:[30]Changing default ports with ManagedJobFactoryClientHelper
- Bug 4661:[31]Example submit job
- Bug 3638:[32]Missing SEG developer documentation
- Bug 4118:[33]Rendezvous Documentation Needs Improvement
- Bug 3672:[34]Streaming with PBS fails
- Bug 3242:[35]Software selection thru WS GRAM RSL
- Bug 2250:[36]delegation required resource property
- Bug 2578:[37]reliable state change notification
- Bug 2579:[38]reliable state change notification
- Bug 2624:[39]Multiple job hold states and parameterized release operation
- Bug 3088:[40]Default Job Environment
- Bug 3384:[41]Inconsistent jobType/count parameter semantics
- Bug 3460:[42]State Mask for Notifications
- Bug 3480:[43]Allow usage of ssh to start SEG
- Bug 3529:[44]setup/postinstall fatal errors should be warnings
- Bug 3569:[45]Selectable jobType default per factory
- Bug 3575:[46]SEG dependent on GLOBUS_LOCATION env var
- Bug 3714:[47]Add Job elements to GLUECE RP

[28] http://bugzilla.globus.org/globus/show_bug.cgi?id=4533
[29] http://bugzilla.globus.org/globus/show_bug.cgi?id=4539
[30] http://bugzilla.globus.org/globus/show_bug.cgi?id=4560
[31] http://bugzilla.globus.org/globus/show_bug.cgi?id=4661
[32] http://bugzilla.globus.org/globus/show_bug.cgi?id=3638
[33] http://bugzilla.globus.org/globus/show_bug.cgi?id=4118
[34] http://bugzilla.globus.org/globus/show_bug.cgi?id=3672
[35] http://bugzilla.globus.org/globus/show_bug.cgi?id=3242
[36] http://bugzilla.globus.org/globus/show_bug.cgi?id=2250
[37] http://bugzilla.globus.org/globus/show_bug.cgi?id=2578
[38] http://bugzilla.globus.org/globus/show_bug.cgi?id=2579
[39] http://bugzilla.globus.org/globus/show_bug.cgi?id=2624
[40] http://bugzilla.globus.org/globus/show_bug.cgi?id=3088
[41] http://bugzilla.globus.org/globus/show_bug.cgi?id=3384
[42] http://bugzilla.globus.org/globus/show_bug.cgi?id=3460
[43] http://bugzilla.globus.org/globus/show_bug.cgi?id=3480
[44] http://bugzilla.globus.org/globus/show_bug.cgi?id=3529
[45] http://bugzilla.globus.org/globus/show_bug.cgi?id=3569
[46] http://bugzilla.globus.org/globus/show_bug.cgi?id=3575
[47] http://bugzilla.globus.org/globus/show_bug.cgi?id=3714

- Bug 3741:[48] pluggable job proxy creation
- Bug 3748:[49] WS-GRAM Plugable Resource Manager Backend
- Bug 3751:[50] convert persistence data store from files to a database
- Bug 3803:[51] Default scratchDirectory doesn't exist
- Bug 3897:[52] Must modify Globus in order to use authorization callouts
- Bug 4178:[53] no job output when streaming with globusrun-ws
- Bug 4207:[54] Enabling dynamic job description variables using softenv
- Bug 4410:[55] GT4 WS-GRAM Auditing
- Bug 4431:[56] freeze by Unsubmitted on personal WS GRAM
- Bug 4474:[57] globus-gridmap-and-execute problem with additional PDPs
- Bug 4515:[58] throughput tester - gram errors for cvs HEAD
- Bug 4528:[59] WS-GRAM Auditing Test Integration on TeraGrid
- Bug 4545:[60] WS GRAM JSDL Support in GT 4.2
- Bug 4550:[61] Multijob code not checking for existence of job credential
- Bug 3126:[62] RFT resource reference property TransferKey is poorly named
- Bug 2980:[63] Troubleshooting globusrun-ws

5. For More Information

Click here[64] for more information about this component.

[48] http://bugzilla.globus.org/globus/show_bug.cgi?id=3741
[49] http://bugzilla.globus.org/globus/show_bug.cgi?id=3748
[50] http://bugzilla.globus.org/globus/show_bug.cgi?id=3751
[51] http://bugzilla.globus.org/globus/show_bug.cgi?id=3803
[52] http://bugzilla.globus.org/globus/show_bug.cgi?id=3897
[53] http://bugzilla.globus.org/globus/show_bug.cgi?id=4178
[54] http://bugzilla.globus.org/globus/show_bug.cgi?id=4207
[55] http://bugzilla.globus.org/globus/show_bug.cgi?id=4410
[56] http://bugzilla.globus.org/globus/show_bug.cgi?id=4431
[57] http://bugzilla.globus.org/globus/show_bug.cgi?id=4474
[58] http://bugzilla.globus.org/globus/show_bug.cgi?id=4515
[59] http://bugzilla.globus.org/globus/show_bug.cgi?id=4528
[60] http://bugzilla.globus.org/globus/show_bug.cgi?id=4545
[61] http://bugzilla.globus.org/globus/show_bug.cgi?id=4550
[62] http://bugzilla.globus.org/globus/show_bug.cgi?id=3126
[63] http://bugzilla.globus.org/globus/show_bug.cgi?id=2980
[64] index.html

Chapter 7. GT 4.0.4 Incremental Release Notes: WS GRAM

1. Introduction

These release notes are for the incremental release 4.0.4. It includes a summary of changes since 4.0.3, bug fixes since 4.0.3 and any known problems that still exist at the time of the 4.0.4 release. This page is in addition to the top-level 4.0.4 release notes at http://www.globus.org/toolkit/releasenotes/4.0.4.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS GRAM 4.0 Release Notes[1].

2. Changes Summary

The most significant enhancement in 4.0.4 was the improvement in processing 1000+ job runs from a single client. This type and scale of processing is much more reliable in 4.0.4 than 4.0.3. Upgrading is recommended. More can be read from the details of these (ongoing) campaigns:

- performance improvements for large run job submissions using condor-g[2]
- GRAM2 GRAM4 Performance Comparison[3]

NOTE: Between GT releases 4.0.3 and 4.0.4, there was a GRAM release in VDT 1.6.x. The GRAM release in VDT 1.6.x included additional functionality that is NOT included in 4.0.4. This functionality will be added in the following point release - 4.0.5. Below is a list of the functionality in VDT 1.6 that is NOT in 4.0.4

- GRAM service auditing support[4]
- For audit support, change use of client-generated job resource keys[5]
- Default job description subtitution variables[6]
- Cache user home directory to improve performance[7]
- globus-job-*-ws tools[8]
- job description extension support[9]
- softenv extension support[10]
- Pre-WS GRAM LRM job monitoring with SEG[11]

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=4664
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=4751
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=4409
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=4984
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=4207
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=3841
[8] http://www-unix.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Command-line_Frag.html
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=3766
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=3967
[11] http://www-unix.mcs.anl.gov/~bester/campaigns/521/using-seg-with-jm.html

A campaign tracking the work to add this functionality in 4.0.5 is here[12]

3. Bug Fixes

- Bug 5027:[13] more efficient processing of internal job states
- Bug 4865:[14] Events from SEG to JobStateMonitor are deleted too early i.e. jobs keep stuck
- Bug 4777:[15] Excessive memory usage for SEG modules
- Bug 4983:[16] Flag that triggers default SoftEnv keys usage in job submission
- Bug 4706:[17] Fork jobs without arguments may receive uninitialized data as arguments
- Bug 4764:[18] globusrun-ws output for bad -Ft is unhelpful.
- Bug 4732:[19] globusrun-ws segmentation fault (FC5 x86_64)
- Bug 4884:[20] GramJob post-submit subscriptions broken
- Bug 4860:[21] LSF jobmanager doesn't preserve environment variables with spaces

4. Known Problems

- Bug 4761:[22]Scheduler Tutorial is missing WS-GRAM setup package
- Bug 4778:[23]WS-Fork job manager doesn't set environment up for mpi jobs
- Bug 4787:[24]no lifetime management for WS Rendezvous
- Bug 4790:[25]stdout RP gets null
- Bug 4817:[26]Condor OS and ARCH do not have dynamic defaults
- Bug 4859:[27]globusrun-ws staging error after gt4 deployed to tomcat
- Bug 4864:[28]environment variables containing '=' get escaped
- Bug 4874:[29]LRUCache problem during job submission from Condor-G to GRAM4

[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=4924
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=5027
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=4865
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=4777
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=4983
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=4706
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=4764
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=4732
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=4884
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=4860
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=4761
[23] http://bugzilla.globus.org/globus/show_bug.cgi?id=4778
[24] http://bugzilla.globus.org/globus/show_bug.cgi?id=4787
[25] http://bugzilla.globus.org/globus/show_bug.cgi?id=4790
[26] http://bugzilla.globus.org/globus/show_bug.cgi?id=4817
[27] http://bugzilla.globus.org/globus/show_bug.cgi?id=4859
[28] http://bugzilla.globus.org/globus/show_bug.cgi?id=4864
[29] http://bugzilla.globus.org/globus/show_bug.cgi?id=4874

- Bug 4898:[30]Streaming problem with globusrun-ws
- Bug 4908:[31]Stage out failes in JobManager.pm
- Bug 4918:[32]user account details are cached even for unknown users
- Bug 4944:[33]Multijob resources never yield to memory pressure and can't be destroyed
- Bug 4968:[34]WS-Notification error: java.lang.ClassCastException: org.globus.exec.generated.StateChangeNotificationMessageWrapperType cannot be cast to org.w3c.dom.Element
- Bug 4981:[35]Hundreds of thousands of bogus entries in PBS server logs
- Bug 4984:[36]Change in behaviour in usage of client-generated job resource keys
- Bug 4989:[37]PBS environment var limit not tested
- Bug 5015:[38]null pointer exception when submitting job to nightly build
- Bug 5020:[39]implementation of myceil in pbs.pm
- Bug 4513:[40]LD_LIBRARY_PATH should not be set if no library_path is specified
- Bug 4550:[41]Multijob code not checking for existence of job credential
- Bug 4474:[42]globus-gridmap-and-execute problem with additional PDPs
- Bug 3529:[43]setup/postinstall fatal errors should be warnings
- Bug 3726:[44]GlobusRun error message typo
- Bug 3910:[45]Bad permissions on condor log file prevents job submissions
- Bug 4684:[46]Loading persisted jobs with expired delegation resources causes stacktraces
- Bug 5009:[47]globusrun-ws output for bad -Ft is unhelpful (part2)
- Bug 5017:[48]gram[24] tests that need to be updated
- Bug 3948:[49]Service must release all of its resources on deactivation

[30] http://bugzilla.globus.org/globus/show_bug.cgi?id=4898
[31] http://bugzilla.globus.org/globus/show_bug.cgi?id=4908
[32] http://bugzilla.globus.org/globus/show_bug.cgi?id=4918
[33] http://bugzilla.globus.org/globus/show_bug.cgi?id=4944
[34] http://bugzilla.globus.org/globus/show_bug.cgi?id=4968
[35] http://bugzilla.globus.org/globus/show_bug.cgi?id=4981
[36] http://bugzilla.globus.org/globus/show_bug.cgi?id=4984
[37] http://bugzilla.globus.org/globus/show_bug.cgi?id=4989
[38] http://bugzilla.globus.org/globus/show_bug.cgi?id=5015
[39] http://bugzilla.globus.org/globus/show_bug.cgi?id=5020
[40] http://bugzilla.globus.org/globus/show_bug.cgi?id=4513
[41] http://bugzilla.globus.org/globus/show_bug.cgi?id=4550
[42] http://bugzilla.globus.org/globus/show_bug.cgi?id=4474
[43] http://bugzilla.globus.org/globus/show_bug.cgi?id=3529
[44] http://bugzilla.globus.org/globus/show_bug.cgi?id=3726
[45] http://bugzilla.globus.org/globus/show_bug.cgi?id=3910
[46] http://bugzilla.globus.org/globus/show_bug.cgi?id=4684
[47] http://bugzilla.globus.org/globus/show_bug.cgi?id=5009
[48] http://bugzilla.globus.org/globus/show_bug.cgi?id=5017
[49] http://bugzilla.globus.org/globus/show_bug.cgi?id=3948

- Bug 3384:[50]Inconsistent jobType/count parameter semantics
- Bug 3571:[51]ant not found during install
- Bug 3575:[52]SEG dependent on GLOBUS_LOCATION env var
- Bug 3672:[53]Streaming with PBS fails
- Bug 3803:[54]Default scratchDirectory doesn't exist
- Bug 4216:[55]Empty submit_test/submitxxx.err causes FAILure in Local tests
- Bug 4452:[56]job submission response is effected by java 1.5 thread processing
- Bug 4464:[57]setting the mpirun/exec path used for mpi jobs
- Bug 4533:[58]make-test-script.pl bug
- Bug 4597:[59]Globus/Condor integration issues on HP-UX
- Bug 4719:[60]globus runs /usr/bin/env without checking for \u
- Bug 4734:[61]Missing wsa:Action for GRAM4 rendezvous register operations
- Bug 4749:[62]client receives no state notifications if RSL gram:count is big

5. For More Information

Click here[63] for more information about this component.

[50] http://bugzilla.globus.org/globus/show_bug.cgi?id=3384
[51] http://bugzilla.globus.org/globus/show_bug.cgi?id=3571
[52] http://bugzilla.globus.org/globus/show_bug.cgi?id=3575
[53] http://bugzilla.globus.org/globus/show_bug.cgi?id=3672
[54] http://bugzilla.globus.org/globus/show_bug.cgi?id=3803
[55] http://bugzilla.globus.org/globus/show_bug.cgi?id=4216
[56] http://bugzilla.globus.org/globus/show_bug.cgi?id=4452
[57] http://bugzilla.globus.org/globus/show_bug.cgi?id=4464
[58] http://bugzilla.globus.org/globus/show_bug.cgi?id=4533
[59] http://bugzilla.globus.org/globus/show_bug.cgi?id=4597
[60] http://bugzilla.globus.org/globus/show_bug.cgi?id=4719
[61] http://bugzilla.globus.org/globus/show_bug.cgi?id=4734
[62] http://bugzilla.globus.org/globus/show_bug.cgi?id=4749
[63] index.html

Chapter 8. GT 4.0.5 Incremental Release Notes: WS GRAM

1. Introduction

These release notes are for the incremental release 4.0.5. It includes a summary of changes since 4.0.4, bug fixes since 4.0.4 and any known problems that still exist at the time of the 4.0.5 release. This page is in addition to the top-level 4.0.5 release notes at http://www.globus.org/toolkit/releasenotes/4.0.5.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS GRAM 4.0 Release Notes[1].

2. Changes Summary

3. Bug Fixes

- Bug 5027:[2] more efficient processing of internal job states
- Bug 4865:[3] Events from SEG to JobStateMonitor are deleted too early i.e. jobs keep stuck
- Bug 4777:[4] Excessive memory usage for SEG modules
- Bug 4983:[5] Flag that triggers default SoftEnv keys usage in job submission
- Bug 4706:[6] Fork jobs without arguments may receive uninitialized data as arguments
- Bug 4764:[7] globusrun-ws output for bad -Ft is unhelpful.
- Bug 4732:[8] globusrun-ws segmentation fault (FC5 x86_64)
- Bug 4884:[9] GramJob post-submit subscriptions broken
- Bug 4860:[10] LSF jobmanager doesn't preserve environment variables with spaces

4. Known Problems

- Bug 4761:[11]Scheduler Tutorial is missing WS-GRAM setup package

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=5027
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=4865
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=4777
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=4983
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=4706
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=4764
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=4732
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=4884
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=4860
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=4761

- Bug 4778:[12]WS-Fork job manager doesn't set environment up for mpi jobs
- Bug 4787:[13]no lifetime management for WS Rendezvous
- Bug 4790:[14]stdout RP gets null
- Bug 4817:[15]Condor OS and ARCH do not have dynamic defaults
- Bug 4859:[16]globusrun-ws staging error after gt4 deployed to tomcat
- Bug 4864:[17]environment variables containing '=' get escaped
- Bug 4874:[18]LRUCache problem during job submission from Condor-G to GRAM4
- Bug 4898:[19]Streaming problem with globusrun-ws
- Bug 4908:[20]Stage out failes in JobManager.pm
- Bug 4918:[21]user account details are cached even for unknown users
- Bug 4944:[22]Multijob resources never yield to memory pressure and can't be destroyed
- Bug 4968:[23]WS-Notification error: java.lang.ClassCastException: org.globus.exec.generated.StateChangeNotificationMessageWrapperType cannot be cast to org.w3c.dom.Element
- Bug 4981:[24]Hundreds of thousands of bogus entries in PBS server logs
- Bug 4984:[25]Change in behaviour in usage of client-generated job resource keys
- Bug 4989:[26]PBS environment var limit not tested
- Bug 5015:[27]null pointer exception when submitting job to nightly build
- Bug 5020:[28]implementation of myceil in pbs.pm
- Bug 4513:[29]LD_LIBRARY_PATH should not be set if no library_path is specified
- Bug 4550:[30]Multijob code not checking for existence of job credential
- Bug 4474:[31]globus-gridmap-and-execute problem with additional PDPs

[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=4778
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=4787
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=4790
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=4817
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=4859
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=4864
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=4874
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=4898
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=4908
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=4918
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=4944
[23] http://bugzilla.globus.org/globus/show_bug.cgi?id=4968
[24] http://bugzilla.globus.org/globus/show_bug.cgi?id=4981
[25] http://bugzilla.globus.org/globus/show_bug.cgi?id=4984
[26] http://bugzilla.globus.org/globus/show_bug.cgi?id=4989
[27] http://bugzilla.globus.org/globus/show_bug.cgi?id=5015
[28] http://bugzilla.globus.org/globus/show_bug.cgi?id=5020
[29] http://bugzilla.globus.org/globus/show_bug.cgi?id=4513
[30] http://bugzilla.globus.org/globus/show_bug.cgi?id=4550
[31] http://bugzilla.globus.org/globus/show_bug.cgi?id=4474

- Bug 3529:[32]setup/postinstall fatal errors should be warnings
- Bug 3726:[33]GlobusRun error message typo
- Bug 3910:[34]Bad permissions on condor log file prevents job submissions
- Bug 4684:[35]Loading persisted jobs with expired delegation resources causes stacktraces
- Bug 5009:[36]globusrun-ws output for bad -Ft is unhelpful (part2)
- Bug 5017:[37]gram[24] tests that need to be updated
- Bug 3948:[38]Service must release all of its resources on deactivation
- Bug 3384:[39]Inconsistent jobType/count parameter semantics
- Bug 3571:[40]ant not found during install
- Bug 3575:[41]SEG dependent on GLOBUS_LOCATION env var
- Bug 3672:[42]Streaming with PBS fails
- Bug 3803:[43]Default scratchDirectory doesn't exist
- Bug 4216:[44]Empty submit_test/submitxxx.err causes FAILure in Local tests
- Bug 4452:[45]job submission response is effected by java 1.5 thread processing
- Bug 4464:[46]setting the mpirun/exec path used for mpi jobs
- Bug 4533:[47]make-test-script.pl bug
- Bug 4597:[48]Globus/Condor integration issues on HP-UX
- Bug 4719:[49]globus runs /usr/bin/env without checking for \u
- Bug 4734:[50]Missing wsa:Action for GRAM4 rendezvous register operations
- Bug 4749:[51]client receives no state notifications if RSL gram:count is big

[32] http://bugzilla.globus.org/globus/show_bug.cgi?id=3529
[33] http://bugzilla.globus.org/globus/show_bug.cgi?id=3726
[34] http://bugzilla.globus.org/globus/show_bug.cgi?id=3910
[35] http://bugzilla.globus.org/globus/show_bug.cgi?id=4684
[36] http://bugzilla.globus.org/globus/show_bug.cgi?id=5009
[37] http://bugzilla.globus.org/globus/show_bug.cgi?id=5017
[38] http://bugzilla.globus.org/globus/show_bug.cgi?id=3948
[39] http://bugzilla.globus.org/globus/show_bug.cgi?id=3384
[40] http://bugzilla.globus.org/globus/show_bug.cgi?id=3571
[41] http://bugzilla.globus.org/globus/show_bug.cgi?id=3575
[42] http://bugzilla.globus.org/globus/show_bug.cgi?id=3672
[43] http://bugzilla.globus.org/globus/show_bug.cgi?id=3803
[44] http://bugzilla.globus.org/globus/show_bug.cgi?id=4216
[45] http://bugzilla.globus.org/globus/show_bug.cgi?id=4452
[46] http://bugzilla.globus.org/globus/show_bug.cgi?id=4464
[47] http://bugzilla.globus.org/globus/show_bug.cgi?id=4533
[48] http://bugzilla.globus.org/globus/show_bug.cgi?id=4597
[49] http://bugzilla.globus.org/globus/show_bug.cgi?id=4719
[50] http://bugzilla.globus.org/globus/show_bug.cgi?id=4734
[51] http://bugzilla.globus.org/globus/show_bug.cgi?id=4749

5. For More Information

Click here[52] for more information about this component.

[52] index.html

Chapter 9. GT 4.0 WS GRAM : System Administrator's Guide

1. Introduction

This guide contains advanced configuration information for system administrators working with WS GRAM. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation. It also describes additional prerequisites and host settings necessary for WS GRAM operation. Readers should be familiar with the Key Concepts[1] and Implementation Approach[2] for WS GRAM to understand the motivation for and interaction between the various deployed components.

Important

The information in this WS GRAM Admin Guide is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the GT 4.0 System Administrator's Guide[3]. Read through this guide before continuing!

2. Building and Installing

WS GRAM is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide[4].

2.1. Installation Requirements

2.1.1. Transport Level Security (TLS)

In order to use WS GRAM, the container must be started with Transport Level security. The "-nosec" option should *not* be used with `globus-start-container`.

2.1.2. Functioning sudo

WS GRAM requires that the *sudo* command is installed and functioning on the service host where WS GRAM software will execute.

Authorization rules will need to be added to the `sudoers` file to allow the WS GRAM service account to execute (without a password) the *scheduler adapter* in the accounts of authorized GRAM users. For configuration details, see the Configuring sudo section.

Platform Note: On AIX, sudo is not installed by default, but it is available as source and rpm here: AIX 5L Toolbox for Linux Applications[5]

[1] http://www.globus.org/toolkit/docs/4.0/execution/key/index.html
[2] http://www.globus.org/toolkit/docs/4.0/execution/WS_GRAM_Approach.html
[3] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[4] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[5] http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html

2.1.3. Local Scheduler

WS GRAM depends on a local mechanism for starting and controlling jobs. Included in the WS GRAM software is a Fork *scheduler*, which requires no special software installed to execute jobs on the local host. However, to enable WS GRAM to execute and manage jobs to a *batch scheduler*, the scheduler software must be installed and configured prior to configuring WS GRAM.

2.1.4. Scheduler Adapter

WS GRAM depends on scheduler adapters to translate the WS GRAM *job description* document into commands understood by the local scheduler, as well as monitor the jobs.

Scheduler adapters included in the GT 4.0 release are: PBS[6], Condor[7], LSF[8]

Other third party scheduler adapters available for GT 4.0.x releases:

- Sun Grid Engine[9]
- LoadLeveler - as of release 3.3.1 IBM LoadLeveler includes a GRAM Scheduler Adapter. For more information see "What's new" in the LoadLeveler product documentation[10]
- GridWay[11] - installation and configuration guide is here[12]

For configuration details, see the Configuring scheduler adapters section.

2.1.5. GridFTP

Though staging directives are processed by RFT (see next section), RFT uses GridFTP servers underneath to do the actual data movement. As a result, *there must be at least one GridFTP server that shares a file system with the execution nodes*. There is no special process to get staged files onto the execution node before the job executable is run. See the Non-default GridFTP server section of this admin guide for details on how to configure WS GRAM for your GridFTP servers used in your execution environment.

2.1.6. Reliable File Transfer Service (RFT)

WS GRAM depends on RFT to perform file staging and cleanup directives in a job description. For configuration details, see the RFT admin guide[13] *Important:* Jobs requesting these functions will fail if RFT is not properly setup.

[6] http://www.openpbs.org/
[7] http://www.cs.wisc.edu/condor/
[8] http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/
[9] http://www.lesc.ic.ac.uk/projects/SGE-GT4.html
[10] http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html
[11] http://www.grid4utility.org/software.php
[12] http://www.grid4utility.org/documents.php
[13] http://www.globus.org/toolkit/docs/4.0/data/rft/admin-index.html

3. Configuring

3.1. Typical Configuration

3.1.1. Configuring sudo

When the credentials of the service account and the job submitter are different (multi user mode), then GRAM will prepend a call to sudo to the local adapter callout command. *Important:* If sudo is not configured properly, the command and thus job will fail.

As *root*, here are the two lines to add to the /etc/sudoers file for each GLOBUS_LOCATION installation, where /opt/globus/GT4.0.0 should be replaced with the GLOBUS LOCATION for your installation:

```
# Globus GRAM entries
               globus  ALL=(username1,username2)
               NOPASSWD: /opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute
               -g /etc/grid-security/grid-mapfile
               /opt/globus/GT4.0.0/libexec/globus-job-manager-script.pl *
               globus  ALL=(username1,username2)
               NOPASSWD: /opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute
               -g /etc/grid-security/grid-mapfile
               /opt/globus/GT4.0.0/libexec/globus-gram-local-proxy-tool *
```

The `globus-gridmap-and-execute` program is used to ensure that GRAM only runs programs under accounts that are in the grid-mapfile. In the sudo configuration, it is the first program called. It looks up the account in the grid-mapfile and then runs the requested command. It is redundant if sudo is properly locked down. This tool could be replaced with your own authorization program.

3.1.2. Configuring Scheduler Adapters

The WS GRAM scheduler adapters included in the release tarball are: *PBS*, *Condor* and *LSF*. To install, follow these steps (shown for pbs):

```
    % cd $GLOBUS_LOCATION\gt4.0.0-all-source-installer

               % make gt4-gram-pbs

               % make install
```

Using PBS as the example, make sure the scheduler commands are in your path (qsub, qstat, pbsnodes).

For PBS, another setup step is required to configure the remote shell for rsh access:

```
               % cd $GLOBUS_LOCATION/setup/globus

               % ./setup-globus-job-manager-pbs --remote-shell=rsh
```

The last thing is to define the GRAM and GridFTP file system mapping for PBS. A default mapping in this file is created to allow simple jobs to run. However, the actual file system mappings for your compute resource should be entered to ensure:

- files staging is performed correctly
- jobs with erroneous file path directives are rejected

Done! You have added the PBS scheduler adapters to your GT installation.

Note for future GT builds with scheduler adapters: scheduler adapters can be enabled by adding --enable-wsgram-pbs to the configure line when building the entire toolkit.

```
% configure --prefix=$GLOBUS_LOCATION --enable-wsgram-pbs ...
% make
% make install
```

3.2. Non-default Configuration

3.2.1. Non-default Credentials

To run the container using just a user proxy, instead of host creds, edit the `$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml` file, and either comment out the credentials section...

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
<!--
<credential>
<key-file value="/etc/grid-security/containerkey.pem"/>
<cert-file value="/etc/grid-security/containercert.pem"/>
<credential>
-->
<gridmap value="/etc/grid-security/grid-mapfile"/>
<securityConfig>
```

or replace the credentials section with a proxy file location...

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
<proxy-file value="<PATH TO PROXY FILE>"/>
<gridmap value="/etc/grid-security/grid-mapfile"/>
<securityConfig>
```

Running in personal mode (user proxy), another GRAM configuration setting is required. For GRAM to authorize the RFT service when performing staging functions, it needs to know the subject DN for verification. Here are the steps:

```
% cd $GLOBUS_LOCATION/setup/globus
```

```
% ./setup-gram-service-common --staging-subject=
"/DC=org/DC=doegrids/OU=People/CN=Stuart Martin 564720"
```

You can get your subject DN by running this command:

```
% grid-cert-info -subject
```

3.2.2. Non-default GridFTP server

By default, the GridFTP server is assumed to run as root on localhost:2811. If this is not true for your site then change it by editing the GridFTP host and/or port in the GRAM and GridFTP file system mapping config file: `$GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml`.

3.2.3. Non-default container port

By default, the globus services will assume 8443 is the port the Globus container is using. However the container can be run under a non-standard port, for example:

```
% globus-start-container -p 4321
```

3.2.4. Non-default gridmap

If you wish to specify a non-standard gridmap file in a multi-user installation, two basic configurations need to be changed:

- $GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml
  - As specified in the gridmap config[14] instructions, add a <gridmap value="..."/> element to the file appropriately.
- /etc/sudoers
  - Change the file path after all -g options

    ```
    -g /path/to/grid-mapfile
    ```

    .

Example: *global_security_descriptor.xml*

```
...
<gridmap value="/opt/grid-mapfile"/>
...
```

sudoers

[14] http://www.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html#s-authzframe-secdesc-configGridmap

```
...

# Globus GRAM entries
globus  ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute
-g /opt/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-job-manager-script.pl *
globus  ALL=(username1,username2)
NOPASSWD: /opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute
-g /opt/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-gram-local-proxy-tool *

...
```

3.2.5. Non-default RFT deployment

RFT is used by GRAM to stage files in and out of the job execution environment. In the default configuration, RFT is hosted in the same container as GRAM and is assumed to have the same service path and standard service names. This need not be the case. For example, the most likely alternative scenario is that RFT would be hosted seperately in a container on a different machine. In any case, both the RFT and the Delegation Service endpoints need to be adjustable to allow this flexibility. The following options can be passed to the *setup-gram-service-common* script to affect these settings:

```
--staging-protocol=<protocol>
--staging-host=<host>
--staging-port=<port>
--staging-service-path=<RFT and Delegation factory service path>
--staging-factory-name=<RFT factory service name>
--staging-delegation-factory-name=<name of Delegation factory service used by
```

for example

```
% setup-gram-service-common \
--staging-protocol=http
--staging-host=somemachine.fakedomain.net
--staging-port=8444
--staging-service-path=/tomcat/services/
--staging-factory-name=MyReliableFileTransferFactoryService
--staging-delegation-factory-name=MyDelegationFactoryServiceForRFT
```

will internally cause the GRAM service code to construct the following EPR addresses:

```
http://somemachine.fakedomain.net:8444/tomcat/services/MyReliableFileTransferF

http://somemachine.fakedomain.net:8444/tomcat/services/MyDelegationFactoryServ
```

3.3. Locating configuration files

All the GRAM service configuration files are located in subdirectories of the `$GLOBUS_LOCATION/etc` directory. The names of the GRAM configuration directories all start with `gram-service`. For instance, with a default GRAM installation, the command line:

```
% ls etc | grep gram-service
```

gives the following output:

```
gram-service
gram-service-Fork
gram-service-Multi
```

3.4. Web service deployment configuration

The file `$GLOBUS_LOCATION/etc/gram-service/server-config.wsdd` contains information necessary to deploy and instantiate the GRAM services in the Globus container.

Three GRAM services are deployed:

- ManagedExecutableJobService: service invoked when querying or managing an *executable job*
- ManagedMultiJobService: service invoked when querying or managing a *multijob*
- ManagedJobFactoryService: service invoked when submitting a job

Each service deployment information contains the name of the Java service implementation class, the path to the WSDL service file, the name of the operation providers that the service reuses for its implementation of WSDL-defined operations, etc. More information about the service deployment configuration information can be found here[15].

3.5. JNDI application configuration

The configuration of WSRF resources and application-level service configuration not related to service deployment is contained in JNDI[16] files. The JNDI-based GRAM configuration is of two kinds:

3.5.1. Common job factory configuration

The file `$GLOBUS_LOCATION/etc/gram-service/jndi-config.xml` contains configuration information that is common to every local resource manager.

More precisely, the configuration data it contains pertains to the implementation of the GRAM WSRF resources (factory resources and job resources), as well as initial values of WSRF resource properties that are always published by any Managed Job Factory WSRF resource.

The data is categorized by service, because according to WSRF, in spite of the service/resource separation of concern, a given service will use only one XML Schema type of resource. In practice it is therefore clearer to categorize the

[15] http://www.globus.org/toolkit/docs/4.0/common/javawscore/Java_WS_Core_Public_Interfaces.html#config
[16] http://java.sun.com/products/jndi/

configuration resource implementation by service, even if theoretically speaking a given resource implementation could be used by several services. For more information, refer to the Java WS Core documentation[17].

Here is the decomposition, in JNDI objects, of the common configuration data, categorized by service. Each XYZHome object contains the same Globus Core-defined information for the implementation of the WSRF resource, such as the Java implementation class for the resource (`resourceClass` datum), the Java class for the resource key (`resourceKeyType` datum), etc.

- ManagedExecutableJobService
  - ManagedExecutableJobHome: configuration of the implementation of resources for the service.
- ManagedMultiJobService
  - ManagedMultiJobHome: configuration of the implementation of resources for the service
- ManagedJobFactoryService
  - FactoryServiceConfiguration: this encapsulates configuration information used by the factory service. Currently this identifies the service to associate to a newly created job resource in order to create an endpoint reference and return it.
  - ManagedJobFactoryHome: implementation of resources for the service resourceClass
  - FactoryHomeConfiguration: this contains GRAM application-level configuration data i.e. values for resource properties common to all factory resources. For instance, the path to the Globus installation, host information such as CPU type, manufacturer, operating system name and version, etc.

3.5.2. Local resource manager configuration

When a SOAP call is made to a GRAM factory service in order to submit a job, the call is actually made to a GRAM service-resource pair, where the factory resource represents the local resource manager to be used to execute the job.

There is one directory `gram-service-<manager>/` for each local resource manager supported by the GRAM installation.

For instance, let's assume the command line:

```
% ls etc | grep gram-service-
```

gives the following output:

```
gram-service-Fork
gram-service-LSF
gram-service-Multi
```

In this example, the Multi, Fork and *LSF* job factory resources have been installed. `Multi` is a special kind of local resource manager which enables the GRAM services to support multijobs[18].

[17] http://www.globus.org/toolkit/docs/4.0/common/javawscore/index.html
[18] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/user-index.html#s-wsgram-user-specifyingmultijob

The JNDI configuration file located under each manager directory contains configuration information for the GRAM support of the given local resource manager, such as the name that GRAM uses to designate the given resource manager. This is referred to as the *GRAM name* of the local resource manager.

For instance, `$GLOBUS_LOCATION/etc/gram-service-Fork/jndi-config.xml` contains the following XML element structure:

```
    <service name="ManagedJobFactoryService">
        <!-- LRM configuration:  Fork -->
        <resource
            name="ForkResourceConfiguration"
            type="org.globus.exec.service.factory.FactoryResourceConfiguration">
            <resourceParams>
                [...]
                <parameter>
                    <name>
                        localResourceManagerName
                    </name>
                    <value>
                        Fork
                    </value>
                </parameter>
                <!-- Site-specific scratchDir
Default: ${GLOBUS_USER_HOME}/.globus/scratch
                <parameter>
                    <name>
                        scratchDirectory
                    </name>
                    <value>
                        ${GLOBUS_USER_HOME}.globus/scratch
                    </value>
                </parameter>
                -->
            </resourceParams>
        </resource>
    </service>
```

In the example above, the name of the local resource manager is `Fork`. This value can be used with the GRAM command line client in order to specify which factory resource to use when submitting a job. Similarly, it is used to create an endpoint reference to the chosen factory WS-Resource when using the GRAM client API.

In the example above, the *scratchDirectory* is set to `${GLOBUS_USER_HOME}/.globus/scratch`. This is the default setting. It can be configured to point to an alternate file system path that is common to the compute cluster and is typically less reliable (auto purging), while offering a greater amount of disk space (thus "scratch").

3.6. Security descriptor

The file `$GLOBUS_LOCATION/etc/gram-service/managed-job-factory-security-config.xml` contains the Core security configuration for the GRAM `ManagedJobFactory` service:

- default security information for all remote invocations, such as:
    - the authorization method, based on a Gridmap file (in order to resolve user credentials to local user names)

    - limited proxy credentials will be rejected

- security information for the `createManagedJob` operation

The file `$GLOBUS_LOCATION/etc/gram-service/managed-job-security-config.xml` contains the Core security configuration for the GRAM job resources:

- The default is to only allow the identity that called the createManagedJob operation to access the resource.

Note, that by default two gridmap checks are done during a invocation of WS-GRAM:

1. One gridmap check is be done by the container as configured by the gridmap element in `$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml`

2. Another check is done by WS-GRAM when it calls the Perl modules which are used for job submission to the underlying local resource manager, as configured by the `authz` element which is by default set to gridmap in `$GLOBUS_LOCATION/etc/gram-service/managed-job-factory-security-config.xml` and `$GLOBUS_LOCATION/etc/gram-service/managed-job-security-config.xml`. This check is done for additional security reasons to make sure that a potentially hacked globus user account still only can act on behalf of the users which are defined in a grid-mapfile.

The second gridmap check can be avoided by removing the `authz` element from both WS-GRAM security descriptors. This however does not mean, that no authorization check is done. The container still checks if the client is authorized as defined in `$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml` but there's no further authorization check before calling the Perl modules. It's up to the GT4 container administrator to decide whether he wants to have that additional authorization check or not.

Note: GRAM does not override the container security credentials defined in `$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml`. These are the credentials used to authenticate all service requests.

3.7. GRAM and GridFTP file system mapping

The file `$GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml` contains information to associate local resource managers with GridFTP servers. GRAM uses the GridFTP server (via RFT) to perform all file staging directives. Since the GridFTP server and the Globus service container can be run on separate hosts, a mapping is needed between the common file system paths of these 2 hosts. This enables the GRAM services to resolve file:/// staging directives to the local GridFTP URLs.

Below is the default Fork entry. Mapping a jobPath of / to ftpPath of / will allow any file staging directive to be attempted.

```
<map>
    <scheduler>Fork</scheduler>
    <ftpServer>
       <protocol>gsiftp</protocol>
       <host>myhost.org</host>
       <port>2811</port>
    </ftpServer>
    <mapping>
       <jobPath>/</jobPath>
       <ftpPath>/</ftpPath>
    </mapping>
</map>
```

For a *scheduler*, where jobs will typically run on a compute node, a default entry is not provided. This means staging directives will fail until a mapping is entered. Here is an example of a compute cluster with *PBS* installed that has 2 common mount points between the front end host and the GridFTP server host.

```
<map>
    <scheduler>PBS</scheduler>
    <ftpServer>
       <protocol>gsiftp</protocol>
       <host>myhost.org</host>
       <port>2811</port>
    </ftpServer>
    <mapping>
       <jobPath>/pvfs/mount1/users</jobPath>
       <ftpPath>/pvfs/mount2/users</ftpPath>
    </mapping>
    <mapping>
       <jobPath>/pvfs/jobhome</jobPath>
       <ftpPath>/pvfs/ftphome</ftpPath>
    </mapping>
</map>
```

The file system mapping schema doc is here[19].

3.8. Scheduler-Specific Configuration Files

In addition to the service configuration described above, there are scheduler-specific configuration files for the Scheduler Event Generator modules. These files consist of name=value pairs separated by newlines. These files are:

[19] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_fs_map.html

Table 9.1. Scheduler-Specific Configuration Files

$GLOBUS_LOCATION/etc/globus-fork.conf	Configuration for the Fork *SEG* module implementation. The attributes names for this file are: log_path — Path to the SEG Fork log (used by the globus-fork-starter and the SEG). The value of this should be the path to a world-writable file. The default value for this created by the Fork setup package is $GLOBUS_LOCATION/var/globus-fork.log. This file must be readable by the account that the SEG is running as.
$GLOBUS_LOCATION/etc/globus-condor.conf	Configuration for the *Condor* SEG module implementation. The attributes names for this file are: log_path — Path to the SEG Condor log (used by the Globus::GRAM::JobManager::condor perl module and Condor SEG module. The value of this should be the path to a world-readable and world-writable file. The default value for this created by the Fork setup package is $GLOBUS_LOCATION/var/globus-condor.log
$GLOBUS_LOCATION/etc/globus-pbs.conf	Configuration for the PBS SEG module implementation. The attributes names for this file are: log_path — Path to the SEG PBS logs (used by the Globus::GRAM::JobManager::pbs perl module and PBS SEG module. The value of this should be the path to the directory containing the server logs generated by PBS. For the SEG to operate, these files must have file permissions such that the files may be read by the user the SEG is running as.
$GLOBUS_LOCATION/etc/globus-lsf.conf	Configuration for the LSF SEG module implementation. The attributes names for this file are: log_path — Path to the SEG LSF log directory. This is used by the LSF SEG module. The value of this should be the path to the directory containing the server logs generated by LSF. For the SEG to operate, these files must have file permissions such that the files may be read by the user the SEG is running as.

3.9. WS GRAM auto-registration with default WS MDS Index Service

With a default GT 4.0.1 installation, the WS GRAM service is automatically registered with the default WS MDS Index Service[20] running in the same container for monitoring and discovery purposes.

Note

If you are using GT 4.0.0, we strongly recommend upgrading to 4.0.1 to take advantage of this capability.

[20] http://www.globus.org/toolkit/docs/4.0/info/index/

However, if must use GT 4.0.0, or if this registration was turned off and you want to turn it back on, this is how it is configured:

There is a jndi resource defined in `$GLOBUS_LOCATION/etc/gram-service/jndi-config.xml` as follows :

```
<resource name="mdsConfiguration"

type="org.globus.wsrf.impl.servicegroup.client.MDSConfiguration">
<resourceParams>
<parameter>
<name>reg</name>
<value>true</value>
</parameter>
<parameter>
<name>factory</name>
<value>org.globus.wsrf.jndi.BeanFactory</value>
</parameter>
</resourceParams>
</resource>
```

To configure the automatic registration of WS GRAM to the default WS MDS Index Service, change the value of the parameter `<reg>` as follows:

- `true` turns on auto-registration; this is the default in GT 4.0.1.
- `false` turns off auto-registration; this is the default in GT 4.0.0.

3.9.1. Configuring resource properties

By default, the `GLUECE:` resource property (which contains GLUE data) is sent to the default Index Service:

You can configure which resource properties are sent in WS GRAM's registration.xml file, `$GLOBUS_LOCATION/etc/gram-service/registration.xml`. The following is the relevant section of the file (as it is set by default):

```
<Content xsi:type="agg:AggregatorContent"
xmlns:agg="http://mds.globus.org/aggregator/types">

<agg:AggregatorConfig xsi:type="agg:AggregatorConfig">

<agg:GetResourcePropertyPollType
    xmlns:glue="http://mds.globus.org/glue/ce/1.1">
<!-- Specifies that the index should refresh information
every 60000 milliseconds (once per minute) -->
<agg:PollIntervalMillis>60000</agg:PollIntervalMillis>

<!-- specifies the resource property that should be
aggregated, which in this case is the GLUE cluster
and scheduler information RP -->

<agg:ResourcePropertyName>glue:GLUECE</agg:ResourcePropertyName>
```

```
</agg:GetResourcePropertyPollType>
</agg:AggregatorConfig>
<agg:AggregatorData/>
</Content>
```

3.10. Registering WS GRAM manually with default WS MDS Index Service

If a third party needs to register an WS GRAM service manually, see Registering with mds-servicegroup-add[21] in the WS MDS Aggregator Framework documentation.

3.11. Configuring support for SoftEnv

Note: This feature is only available beginning from version 4.0.5 of the toolkit.

3.11.1. Overview

SoftEnv is a system designed to make it easier for users to define what applications they want to use, and easier for administrators to make applications available to users. SoftEnv has evolved from the original implementation called Soft designed at Northeastern University in 1994.

In some environments like TeraGrid it's desirable to make use of SoftEnv before a job is submitted to leverage the use of an exactly defined software environment the job will run in.

3.11.2. Configuration

Because this feature is very specific and may not be available on many systems, support for SoftEnv is disabled by default in normal job submissions. There is a parameter in the JNDI configuration of WS GRAM to enable SoftEnv support in job submissions.

SoftEnv support must be enabled on a per-scheduler basis because the internal mechanisms to support SoftEnv vary between the different types of schedulers. Currently only the scheduler Fork, PBS and LSF can be configured to have SoftEnv support enabled, Condor not yet. To enable this feature the parameter 'enableDefaultSoftwareEnvironment' in the scheduler specific JNDI configuration must be set to 'true'. For example to enable SoftEnv support in the Fork scheduler, set the 'enableDefaultSoftwareEnvironment' in $GLOBUS_LOCATION/etc/gram-service-Fork/jndi-config.xml to 'true'.

Enabled SoftEnv support means that a users default environment will be created from his `.soft` file before each job submission automatically. The user doesn't need to provide extra SoftEnv keys in the extensions element of a job description. This is not done if the SoftEnv feature is disabled.

For more information and examples, please look in the SoftEnv section of the user guide[22].

3.11.3. Dependencies

For the scheduler Fork SoftEnv needs to be installed on the host the container is running on. For PBS and LSF SoftEnv needs to be installed on the hosts where the jobs are executed.

[21] http://www.globus.org/toolkit/docs/4.0/info/aggregator/re01.html#mds-servicegroup-add-registering
[22] user-index.html#s-wsgram-user-softenv

3.12. Job Description Document Substitution Variables

By default only four variables can be used in the job description document which are resolved to values in the service. These are

- GLOBUS_USER_HOME
- GLOBUS_USER_NAME
- GLOBUS_SCRATCH_DIR
- GLOBUS_LOCATION

3.12.1. Changes in WS GRAM beginning from GT version 4.0.5

To enable communities to define their own system-wide variables and enable their users to use them in their job descriptions, a new generic variable/value config file was added where these variables can be defined. If a job description document contains one of these variables that file will be used to resolve any matching variables.

A new service parameter in the JNDI container registry defines the path to the variable mapping file. The mapping is done for each scheduler. This file is checked periodically (configurable frequency) to see if it has changed. If so, it is reread and the new content replaces the old.

For example for the scheduler Fork there are the following entries in `$GLOBUS_LOCATION/etc/gram-service-Fork/jndi-config.xml` which can be configured to determine the location and the refresh period of the variable mapping file:

```
<parameter>
  <name>
    substitutionDefinitionsFile
  </name>
  <value>
    /root/vdt-stuff/globus/etc/gram-service-Condor/substitution definition.properties
  </value>
</parameter>
<parameter>
  <name>
    substitutionDefinitionsRefreshPeriod
  </name>
  <value>
    <!-- MINUTES -->
    480
  </value>
</parameter>
```

The use of variables in the job description document that are *not* defined in the variable mapping file leads to the following error during job submission: `'No value found for RSL substitution variable <variableName>'`

3.13. Audit Logging

Note: This feature is only available beginning from version 4.0.5 of the toolkit.

3.13.1. Overview

WS-GRAM provides mechanisms to provide access to audit and accounting information associated with jobs that are submitted to local resource manager (LRM) like PBS, LSF, Condor by WS-GRAM. GRAM is not a local resource manager but rather a protocol engine for communicating with a range of different local resource managers using a standard message format. In some scenarios it is desirable to get an overview over the usage of the underlying LRM like

- What kind of jobs had been submitted via GRAM?
- How long did the processing of a job take?
- How many jobs had been submitted by user X?

The following three usecases give a better overview about the meaning and purpose of auditing and accounting:

1. **Group Access**. A grid resource provider allows a remoteservice (e.g., a gateway or portal) to submit jobs on behalf of multiple users. The grid resource provider only obtains information about the identity of the remote submitting service and thus does not know the identity of the users for which the grid jobs are submitted. This group access is allowed under the condition that the remote service store audit information so that, if and when needed, the grid resource provider can request and obtain information to track a specific job back to an individual user.
2. **Query Job Accounting**. A client that submits a job needs to be able to obtain, after the job has completed, information about the resources consumed by that job. In portal and gateway environments where many users submit many jobs against a single allocation, this per-job accounting information is needed soon after the job completes so that client-side accounting can be updated. Accounting information is sensitive and thus should only be released to authorized parties.
3. **Auditing**. In a distributed multi-site environment, it can be necessary to investigate various forms of suspected intrusion and abuse. In such cases, we may need to access an audit trail of the actions performed by a service. When accessing this audit trail, it will frequently be important to be able to relate specific actions to the user.

The audit record of each job is stored in a DBMS and contains

- *job_grid_id*: String representation of the resource EPR
- *local_job_id*: Job/process id generated by the scheduler
- *subject_name*: Distinguished name (DN) of the user
- *username*: Local username
- *idempotence_id*: Job id generated on the client-side
- *creation_time*: Date when the job resource is created
- *queued_time*: Date when the job is submitted to the scheduler
- *stage_in_grid_id*: String representation of the stageIn-EPR (RFT)
- *stage_out_grid_id*: String representation of the stageOut-EPR (RFT)
- *clean_up_grid_id*: String representation of the cleanUp-EPR (RFT)
- *globus_toolkit_version*: Version of the server-side GT
- *resource_manager_type*: Type of the resource manager (Fork, Condor, ...)

- *job_description*: Complete job description document
- *success_flag*: Flag that shows whether the job failed or finished successfully
- *finished_flag*: Flag that shows whether the job is already fully processed or still in progress

While audit and accounting records may be generated and stored by different entities in different contexts, we assume here that audit records are generated by the GRAM service itself and accounting records by the LRM to which the GRAM service submits jobs. Accounting records could contain all information about the duration and the resource-usage of a job. Audit records are stored in a database indexed by a Grid job identifier (GJID), while accounting records are maintained by the LRM indexed by a local job identifier (JID).

GRAM Service GJID creation

The WS-GRAM service returns an EPR that is used to control the job. The EPR is an XML document and cannot effectively be used as a primary key for a database table. It needs to be converted from an EPR to an acceptable GJID format. A utility class EPRUtil.java is included GT releases beginning from version 4.0.5 and can be used by both the GRAM service before storing the audit record and the GRAM client before getting audit information from the audit database.

To connect the two sets of records, both audit and accounting records, we require that GRAM records theJID in each audit record that it generates. It is then straightforward for an audit service to respond to requests like 'give me the charge of the job with JID x' by first selecting matching record(s) from the audit table and then using the local JID(s) to join to the accounting table of the LRM to access relevant accounting record(s).

We propose a Web Service interface for accessing audit and accounting information. OGSA-DAI is a WSRF service that can create a single virtual database from two or more remote databases. In the future, other per-job information like job performance data could be stored using the GJID or local JID as an index, and then made available in the same virtual database. The rest of this chapter focuses on how to configure WS-GRAM to enable Audit-Logging. A case study for TeraGrid can be read here[23]

OGSA-DAI is available here: http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/[24]

Audit logging in WS-GRAM is done 3 times in a job's lifecycle: When the processing starts, when the job is submitted to the local resource manager and when it's fully processed or when it fails.

More information about how to use this data to get e.g. accounting information of a job, how to query that audit database for information via a Web Services interface etc. please go here[25]

3.13.2. Configuration

3.13.2.1. Log4J

Add the following lines to the Log4j configuration in `$GLOBUS_LOCATION/etc/container.log4j.properties` to enable audit logging:

```
# GRAM AUDIT
log4j.category.org.globus.exec.service.exec.StateMachine.audit=DEBUG, AUDIT
log4j.appender.AUDIT=org.globus.exec.utils.audit.AuditDatabaseAppender
log4j.appender.AUDIT.layout=org.apache.log4j.PatternLayout
log4j.additivity.org.globus.exec.service.exec.StateMachine.audit=false
```

[23] http://www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit
[24] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/
[25] http://www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit

3.13.2.2. Database

Audit records are stored in a database which must be set up once. Currently we provide schemas for

- MySQL (`$GLOBUS_LOCATION/share/gram-service/gram_audit_schema_mysql.sql`)
- Postgres (`$GLOBUS_LOCATION/share/gram-service/gram_audit_schema_postgres-8.0.sql`)

The following describes how to set up the database for audit records in MySQL:

1. Create a database inside of MySQL
2. Grant necessary privileges to the user who is configured in the JNDI registry of WS-GRAM
3. Use the schema to create the table

```
host:~ feller$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.0.37 MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database auditDatabase;
Query OK, 1 row affected (0.09 sec)

mysql> GRANT ALL ON auditDatabase.* to globus@localhost identified by "foo";
Query OK, 0 rows affected (0.32 sec)

mysql> exit
Bye
host:~ feller$ mysql -u globus -p auditDatabase < ${GLOBUS_LOCATION}/share/gram-service/gr
Enter password:
host:~ feller$
```

3.13.2.3. JNDI

Add or modify the database configuration where the audit records are stored in `$GLOBUS_LOCATION/etc/gram-service/jndi-config.xml` :

```
<resource name="auditDatabaseConfiguration" type="org.globus.exec.service.utils.AuditDatab
  <resourceParams>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrf.jndi.BeanFactory</value>
    </parameter>
    <parameter>
      <name>driverClass</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:mysql://<host>[:port]/auditDatabase</value>
    </parameter>
```

```
    <parameter>
      <name>user</name>
      <value>globus</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>foo</value>
    </parameter>
    <parameter>
      <name>globusVersion</name>
      <value>4.0.3</value>
    </parameter>
  </resourceParams>
</resource>
```

3.13.3. Dependencies

Database

Currently database schemas for PostgreSQL and MySQL are provided to create the audit database table.

4. Deploying

WS GRAM is deployed as part of a standard toolkit installation. Please refer to the GT 4.0 System Administrator's Guide[26] for details.

4.1. Deploying in Tomcat

WS GRAM has been tested to work without any additional setup steps when deployed into Tomcat. Please see the Java WS Core admin guide section on deploying GT4 services into Tomcat[27] for instructions. Also, for details on tested containers, see the WS GRAM release notes[28].

Note

Currently only a single deployment is supported because of a limitation in the execution of the Scheduler Event Generator. One must set GLOBUS_LOCATION before starting Tomcat.

5. Job Description Extensions Support

Note

This feature has been added in GT 4.0.5. For versions older than 4.0.5 an update package is available to upgrade one's installation. See the downloads[29] page for the latest links.

The WS-GRAM job description schema includes a section for extending the job description with custom elements. To make sense of this in the resource manager adapter Perl scripts, a Perl module named Globus::GRAM::ExtensionsHandler is provided to turn these custom elements into parameters that the adapter scripts can understand.

[26] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[27] http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying
[28] WS_GRAM_Release_Notes.html#s-wsgram-Release_Notes-testedplatforms
[29] http://www.globus.org/toolkit/downloads/development/

It should be noted that although non-GRAM XML elements only are allowed in the `<extensions>` element of the job description, the extensions handler makes no distinction based on namespace. Thus, `<foo:myparam>` and `<bar:myparam>` will both be treated as just `<myparam>`.

Familiarity with the adapter scripts is assumed in the following sub-sections.

5.1. Requirements for Extensions Support

- XML::Parser Perl module

5.2. Supported Extension Constructs

5.2.1. Simple String Parameters

Simple string extension elements are converted into single-element arrays with the name of the unqualified tag name of the extension element as the array's key name in the Perl job description hash. Simple string extension elements can be considered a special case of the string array construct in the next section.

For example, adding the following element to the `<extensions>` element of the job description as follows:

```
<extensions>
    <myparam>yahoo!</myparam>
</extensions>
```

will cause the `$description->myparam()` to return the following value:

```
'yahoo!'
```

5.2.2. String Array Parameters

String arrays are a simple iteration of the simple string element construct. If you specify more than one simple string element in the job description, these will be assembled into a multi-element array with the unqualified tag name of the extension elements as the array's key name in the Perl job description hash.

For example:

```
<extensions>
    <myparams>Hello</myparams>
    <myparams>World!</myparams>
</extensions>
```

will cause the `$description->myparams()` to return the following value:

```
[ 'Hello', 'World!' ]
```

5.2.3. Name/Value Parameters

Name/value extension elements can be thought of as string arrays with an XML attribute 'name'. This will cause the creation of a two-dimensional array with the unqualified extension element tag name as the name of the array in the Perl job description hash.

For example:

```
<extensions>
    <myvars name="pi">3.14159</myvars>
    <myvars name="mole">6.022 x 10^23</myvars>
</extensions>
```

will cause the `$description->myvars()` to return the following value:

```
[ [ 'pi', '3.14159'], ['mole', '6.022 x 10^23'] ]
```

5.2.4. PBS Node Selection Parameters

In addition to the `globus_gram_job_manager` update package, the `globus_gram_job_manager_setup_pbs` update package is required to take advantage of the PBS node selection extensions.

Node selection constraints in PBS can be specified in two ways, generally using a construct intended to eventually apply to all resource managers which support node selection, or explicitly by sepcifiying a simple string element. The former will be more portable, but the later will appeal to those familiar with specifying node constraints for PBS jobs.

To specify PBS node selection constraints explicitly, one can simply constuct a single, simple string extension element named `nodes` with a value that conforms to the `#PBS -l nodes=...` PBS job description directive. The Globus::GRAM::ExtensionsHandler module will make this available to the PBS adapter script by invoking `$description->{nodes}`. The updated PBS adapter package checks for this value and will create a directive in the PBS job description using this value.

To use the generic construct for specifying node selection constraints, use the `resourceAllocationGroup` element:

```
<extensions>
    <resourceAllocationGroup>
        <!-- Optionally select hosts by type and number... -->
        <hostType>...</hostType>
        <hostCount>...</hostCount>

        <!-- *OR* by host names -->

        <hostName>...</hostName>
        <hostName>...</hostName>
        . . .


        <!-- With a total CPU count for this group... -->
        <cpuCount>...</cpuCount>

        <!-- *OR* an explicit number of CPUs per node... -->
        <cpusPerHost>...</cpusPerHost>
        . . .


        <!-- And a total process count for this group... -->
        <processCount>...</processCount>
```

```
        <!-- *OR* an explicit number of processes per node... -->
        <processesPerHost>...</processesPerHost>
    </resourceAllocationGroup>
</extensions>
```

Extension elements specified according to the above pseudo-schema will be converted to an appropriate `nodes` parameter which will be treated as if an explicit `nodes` extension element were specified. Multiple `resourceAllocationGroup` elements may be specified. This will simply append the constraints to the `nodes` paramater with a '+' separator. Note that one cannot specify both hostType/hostCount and hostName elements. Similarly, one cannot specify both processCount and processesPerHost elements.

Here are some examples of using `resourceAllocationGroup`:

```
<!-- #PBS -l nodes=1:ppn=10 -->
<!-- 10 processes -->
<extensions>
    <resourceAllocationGroup>
        <cpuCount>10</cpuCount>
        <processCount>10</processCount>
    </resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=activemural:ppn=10+5:ia64-compute:ppn=2 -->
<!-- 1 process (process default) -->
<extensions>
    <resourceAllocationGroup>
        <hostType>activemural</hostType>
        <cpuCount>10</cpuCount>
    </resourceAllocationGroup>
    <resourceAllocationGroup>
        <hostType>ia64-compute</hostType>
        <hostCount>5</hostCount>
        <cpusPerHost>2</cpusPerHost>
    </resourceAllocationGroup>
</extensions>

<!-- #PBS -l nodes=vis001:ppn=5+vis002:ppn=5+comp014:ppn=2+comp015:ppn=2 -->
<!-- 15 total processes -->
<extensions>
    <resourceAllocationGroup>
        <hostName>vis001</hostName>
        <hostName>vis002</hostName>
        <cpuCount>10</cpuCount>
        <processesPerHost>5</processesPerHost>
    </resourceAllocationGroup>
    <resourceAllocationGroup>
        <hostName>comp014</hostName>
        <hostName>comp015</hostName>
        <cpusPerHost>2</cpusPerHost>
        <processCount>5</processCount>
```

```
    </resourceAllocationGroup>
</extensions>
```

5.3. Customizing Extensions Support

Two Perl modules will have to be edited to customize extensions support. The first is `ExtensionsHandler.pm`. This is where the WS-GRAM job description XML of the `extensions` element is parsed and entries are added or appended to the Perl job description hash. The second module that needs to be edited is the particular resource manager adapter module that will use any new hash entries to either alter it's behavior or create additional parameters in the resource manager job description.

5.3.1. Customizing ExtensionsHandler.pm

For starters, this module logs various things to the log file specified in the `logfile` extension element. If you place this element at the start of the extensions you are creating support for, then you can look at the specified log file to get some idea of what the handler is doing. You can add new logging lines by using the `$self->log()` function. This simply takes a string that gets appended to the log file with a prefix of "`<date string> EXTENSIONS HANDLER:`".

There are three main subroutines that are used to handle parsing events and process them accordingly: `Char()`, `StartTag()`, and `EndTag()`. More handlers can be specified for other specific events when creating the `XML::Parser` instance in `new()` (see the XML::Parser[30] documentation for details). Descriptions of what the three main subroutines do currently are layed out bellow. Modify the subroutines as neccesary to achieve your goal.

`Char()` doesn't do anything but collect CDATA found between the current element start and end tags. You can access the CDATA for the current element by using `$self->{CDATA}`.

`StartTag()` is responsible for collecting the attributes associated with the element. It also increments the counter which keeps track of the number of child elements to the current extension element, and pushes the current element name onto the `@scope` queue for later use.

`EndTag()` is used for taking the CDATA collected by Char() and creating new Perl job description hash entries. This is most likely where you will need to do most of your work when adding support for new extension elements. Two useful variables are `$currentScope` and `$parentScope`. These indicate the current element that is being parsed and the parent of the element being parsed respectively. This is useful for establishing a context from which to work from. The `@scope` queue is poped at the end of this subroutine.

5.3.2. Customizing the Adapter Module

There is not much to say here. Each adapter is different. Spend some time trying to understand what the adapter does and then make and test your changes. Any new hash entries you created in `ExtensionsHandler.pm` can be accessed by calling `$description->entryname()`, where 'entryname' is the name of the entry that was added. See the construct documentation above for more details.

6. Testing

See the WS GRAM User's Guide[31] for information about submitting a test job.

[30] http://search.cpan.org/~coopercl/XML-Parser-2.31/Parser.pm
[31] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/user-index.html#s-wsgram-user-usagescenarios

7. Security Considerations

No special security considerations exist at this time.

8. Troubleshooting

When I submit a streaming or staging job, I get the following error: ERROR service.TransfreWork Terminal transfer error: [Caused by: Authentication failed[Caused by: Operation unauthorized(Mechanism le vel: Authorization failed. Expected"/CN=host/localhost.localdomain" target but r eceived "/O=Grid/OU=GlobusTest/OU=simpleCA-my.machine.com/CN=host/my.machine.com ")

- Check $GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml for the use of "localhost" or "127.0.0.1" instead of the public hostname (in the example above, "my.machine.com"). Change these uses of the loopback hostname or IP to the public hostname as neccessary.

Fork jobs work fine, but submitting PBS jobs with globusrun-ws hangs at "Current job state: Unsubmitted"

- Make sure the the log_path in $GLOBUS_LOCATION/etc/globus-pbs.conf points to locally accessible scheduler logs that are readable by the user running the container. The Scheduler Event Generator (SEG) will not work without local scheduler logs to monitor. This can also apply to other resource managers, but is most comonly seen with PBS.

- If the SEG configuration looks sane, try running the SEG tests. They are located in $GLOBUS_LOCATION/test/globus_scheduler_event_generator_*_test/. If Fork jobs work, you only need to run the PBS test. Run each test by going to the associated directory and run ./TESTS.pl. If any tests fail, report this to the gram-dev@globus.org mailing list.

- If the SEG tests succeed, the next step is to figure out the ID assigned by PBS to the queued job. Enable GRAM debug logging by uncommenting the appropriate line in the $GLOBUS_LOCATION/container-log4j.properties configuration file. Restart the container, run a PBS job, and search the container log for a line that contains "Received local job ID" to obtain the local job ID.

- Once you have the local job ID you can check the latest PBS logs pointed to by the value of "log_path" in $GLOBUS_LOCATION/etc/globus-pbs.conf to make sure the job's status is being logged. If the status is not being logged, check the documentation for your flavor of PBS to see if there's any futher configuration that needs to be done to enable job status logging. For example, PBS Pro requires a sufficient -e <bitmask> option added to the pbs_server command line to enable enough logging to satisfy the SEG.

- If the correct status is being logged, try running the SEG manually to see if it is reading the log file properly. The general form of the SEG command line is as follows: `$GLOBUS_LOCATION/libexec/globus-scheduler-event-generator -s pbs -t <timestamp>` The timestamp is in seconds since the epoch and dictates how far back in the log history the SEG should scan for job status events. The command should hang after dumping some status data to stdout. If no data appears, change the timestamp to an earlier time. If nothing ever appears, report this to the gram-user@globus.org mailing list.

- If running the SEG manually succeeds, try running another job and make sure the job process actually finishes and PBS has logged the correct status before giving up and cancelling globusrun-ws. If things are still not working, report your problem and exactly what you have tried to remedy the situtation to the gram-user@globus.org mailing list.

The job manager detected an invalid script response

- Check for a restrictive umask. When the service writes the native scheduler *job description* to a file, an overly restrictive umask will cause the permissions on the file to be such that the submission script run through *sudo* as the user cannot read the file (bug #2655).

When restarting the container, I get the following error: Error getting delegation resource

- Most likely this is simply a case of the delegated credential expiring. Either refresh it for the affected job or destroy the job resource.

The user's home directory has not been determined correctly

- This occurs when the administrator changed the location of the users's home directory and did not restart the GT4 container afterwards. Beginning from version 4.0.3 of the GT, WS-GRAM determines a user's home directory only once in the lifetime of a container (when the user submits the first job). Subsequently submitted jobs will use the cached home directory during job execution.

9. Usage statistics collection by the Globus Alliance

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 (i.e. when Done or Failed state is entered).

- job creation timestamp (helps determine the rate at which jobs are submitted)
- *scheduler* type (Fork, *PBS*, *LSF*, *Condor*, etc...)
- jobCredentialEndpoint present in *RSL* flag (to determine if server-side user proxies are being used)
- fileStageIn present in RSL flag (to determine if the staging in of files is used)
- fileStageOut present in RSL flag (to determine if the staging out of files is used)
- fileCleanUp present in RSL flag (to determine if the cleaning up of files is used)
- CleanUp-Hold requested flag (to determine if streaming is being used)
- job type (Single, Multiple, MPI, or Condor)
- gt2 error code if job failed (to determine common scheduler script errors users experience)
- fault class name if job failed (to determine general classes of common faults users experience)

If you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration[32] for instructions.

Also, please see our policy statement[33] on the collection of usage statistics.

[32] http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets
[33] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 10. GT 4.0 WS GRAM: User's Guide

1. Introduction

GRAM services provide secure job submission to many types of *job schedulers* for users who have the right to access a job hosting resource in a Grid environment. The existence of a valid proxy is in fact required for job submission. All GRAM job submission options are supported transparently through the embedded request document input. In fact, the job startup is done by submitting a client-side provided *job description* to the GRAM services. This submission can be made by end-users with the GRAM command-line tools.

2. New Functionality

2.1. Submission ID

i A submission ID may be used in the GRAM protocol for reliability in the face of message faults or other transient errors in order to ensure that at most one instance of a job is executed, i.e. to prevent accidental duplication of jobs under rare circumstances with client retry on failure. By default, the *globusrun-ws* program will generate a submission ID (*uuid*). One can override this behavior by supplying a submission ID as a command line argument.

If a user is unsure whether a job was submitted successfully, he should resubmit using the same ID as was used for the previous attempt.

2.2. Job hold and release

It is possible to specify in a job description that the job be put on hold when it reaches a chosen state (see GRAM Approach[1] documentation for more information about the executable job state machine, and see the job description XML schema documentation[2]for information about how to specify a held state). This is useful for instance when a GRAM client wishes to directly access output files written by the job (as opposed to waiting for the stage-out step to transfer files from the job host). The client would request that the file cleanup process be held until released, giving the client an opportunity to fetch all remaining/buffered data after the job completes but *before* the output files are deleted.

This is used by `globusrun-ws` in order to ensure client-side streaming of remote files in batch mode.

2.3. MultiJobs

The new job description XML schema allows for specification of a *multijob* i.e. a job that is itself composed of several executable jobs. This is useful in order to bundle a group of jobs together and submit them as a whole to a remote GRAM installation.

2.4. Job and process rendezvous

WS GRAM services implement a rendezvous[3] mechanism to perform synchronization between job processes in a multiprocess job and between subjobs in a multijob. The job application can in fact register binary information, for

[1] ../WS_GRAM_Approach.html
[2] schemas/mj_types.html#element_holdState
[3] ../wsrendezvous/index.html

instance process information or subjob information, and get notified when all the other processes or subjobs have registered their own information. This is for instance useful for parallel jobs which need to rendezvous at a "barrier" before proceeding with computations, in the case when no native application API is available to help do the rendezvous.

3. Changed Functionality

3.1. Independent resource keys

Note: This change is done in GT 4.0.5

WS GRAM enables the client to add a self-generated resource key to the input type when submitting a new job request to the ManagedJobFactoryService (MJFS). This enables the client to keep in contact to the job in case the server fails after the job was created but before the EndpointReference (EPR) of the newly created job was sent to the client.The client is then able to create an EPR itself with the self-generated job UUID and the address of the ManagedExecutableJobService (MEJS) and query for the state of the job.

In former versions of WS GRAM the job UUID that was generated on the client-side was used in WS GRAM as the resource key of the created job resource. This has changed: WS GRAM now creates its own job UUID even if the client provides one in the input of its call to the MJFS and returns this job key inside the EPR which is returned to the client. With the self-generated job key the client can still contact the MJFS. The MJFS will simply use that mapping then. But the client can't contact the MEJS with that self-generated job key as part of an EPR in order to query for job state.

3.1.1. Open Questions

- I can't see that the mappings added to the idempotenceIdMap are removed at any time. Should we add this in the remove()-method of the MEJR?

4. Usage scenarios

4.1. Generating a valid proxy

In order to generate a valid proxy file, use the grid-proxy-init[4] tool available under `$GLOBUS_LOCATION/bin`:

```
% bin/grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA.mymachine/OU=mymachine/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ............................... Done
Your proxy is valid until: Tue Oct 26 01:33:42 2004
```

4.2. Submitting a simple job

Use the `globusrun-ws` program to submit a simple job without writing a job description document. Use the -c argument, a job description will be generated assuming the first arg is the executable and the remaining are arguments. For example:

```
% globusrun-ws -submit -c /bin/touch touched_it
```

[4] ../../security/prewsaa/user-index.html#grid-proxy-init

```
Submitting job...Done.
Job ID: uuid:4a92c06c-b371-11d9-9601-0002a5ad41e5
Termination time: 04/23/2005 20:58 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Confirm that the job worked by verifying the file was touched:

```
% ls -l ~/touched_it
-rw-r--r--  1 smartin globdev 0 Apr 22 15:59 /home/smartin/touched_it

% date
Fri Apr 22 15:59:20 CDT 2005
```

Note: you did not tell globusrun-ws where to run your job, so the default of localhost was used.

4.3. Submitting a job with the contact string

Use globusrun-ws to submit the same touch job, but this time specify the contact string.

```
% globusrun-ws -submit -F https://lucky0.mcs.anl.gov:8443/wsrf/services/ManagedJobFacto
Submitting job...Done.
Job ID: uuid:3050ad64-b375-11d9-be11-0002a5ad41e5
Termination time: 04/23/2005 21:26 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Try the same job to a remote host. Type globusrun-ws -help to learn the details about the contact string.

4.4. Submitting a job with the job description

The specification of a job to submit is to be written by the user in a job description XML file.

Here is an example of a simple job description:

```
<job>
    <executable>/bin/echo</executable>
    <argument>this is an example_string </argument>
    <argument>Globus was here</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

Tell globusrun-ws to read the job description from a file, using the -f argument:

```
% bin/globusrun-ws -submit -f test_super_simple.xml
```

```
Submitting job...Done.
Job ID: uuid:c51fe35a-4fa3-11d9-9cfc-000874404099
Termination time: 12/17/2004 20:47 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Note the usage of the substitution variable `${GLOBUS_USER_HOME}` which resolves to the user home directory.

Here is an example with more job description parameters:

```
<?xml version="1.0" encoding="UTF-8"?>
<job>
    <executable>/bin/echo</executable>
    <directory>/tmp</directory>
    <argument>12</argument>
    <argument>abc</argument>
    <argument>34</argument>
    <argument>this is an example_string </argument>
    <argument>Globus was here</argument>
    <environment>
        <name>PI</name>
        <value>3.141</value>
    </environment>
    <stdin>/dev/null</stdin>
    <stdout>stdout</stdout>
    <stderr>stderr</stderr>
    <count>2</count>
</job>
```

Note that in this example, a `<directory>` element specifies the current directory for the execution of the command on the execution machine to be `/tmp`, and the standard output is specified as the relative path `stdout`. The output is therefore written to `/tmp/stdout`:

```
% cat /tmp/stdout
12 abc 34 this is an example_string  Globus was here
```

4.5. Delegating credentials

There are three different uses of delegated credentials: 1) for use by the *MEJS* to create a remote user proxy, 2) for use by the MEJS to contact RFT, and 3) for use by RFT to contact the GridFTP servers. The EPRs to each of these are specified in three job description elements--they are jobCredentialEndpoint, stagingCredentialEndpoint, and transferCredentialEndpoint respectively. Please see the job description schema[5] and RFT transfer request schema[6] documentation for more details about these elements.

The `globusrun-ws` client can either delegate these credentials automatically for a particular job, or it can reuse pre-delegated credentials (see next paragraph) through the use of command-line arguments for specifying the credentials' EPR files. Please see the WS GRAM command-line tools documentation[7] for details on these command-line arguments.

[5] #s-wsgram-Public_Interfaces-domain-schema
[6] schemas/rft_types.html
[7] #s-wsgram-user-commandline

It is possible to use delegation command-line clients[8] to obtain and refresh delegated credentials in order to use them when submitting jobs to WS GRAM. This, for instance, enables the submission of many jobs using a shared set of delegated credentials. This can significantly decrease the number of remote calls for a set of jobs, thus improving performance.

4.6. Finding which schedulers are interfaced by the WS GRAM installation

Unfortunately there is no option yet to print the list of local resource managers supported by a given WS-GRAM service installation. But there is a way to check, whether WS-GRAM supports a certain local resource manager or not. The following command gives an example of how a client could check if Condor is available at the remote site:

```
wsrf-query \
    -s https://<hostname>:<port>/wsrf/services/ManagedJobFactoryService \
    -key {http://www.globus.org/namespaces/2004/10/gram/job}ResourceID Condor \
    "//*[local-name()='version']"
```

Replace host and port settings with the values you need. If Condor is available on the server-side, the output should look something like the following:

```
 <ns1:version xmlns:ns1="http://mds.globus.org/metadata/2005/02">4.0.3</ns1:version>
```

In this example the output indicates, that a GT is listening on the server-side, that Condor is available and that the GT version is 4.0.3. If no GT at all is running at the specified host and/or port or if the specified local resource manager is not available on the server-side, the output will be an error message.

On the server-side the *GRAM name* of local resource managers for which GRAM support has been installed can be obtained by looking at the GRAM configuration on the GRAM server-side machine, as explained here.

The GRAM name of the local resource manager can be used with the *factory type* option of the job submission command-line tool to specify which factory resource to use when submitting a job.

4.7. Specifying file staging in the job description

In order to do file staging one must add specific elements to the job description and delegate credentials appropriately (see Delegating credentials[9]). The file transfer directives follow the RFT syntax[10], which allows only for third-party transfers. Each file transfer must therefore specify a source URL and a destination URL. URLs are specified as GridFTP URLs (for remote files) or as file URLs (for files local to the service--these are converted internally to full GridFTP URLs by the service).

For instance, in the case of staging a file *in*, the source URL would be a GridFTP URL (for instance `gsiftp://job.submitting.host:2811/tmp/mySourceFile`) resolving to a source document accessible on the file system of the job submission machine (for instance `/tmp/mySourceFile`). At run-time the Reliable File Transfer service used by the MEJS on the remote machine would reliably fetch the remote file using the GridFTP protocol and write it to the specified local file (for instance `file:///${GLO-`

[8] ../../security/delegation/user-index.html#commandline
[9] #s-wsgram-user-delegating
[10] schemas/rft_types.html

BUS_USER_HOME}/my_transfered_file, which resolves to ~/my_transfered_file). Here is how the stage-in directive would look like:

```
    <fileStageIn>
        <transfer>
            <sourceUrl>gsiftp://job.submitting.host:2811/tmp/mySourceFile</sourceUrl>
            <destinationUrl>file:///${GLOBUS_USER_HOME}/my_transfered_file</destinationUrl
        </transfer>
    </fileStageIn>
```

Note: additional RFT-defined quality of service requirements can be specified for each transfer. See the RFT documentation for more information.

Here is an example job description with file stage-in and stage-out:

```
<job>
    <executable>my_echo</executable>
    <directory>${GLOBUS_USER_HOME}</directory>
    <argument>Hello</argument>
    <argument>World!</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
    <fileStageIn>
        <transfer>
            <sourceUrl>gsiftp://job.submitting.host:2811/bin/echo</sourceUrl>
            <destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
        </transfer>
    </fileStageIn>
    <fileStageOut>
        <transfer>
            <sourceUrl>file:///${GLOBUS_USER_HOME}/stdout</sourceUrl>
            <destinationUrl>gsiftp://job.submitting.host:2811/tmp/stdout</destinationUrl>
        </transfer>
    </fileStageOut>
    <fileCleanUp>
        <deletion>
            <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
        </deletion>
    </fileCleanUp>
</job>
```

Note that the job description XML does not need to include a reference to the schema that describes its syntax. As a matter of fact it is possible to omit the namespace in the GRAM job description XML elements as well. The submission of this job to the GRAM services causes the following sequence of actions:

1. The /bin/echo executable is transfered from the submission machine to the GRAM host file system. The destination location is the HOME directory of the user on behalf of whom the job is executed by the GRAM services (see <fileStageIn>).

2. The transfered executable is used to print a test string (see <executable>, <directory> and the <argument> elements) on the standard output, which is redirected to a local file (see <stdout>).

3. The standard output file is transfered to the submission machine (see <fileStageOut>).

4. The file that was initially transfered during the stage-in phase is removed from the file system of the GRAM installation (see `<fileCleanup>`).

4.8. Specifying and handling custom job description extensions.

Note

This feature has been added in GT 4.0.5. For versions older than 4.0.5 an update package is available to upgrade one's installation. See the downloads[11] page for the latest links.

Basic support is provided for specifying custom extensions to the job description. There are plans to improve the usability of this feature, but at this time it involves a bit of work.

Specifying the actual custom elements in the job description is trivial. Simply add any elements that you need between the beginning and ending `extensions` tags at the bottom of the job description as in the following basic example:

```
<job>
    <executable>/home/user1/myapp</executable>
    <extensions>
        <myData>
            <var1>hello</var1>
            <var2>world</var2>
        </myData>
    </extensions>
</job>
```

To handle this data, you will have to alter the appropriate perl scheduler script (i.e. fork.pm for the Fork scheduler, etc...) to parse the data returned from the `$description->extensions()` sub.

More information about job description extension support can be found in the Admin guide[12]

4.9. Specifying and submitting a multijob

The job description XML schema allows for specification of a *multijob* i.e. a job that is itself composed of several executable jobs, which we will refer to as *subjobs* (*note:* subjobs cannot be multijobs, so the structure is not recursive). This is useful for instance in order to bundle a group of jobs together and submit them as a whole to a remote GRAM installation.

Note that no relationship can be specified between the subjobs of a multijob. The subjobs are submitted to job factory services in their order of appearance in the multijob description.

Within a multijob description[13], each subjob description must come along with an endpoint for the factory to submit the subjob to. This enables the at-once submission of several jobs to different hosts. The factory to which the multijob is submitted acts as an intermediary tier between the client and the eventual executable job factories.

Here is an example of a multijob description:

[11] http://www.globus.org/toolkit/downloads/development/
[12] admin-index.html#s-wsgram-admin-extensions
[13] schemas/gram_job_description.html#element_multiJob

```
<?xml version="1.0" encoding="UTF-8"?>
<multiJob xmlns:gram="http://www.globus.org/namespaces/2004/10/gram/job"
     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing">
    <factoryEndpoint>
        <wsa:Address>
            https://localhost:8443/wsrf/services/ManagedJobFactoryService
        </wsa:Address>
        <wsa:ReferenceProperties>
            <gram:ResourceID>Multi</gram:ResourceID>
        </wsa:ReferenceProperties>
    </factoryEndpoint>
    <directory>${GLOBUS_LOCATION}</directory>
    <count>1</count>

    <job>
        <factoryEndpoint>
            <wsa:Address>https://localhost:8443/wsrf/services/ManagedJobFactoryService</ws
            <wsa:ReferenceProperties>
                <gram:ResourceID>Fork</gram:ResourceID>
            </wsa:ReferenceProperties>
        </factoryEndpoint>
        <executable>/bin/date</executable>
        <stdout>${GLOBUS_USER_HOME}/stdout.p1</stdout>
        <stderr>${GLOBUS_USER_HOME}/stderr.p1</stderr>
        <count>2</count>
    </job>

    <job>
        <factoryEndpoint>
            <wsa:Address>https://localhost:8443/wsrf/services/ManagedJobFactoryService</ws
            <wsa:ReferenceProperties>
                <gram:ResourceID>Fork</gram:ResourceID>
            </wsa:ReferenceProperties>
        </factoryEndpoint>
        <executable>/bin/echo</executable>
        <argument>Hello World!</argument>
        <stdout>${GLOBUS_USER_HOME}/stdout.p2</stdout>
        <stderr>${GLOBUS_USER_HOME}/stderr.p2</stderr>
        <count>1</count>
    </job>

</multiJob>
```

Notes:

- The `<ResourceID>` element within the `<factoryEndpoint>` WS-Addressing endpoint structures must be qualified with the appropriate GRAM namespace.
- Apart from the `factoryEndpoint` element, all elements at the enclosing multijob level act as defaults for the subjob parameters, in this example `<directory>` and `<count>`.
- The default `<count>` value is overridden in the subjob descriptions.

In order to submit a multijob description, use a job submission command-line tool[14] and specify the Managed Job Factory resource to be `Multi`. For instance, submitting the multijob description above using `globusrun-ws`, we obtain:

```
% bin/globusrun-ws -submit -f test_multi.xml
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:bd9cd634-4fc0-11d9-9ee1-000874404099
Termination time: 12/18/2004 00:15 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
Cleaning up any delegated credentials...Done.
```

A multijob resource is created by the factory and exposes a set of WSRF resource properties different than the resource properties of an executable job. The state machine of a multijob is also different since the multijob represents the *overall* execution of all the executable jobs it is composed of.

4.10. Specifying SoftEnv keys in the job description

Note: This feature is only available beginning from version 4.0.5 of the toolkit.

For a short introduction to SoftEnv please have a look at the SoftEnv section in the admin guide[15].

If SoftEnv is enabled on the server-side, nothing must be added to a job description to set up the environment which is specified in the `.soft` file in the remote home directory of the user before the job is submitted to the scheduler. If a different software environment should be used than the one specified in the remote `.soft` file, the user must provide SoftEnv parameters in the extensions element of the job description. The schema of the extension element for software selection in the job description is as follows:

```
<element name="softenv" type="xsd:string">
```

For example, to add the SoftEnv commands "@teragrid-basic", "+intel-compilers", "+atlas", and "+tgcp" to the job process's environment, the user would specify the following "extensions" element in the job description:

```
<extensions>
  <softenv>@teragrid-basic</softenv>
  <softenv>+intel-compilers</softenv>
  <softenv>+atlas</softenv>
  <softenv>+tgcp</softenv>
</extensions>
```

So far there is no way for a user to get information from the remote service itself whether SoftEnv support is enabled or not. The only way to check this so far is to submit a job with `/bin/env` as executable and watch the results.

Scenarios:

1. SoftEnv is **disabled** on the server-side
    a. **The user provides no SoftEnv extensions**

[14] #commandline
[15] admin-index.html#s-wsgram-admin-softenv

No SoftEnv environment is configured then before job submission, even if the user has a `.soft` file in his remote home directory.

b. **The user provides valid SoftEnv extensions**

If SoftEnv is not installed on the server then no environment will be configured

If SoftEnv is installed, the environment the user specifies in the extensions elements overwrites any softenv configuration the user specifies in a `.soft` or a `.nosoft` file in his remote home directory. The environment will be configured as specified by the user in the extension elements before job submission.

c. **The user provides invalid SoftEnv extensions**

If SoftEnv is not installed on the server then no environment will be configured

If SoftEnv is installed, the environment the user specifies in the extensions elements overwrites any softenv configuration the user specifies in a `.soft` or a `.nosoft` file in his remote home directory. Only the valid keys in the SoftEnv extensions elements will be configured. If no valid key is in there then no environment will be configured. SoftEnv warnings are logged to stdout of the job.

In general jobs don't fail if they have SoftEnv extensions in their description and SoftEnv is disabled or even not installed on the service-side. But they will fail if they rely on environments being set up before job submission.

2. SoftEnv is **enabled** on the server-side

a. **The user provides no SoftEnv extensions**

If the user has a `.soft` file (and no `.nosoft` file) in his remote home directory then the environment defined in his `.soft` file will be configured before job submission. If there is a `.soft` file) in his remote home directory no environment will be prepared.

b. **The user provides valid SoftEnv extensions**

The specified environment overwrites any softenv configuration the user specifies in a `.soft` or a `.nosoft` file in his remote home directory. The environment will be configured as specified by the user in the extension elements before job submission.

c. **The user provides invalid SoftEnv extensions**

The specified environment overwrites any softenv configuration the user specifies in a `.soft` or a `.nosoft` file in his remote home directory. Only the valid keys in the SoftEnv extension elements will be configured. If no valid key is in there then no environment will be configured. SoftEnv warnings are logged to stdout of the job.

Note: In the current implementation it is not possible to call executables directly whose paths are defined in SoftEnv without specifiying the complete path to the executable.

Example: Let's say a database query must be executed using the mysql-command. If mysql is not in the default path then the direct use of mysql as executable in the jobs description document will fail, even is the use of SoftEnv is configured. The mysql-command must be written to a script which is in the default path. Thus a job submission with the following job description document will fail:

```
<job>
  ...
  <executable>mysql</executable>
  ...
</job>
```

But when the command is embedded inside a shell script which is specified as the executable in the job description document, it will work.

```
#!/bin/sh
  ...
  mysql ...
  ...
```

Note: The use of invalid SoftEnv keys in the extension part of the job description document does not generate errors.

4.11. Specifying substitution variables in a job description

Job description variables are special strings in a job description that are replaced by the GRAM service with values that the client-side does not *a priori* know. Job description variables can be used in any path-like string or URL specified in the job description. An example of a variable is `${GLOBUS_USER_HOME}`, which represents the path to the HOME directory on the file system where the job is executed. The set of variables is fixed in the gram service implementation. This is different from previous implementations of *RSL* substitutions in GT2 and GT3, where a user could define a new variable for use inside a job description document. This was done to preserve the simplicity of the job description XML schema (relatively to the GT3.2 RSL schema), which does not require a specialized XML parser to serialize a job description document.

Details of the RSL variables are in job description doc[16] and in the substitution variable section[17] of the admin guide.

4.11.1. Changes in WS GRAM beginning from GT version 4.0.5

Beginning from version 4.0.5 additional variables can be defined on the server side for use in the job description. So far the user can't get information from WS GRAM if additional variables are defined on the server-side and if, what their names and values are. This information must be published by the provider so far.

4.12. Specifying a selfgenerated resource key during job submission

WS GRAM enables a client to add a self-generated resource key to the input type when submitting a new job request to the ManagedJobFactoryService (MJFS). The client should make sure to provide a universal unique identifier (UUID) as job resource key. For information about UUID's please read here[18].

Providing its own UUID enables a client to resubmit a job in case the server did not respond to a prior job submission request due to e.g. network failures. If the client submits a job with an already existing resource key a second time, the job will not be started again because it's already running. This avoids unnecessary and undesired resource usage and enables a reliable job submission.

Beginning from version 4.0.5 of the toolkit WS GRAM now creates its own job UUID even if the client provides one in the input of its call to the MJFS and returns this job UUID inside the endpoint reference (EPR) to the client. The client can still contact the ManagedJobFactoryService (MJFS) with the self-generated job resource key in order to re-submit a potentially not 'lost' and submitted job. But the client can't contact the ManagedExecutableJobService (MEJS) with that self-generated job key as part of an EPR in order to query for job state any more. If it's unclear whether a job request has been started by the server, the client has to submit the job with the same job UUID again in order to get an EPR from the MJFS. With this the client then can query for job state or destroy the job.

[16] schemas/gram_job_description.html#SchemaProperties
[17] admin-index.html#s-wsgram-admin-substitution-variables
[18] http://en.wikipedia.org/wiki/Universally_Unique_Identifier

5. Command-line tools

Please see the GT 4.0 WS GRAM Command-line Reference.

6. Graphical user interfaces

There is no support for this type of interface for WS GRAM.

7. Troubleshooting

When I submit a streaming or staging job, I get the following error: ERROR service.TransfreWork Terminal transfer error: [Caused by: Authentication failed[Caused by: Operation unauthorized(Mechanism le vel: Authorization failed. Expected"/CN=host/localhost.localdomain" target but r eceived "/O=Grid/OU=GlobusTest/OU=simpleCA-my.machine.com/CN=host/my.machine.com ")

- Check $GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml for the use of "localhost" or "127.0.0.1" instead of the public hostname (in the example above, "my.machine.com"). Change these uses of the loopback hostname or IP to the public hostname as neccessary.

Fork jobs work fine, but submitting PBS jobs with globusrun-ws hangs at "Current job state: Unsubmitted"

- Make sure the the log_path in $GLOBUS_LOCATION/etc/globus-pbs.conf points to locally accessible scheduler logs that are readable by the user running the container. The Scheduler Event Generator (SEG) will not work without local scheduler logs to monitor. This can also apply to other resource managers, but is most comonly seen with PBS.

- If the SEG configuration looks sane, try running the SEG tests. They are located in $GLOBUS_LOCATION/test/globus_scheduler_event_generator_*_test/. If Fork jobs work, you only need to run the PBS test. Run each test by going to the associated directory and run ./TESTS.pl. If any tests fail, report this to the gram-dev@globus.org mailing list.

- If the SEG tests succeed, the next step is to figure out the ID assigned by PBS to the queued job. Enable GRAM debug logging by uncommenting the appropriate line in the $GLOBUS_LOCATION/container-log4j.properties configuration file. Restart the container, run a PBS job, and search the container log for a line that contains "Received local job ID" to obtain the local job ID.

- Once you have the local job ID you can check the latest PBS logs pointed to by the value of "log_path" in $GLOBUS_LOCATION/etc/globus-pbs.conf to make sure the job's status is being logged. If the status is not being logged, check the documentation for your flavor of PBS to see if there's any futher configuration that needs to be done to enable job status logging. For example, PBS Pro requires a sufficient -e <bitmask> option added to the pbs_server command line to enable enough logging to satisfy the SEG.

- If the correct status is being logged, try running the SEG manually to see if it is reading the log file properly. The general form of the SEG command line is as follows: `$GLOBUS_LOCATION/libexec/globus-scheduler-event-generator -s pbs -t <timestamp>` The timestamp is in seconds since the epoch and dictates how far back in the log history the SEG should scan for job status events. The command should hang after dumping some status data to stdout. If no data appears, change the timestamp to an earlier time. If nothing ever appears, report this to the gram-user@globus.org mailing list.

- If running the SEG manually succeeds, try running another job and make sure the job process actually finishes and PBS has logged the correct status before giving up and cancelling globusrun-ws. If things are still not working,

report your problem and exactly what you have tried to remedy the situtation to the gram-user@globus.org mailing list.

The job manager detected an invalid script response

- Check for a restrictive umask. When the service writes the native scheduler *job description* to a file, an overly restrictive umask will cause the permissions on the file to be such that the submission script run through *sudo* as the user cannot read the file (bug #2655).

When restarting the container, I get the following error: Error getting delegation resource

- Most likely this is simply a case of the delegated credential expiring. Either refresh it for the affected job or destroy the job resource.

The user's home directory has not been determined correctly

- This occurs when the administrator changed the location of the users's home directory and did not restart the GT4 container afterwards. Beginning from version 4.0.3 of the GT, WS-GRAM determines a user's home directory only once in the lifetime of a container (when the user submits the first job). Subsequently submitted jobs will use the cached home directory during job execution.

8. Usage statistics collection by the Globus Alliance

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 (i.e. when Done or Failed state is entered).

- job creation timestamp (helps determine the rate at which jobs are submitted)
- *scheduler* type (Fork, *PBS*, *LSF*, *Condor*, etc...)
- jobCredentialEndpoint present in *RSL* flag (to determine if server-side user proxies are being used)
- fileStageIn present in RSL flag (to determine if the staging in of files is used)
- fileStageOut present in RSL flag (to determine if the staging out of files is used)
- fileCleanUp present in RSL flag (to determine if the cleaning up of files is used)
- CleanUp-Hold requested flag (to determine if streaming is being used)
- job type (Single, Multiple, MPI, or Condor)
- gt2 error code if job failed (to determine common scheduler script errors users experience)
- fault class name if job failed (to determine general classes of common faults users experience)

If you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration[19] for instructions.

Also, please see our policy statement[20] on the collection of usage statistics.

[19] http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets
[20] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 11. GT 4.0 WS GRAM: Developer's Guide

1. Introduction

This guide is intended to help a developer create compatible WS GRAM clients and alternate service implementations.

The key concepts for the GRAM component have not changed. Its purpose is still to provide the mechanisms to execute remote applications for a user. Given an *RSL (Resource Specification Language)* job description, GRAM submits the job to a scheduling system such as *PBS* or *Condor*, or to a simple fork-based way of spawning processes, and monitors it until completion. More details can be found here:

http://www.globus.org/toolkit/docs/3.2/gram/key

2. Before you begin

2.1. Feature summary

New Features new since 3.2

- Support for mpich-g2 jobs:
  - multi-job submission capabilities
  - ability to coordinate processes in a job
  - ability to coordinate subjobs in a multi-job
- Publishing of the job's exit code
- The ability to select the account under which the remote job will be run. If a user's grid credential is mapped to multiple accounts, then the user can specify, in the *RSL*, under which account the job should be run.
- Optional client-specified hold on a state. Released with the new "release" operation.

Other Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- File / directory clean up after job execution (after file stage out)

Deprecated Features

- managed-job-globusrun has been replaced by *globusrun-ws*.
- Service managed data streaming of job's `stdout/err` during execution.

- File staging using the GASS protocol
- File caching of stages files, e.g. GASS Cache

2.2. Tested platforms

Tested platforms for WS GRAM:

- Linux
  - Fedora Core 1 i686
  - Fedora Core 3 i686
  - Fedora Core 3 yup xeon
  - RedHat 7.3 i686
  - RedHat 9 x86
  - Debian Sarge x86
  - Debian 3.1 i686

Tested containers for WS GRAM:

- Java WS Core container
- Tomcat 4.1.31

2.3. Backward compatibility summary

Protocol changes since GT version 3.2:

- The protocol has been changed to be WSRF compliant. There is no backward compatibility between this version and any previous versions.

2.4. Technology dependencies

GRAM depends on the following GT components:

- Java WS Core
- Transport-Level Security
- Delegation Service
- RFT
- GridFTP
- MDS - internal libraries

GRAM depends on the following 3rd party software. The dependency exists only for the *batch scheduler*s configured, thus making job submissions possible to the batch scheduling service:

Scheduler adapters are included in the GT 4.0.x releases for these schedulers:

- *PBS*
- *Condor*
- *LSF*

Other scheduler adapters available for GT 4.0.x releases:

- SGE scheduler adapter interface[1]
- Loadleveler - as of release 3.3.1 IBM LoadLeveler includes a GRAM Scheduler Adapter. For more information see "What's new" in the LoadLeveler product documentation[2]
- other batch schedulers... (where the GRAM scheduler interface has been implemented)

2.5. Security considerations

No special security considerations exist at this time.

3. Architecture and design overview

The GRAM services in GT 4.0 are WSRF compliant. One of the key concepts in the WSRF[3] specification is the decoupling of a service with the public "state" of the service in the interface via the implied resource pattern[4]. Following this concept, the data of GT 4.0 GRAM jobs is published as part of WSRF resources, while there is only one service to start jobs or query and monitor their state. This is different from the OGSI[5] model of GT3 where each job was represented as a separate service. There is still a job factory service that can be called in order to create job instances (represented as WSRF resources). Each scheduling system that GRAM is interfaced with is represented as a separate factory resource. By making a call to the factory service while associating the call to the appropriate factory resource, the job submitting actor can create a job resource mapping to a job in the chosen scheduling system.

3.1. Job States

3.1.1. Overview

The *Managed Executable Job Service (MEJS)* relies on a state machine to handle state transitions. There are two sets of states: external and internal. The external states are those that the user gets in notifications and can be queried as a resource property. The internal states are those that are strictly used by the state machine to step through all the necessary internal tasks that need to be performed for a particular job.

The Managed Multi-Job Service does not rely on a state machine, but instead makes judgements after receiving notifications from the sub-jobs about which external state it should be in. The external states for the *MMJS* are identical to the ones used by the MEJS.

[1] http://www.lesc.ic.ac.uk/projects/SGE-GT4.html
[2] http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html
[3] http://www.globus.org/wsrf/
[4] http://www.globus.org/wsrf/faq.php#wsrf12
[5] http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-04_2002-10-04.pdf

3.1.2. External and Internal States of the Managed Job Services

3.1.2.1. External States of the Managed Job Services

- Unsubmitted
- StageIn
- Pending
- Active
- Suspended
- StageOut
- CleanUp
- Done
- Failed

3.1.2.2. Internal States of the Managed Executable Job Service

- None
- Restart
- Start
- StageIn
- StageInResponse
- Submit
- OpenStdout
- OpenStderr
- WaitingForStateChanges
- MergeStdout
- StageOut
- StageOutResponse
- UserCancel
- UserCancelResponse
- SystemCancel
- CleanUp
- FileCleanUp

- FileCleanUpResponse
- CacheCleanUp
- CacheCleanUpResponse
- ScratchCleanUp
- ScratchCleanUpResponse
- Suspend
- Resume
- Done
- FailureFileCleanUp
- FailureFileCleanUpResponse
- FailureCacheCleanUp
- FailureCacheCleanUpResponse
- FailureScratchCleanUp
- FailureScratchCleanUpResponse
- Failed
- UnsubmittedHold
- StageInHold
- PendingHold
- ActiveHold
- SuspendedHold
- StageOutHold
- CleanUpHold
- DoneHold
- FailedHold

3.1.3. Managed Executable Job Service Internal State Diagram

Here is a diagram illustrating the internal state transitions of the Managed Executable Job Service and how the external states are triggered within this progression: Managed Executable Job Service Internal State Transition Diagram[6].

[6] developer/MEJS_internal_state_transitions.jpg

4. Public interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in Chapter 13, *GT 4.0 Component Guide to Public Interfaces: WS GRAM*.

5. Usage scenarios

5.1. Submitting a job in Java

The following is a general scenario for submitting a job using the Java stubs and APIs. Please consult the Java WS Core API[7], Delegation API[8], Reliable File Transfer API[9], and WS-GRAM API[10] documentation for details on package names for classes referenced in the code excerpts.

1. Class imports[11]
2. Loading the job description[12]
3. Creating the factory service stub[13]
4. Setting stub security parameters[14]
5. Querying factory resource properties[15]
6. Delegating credentials (if needed)[16]
7. Creating the job resource[17]
8. Creating the job service stub[18]
9. Subscribing for job state notifications[19]
10. Releasing any state holds (if necessary)[20]
11. Destroying resources[21]

[7] http://www.globus.org/toolkit/docs/4.0/common/javawscore/Java_WS_Core_Public_Interfaces.html#apis
[8] http://www.globus.org/toolkit/docs/4.0/security/delegation/WS_AA_Delegation_Service_Public_Interfaces.html#apis
[9] http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Public_Interfaces.html#apis
[10] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html#apis
[11] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-imports
[12] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-loadingjd
[13] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-factorystub
[14] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-stubsec
[15] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-queryingrps
[16] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-delegating
[17] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-creatingjob
[18] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-jobstub
[19] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-subscribing
[20] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-releasing
[21] WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-destroying

5.2. Submitting a job in C

The following is a general scenario for submitting a job using the C stubs and APIs. Please consult the C WS Core API[22], WS-GRAM API[23] documentation for details on package names for classes referenced in the code excerpts.

1. Loading the job description[24]
2. Setting the security attributes[25]
3. Creating the factory client handle[26]
4. Querying factory resource properties[27]
5. Creating Notification Consumer[28]
6. Creating the job resource[29]
7. Subscribing for job state notifications[30]
8. Releasing any state holds (if necessary)[31]
9. Destroying resources[32]

6. Tutorials

The following tutorials are available for WS GRAM developers:

- WS-GRAM Scheduler Interface Tutorial[33]

7. Debugging

7.1. Enabling debug logging for GRAM classes

For starters, consult the Debugging section of the Java WS Core Developer's Guide[34] for details about what files to edit and other general log4j configuration information.

To turn on debug logging for the *Managed Executable Job Service (MEJS)*, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.exec=DEBUG
```

[22] http://www.globus.org/toolkit/docs/4.0/common/cwscore/C_WS_Core_Public_Interfaces.html#apis
[23] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html#apis
[24] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-loadingjd
[25] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-security
[26] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-factoryhandle
[27] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-queryingrps
[28] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-consumer
[29] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-creatingjob
[30] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-subscribing
[31] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-releasing
[32] WS_GRAM_C_Scenarios.html#s-wsgram-developer-scenarios-c-destroying
[33] developer/scheduler-tutorial.html
[34] http://www.globus.org/toolkit/docs/4.0/common/javawscore/developer-index.html#s-javawscore-developer-debugging

To turn on debug logging for the delegated proxy management code, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.utils=DEBUG
```

To turn on debug logging for the *Managed Multi Job Service (MMJS)*, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.multi=DEBUG
```

To turn on debug logging for the *Managed Job Factory Service (MJFS)*, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec.service.factory=DEBUG
```

To turn on debug logging for all GRAM code, add the following entry to the `container-log4j.properties` file:

```
log4j.category.org.globus.exec=DEBUG
```

Follow the pattern to turn on logging for other specific packages or classes.

7.2. Instrumented timings logging

Both the service and Java client API code contain special debugging statements which output certain timing data to help in determining performance bottlenecks.

The service code uses the PerformanceLog class to output the timings information. To turn on service timings logging without triggering full debug logging for the service code, add the following lines to the container-log4j.properties file:

```
log4j.category.org.globus.exec.service.factory.ManagedJobFactoryService.performance=DEBUG
log4j.category.org.globus.exec.service.exec.ManagedExecutableJobResource.performance=DEBUG
log4j.category.org.globus.exec.service.exec.StateMachine.performance=DEBUG
```

The Java client API has not been converted over to using the PerformanceLog class, so the debug statements are sent at the INFO level to avoid having to turn on full debug logging. To turn on client timings logging without triggering full debug logging for the client code, add the following line to the container-log4j.properties file:

```
log4j.category.org.globus.exec.client.e=INFO
```

There are two parsing scripts available in the source distribution that aren't distributed in any GPT package for summarizing the service and client timings data. The are located in `ws-gram/service/java/test/throughput/`, and are named `parse-service-timings.pl` and `parse-client-timings.pl`. They both simply take the path of the appropriate log file that contains the timing data. These scripts work fine with log files that have other logging statements mixed with the timing data.

7.3. Debugging script execution

It may be necessary to debug the *scheduler* scripts if jobs aren't being submitted correctly, and either no fault or a less-than-helpful fault is generated. Ideally we would like that this not be necessary; so if you find that you must resort to this, please file a bug report or let us know on the discuss e-mail list.

By turning on debug logging for the MEJS (see above), you should be able to search for *"Perl Job Description"* in the logging output to find the perl form of the *job description* that is sent to the scheduler scripts.

Also by turning on debug logging for the MEJS, you should be able to search for *"Executing command"* in the logging output to find the specific commands that are executed when the scheduler scripts are invoked from the service code. If you saved the perl job description from the previous paragraph, then you can use this to manually run these commands.

There is a perl job description attribute named `logfile` that isn't currently supported in the XML job description that can be used to print debugging info about the execution of the perl scripts. The value for this attribute is a path to a file that will be created. You can add this to the perl job description file that you created from the service debug logging before manually running the script commands.

Beyond the above advice, you may want to edit the perl scripts themselves to print more detailed information. For more information on the location and composition of the scheduler scripts, please consult the WS-GRAM Scheduler Interface Tutorial[35].

8. Troubleshooting

When I submit a streaming or staging job, I get the following error: ERROR service.TransfreWork Terminal transfer error: [Caused by: Authentication failed[Caused by: Operation unauthorized(Mechanism le vel: Authorization failed. Expected"/CN=host/localhost.localdomain" target but r eceived "/O=Grid/OU=GlobusTest/OU=simpleCA-my.machine.com/CN=host/my.machine.com ")

- Check $GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml for the use of "localhost" or "127.0.0.1" instead of the public hostname (in the example above, "my.machine.com"). Change these uses of the loopback hostname or IP to the public hostname as neccessary.

Fork jobs work fine, but submitting PBS jobs with globusrun-ws hangs at "Current job state: Unsubmitted"

- Make sure the the log_path in $GLOBUS_LOCATION/etc/globus-pbs.conf points to locally accessible scheduler logs that are readable by the user running the container. The Scheduler Event Generator (SEG) will not work without local scheduler logs to monitor. This can also apply to other resource managers, but is most comonly seen with PBS.

- If the SEG configuration looks sane, try running the SEG tests. They are located in $GLOBUS_LOCATION/test/globus_scheduler_event_generator_*_test/. If Fork jobs work, you only need to run the PBS test. Run each test by going to the associated directory and run ./TESTS.pl. If any tests fail, report this to the gram-dev@globus.org mailing list.

- If the SEG tests succeed, the next step is to figure out the ID assigned by PBS to the queued job. Enable GRAM debug logging by uncommenting the appropriate line in the $GLOBUS_LOCATION/container-log4j.properties configuration file. Restart the container, run a PBS job, and search the container log for a line that contains "Received local job ID" to obtain the local job ID.

- Once you have the local job ID you can check the latest PBS logs pointed to by the value of "log_path" in $GLOBUS_LOCATION/etc/globus-pbs.conf to make sure the job's status is being logged. If the status is not being

[35] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/developer/scheduler-tutorial-perl.html

logged, check the documentation for your flavor of PBS to see if there's any futher configuration that needs to be done to enable job status logging. For example, PBS Pro requires a sufficient -e <bitmask> option added to the pbs_server command line to enable enough logging to satisfy the SEG.

- If the correct status is being logged, try running the SEG manually to see if it is reading the log file properly. The general form of the SEG command line is as follows: `$GLOBUS_LOCATION/libexec/globus-scheduler-event-generator -s pbs -t <timestamp>` The timestamp is in seconds since the epoch and dictates how far back in the log history the SEG should scan for job status events. The command should hang after dumping some status data to stdout. If no data appears, change the timestamp to an earlier time. If nothing ever appears, report this to the gram-user@globus.org mailing list.

- If running the SEG manually succeeds, try running another job and make sure the job process actually finishes and PBS has logged the correct status before giving up and cancelling globusrun-ws. If things are still not working, report your problem and exactly what you have tried to remedy the situtation to the gram-user@globus.org mailing list.

The job manager detected an invalid script response

- Check for a restrictive umask. When the service writes the native scheduler *job description* to a file, an overly restrictive umask will cause the permissions on the file to be such that the submission script run through *sudo* as the user cannot read the file (bug #2655).

When restarting the container, I get the following error: Error getting delegation resource

- Most likely this is simply a case of the delegated credential expiring. Either refresh it for the affected job or destroy the job resource.

The user's home directory has not been determined correctly

- This occurs when the administrator changed the location of the users's home directory and did not restart the GT4 container afterwards. Beginning from version 4.0.3 of the GT, WS-GRAM determines a user's home directory only once in the lifetime of a container (when the user submits the first job). Subsequently submitted jobs will use the cached home directory during job execution.

9. Related Documentation

No related documentation links have been determined at this time.

10. Internal Components

Internal Components[36]

[36] developer/internal-components.html

Chapter 12. GT 4.0 Component Fact Sheet: Web Service Grid Resource Allocation and Management (WS GRAM)

1. Brief component overview

Web Services Grid Resource Allocation and Management (WS GRAM) component comprises a set of WSRF-compliant Web services to locate, submit, monitor, and cancel jobs on Grid computing resources. WS GRAM is not a *job scheduler*, but rather a set of services and clients for communicating with a range of different batch/cluster job schedulers using a common protocol. WS GRAM is meant to address a range of jobs where reliable operation, stateful monitoring, credential management, and file staging are important.

2. Summary of features

New Features new since 3.2

- Support for mpich-g2 jobs:
  - multi-job submission capabilities
  - ability to coordinate processes in a job
  - ability to coordinate subjobs in a multi-job
- Publishing of the job's exit code
- The ability to select the account under which the remote job will be run. If a user's grid credential is mapped to multiple accounts, then the user can specify, in the *RSL*, under which account the job should be run.
- Optional client-specified hold on a state. Released with the new "release" operation.

Other Supported Features

- Remote job execution and management
- Uniform and flexible interface to batch scheduling systems
- File staging before and after job execution
- File / directory clean up after job execution (after file stage out)

Deprecated Features

- managed-job-globusrun has been replaced by *globusrun-ws*.
- Service managed data streaming of job's `stdout/err` during execution.
- File staging using the GASS protocol
- File caching of stages files, e.g. GASS Cache

3. Usability summary

WS GRAM usability has improved considerably in GT4!

- Improved service performance:
  - Job Concurrency. The maximum number of jobs a gram service can manage at one time.
  - Job Throughput. The rate at which jobs can be processed (e.g. x /bin/date jobs per minute).
  - Job Latency. The rate at which a single operation to the GRAM service can be processes.
  - Details of performance testing can be found here.[1]
- Fault Tolerance. The ability for the GRAM service to recover after a container or host crash. Recovery includes the continued processing and monitoring of all jobs managed by the GRAM service at the time of the crash. The 4.0 GRAM architecture was simplified (from GT3) which is the main reason that fault tolerance has improved.

4. Backward compatibility summary

Protocol changes since GT version 3.2:

- The protocol has been changed to be WSRF compliant. There is no backward compatibility between this version and any previous versions.

5. Technology dependencies

GRAM depends on the following GT components:

- Java WS Core
- Transport-Level Security
- Delegation Service
- RFT
- GridFTP
- MDS - internal libraries

GRAM depends on the following 3rd party software. The dependency exists only for the *batch scheduler*s configured, thus making job submissions possible to the batch scheduling service:

Scheduler adapters are included in the GT 4.0.x releases for these schedulers:

- *PBS*
- *Condor*
- *LSF*

[1] http://www.globus.org/toolkit/docs/4.0/perf_overview.html

Other scheduler adapters available for GT 4.0.x releases:

- SGE scheduler adapter interface[2]
- Loadleveler - as of release 3.3.1 IBM LoadLeveler includes a GRAM Scheduler Adapter. For more information see "What's new" in the LoadLeveler product documentation[3]
- other batch schedulers... (where the GRAM scheduler interface has been implemented)

6. Tested platforms

Tested platforms for WS GRAM:

- Linux
  - Fedora Core 1 i686
  - Fedora Core 3 i686
  - Fedora Core 3 yup xeon
  - RedHat 7.3 i686
  - RedHat 9 x86
  - Debian Sarge x86
  - Debian 3.1 i686

Tested containers for WS GRAM:

- Java WS Core container
- Tomcat 4.1.31

7. Associated standards

WS GRAM does not currently have any associated standards.

8. For More Information

Click here[4] for more information about this component.

[2] http://www.lesc.ic.ac.uk/projects/SGE-GT4.html
[3] http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.infocenter.doc/library.html
[4] index.html

Chapter 13. GT 4.0 Component Guide to Public Interfaces: WS GRAM

1. Semantics and syntax of APIs

1.1. Programming Model Overview

This component consists abstractly of two interfaces: the Managed Job Factory Port Type(MJFPT) and the Managed Job Port Type (MJPT).

In actuality there are three service/resource implementations, two of which implement the basic MJPT. The first one is the service which actually talks to a particular local resource manager to execute a process on the remote computer or cluster. This one is called a *Managed Executable Job Service (MEJS)* and its resource is called the Managed Executable Job Resource (MEJR). The second is a special implementation which accepts a multi-job description, breaks the description up into single-job descriptions, and then submits each of these so-called "sub-jobs" to an MEJS. This implementation is called the *Managed Multi-Job Service (MMJS)*. Its resource is called the Managed Multi-Job Resource (MMJR)

Because of the fact that these two job services use the same port type, the API for accessing both the MEJR and the MMJR are identical. The *MJFS* creates the appropriate job resource depending on the factory resource used to qualify the operation call. Most of the factory resources represent local resource managers used by the MEJS (*PBS*, *LSF*, *Condor*). There is a special Multi factory resource which represents an abstract multi-job resource manager. The appropriate *job description* type is required for the two different types of managed job.

1.2. Component API

Java API Documentation Links (Javadoc)

- Client API [1]
- Auto-Generated Service Stubs and Persistence Data Objects API [2]
- Service API [3]
- Utilities API [4]
- Job Monitoring API [5]

C API Documentation Links

- All C APIs[6]
- WS-GRAM Client Bindings[7] [noframes[8]]

[1] http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_client_java
[2] http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_common_java
[3] http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_service_java
[4] http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_utils_java
[5] http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_gram_job_monitoring_common_java
[6] http://www.globus.org/api/c-globus-4.0/
[7] http://www.globus.org/api/c-globus-4.0/globus_c_gram_client_bindings/html/index.html
[8] http://www.globus.org/api/c-globus-4.0/globus_c_gram_client_bindings/html/modules.html

- WS-Rendezvous Client Bindings[9] [noframes[10]]
- WS-GRAM Scheduler Event Generator[11] [noframes[12]]

2. Semantics and syntax of the WSDL

2.1. Protocol overview

WS-GRAM allows for remote execution and management of programs through the creation of a managed job. The management of the job is taken care of primarily by core toolkit functionality (WS-ResourceLifetime and WS-BaseN implementations). Please see the Java WS Core documentation[13] on notifications and resource lifetime (destruction) for more information.

2.1.1. Managed Job Factory Service (MJFS)

A single MJFS is used to create all jobs for all users. For each local resource manager, a dedicated Managed Job Factory Resource (MJFR) enables the MJFS to publish information about the characteristics of the compute resource, for example:

- host information
- GridFTP URL (for file staging and streaming)
- compute cluster size and configuration, and so on...

In addition, there is a special MJFR which is used for creating MMJRs.

2.1.2. Managed Executable Job Service (MEJS)

A single *MEJS* is used to manage all executable jobs for all users. Each Managed Executable Job Resource (MEJR) enables the MEJS to publish information about the individual job the MEJR represents. This information can be accessed by querying the MEJS for the resource properties of a given MEJR, such as the:

- current job state
- stdout location
- stderr location
- exit code, and so on.

2.1.3. Managed Multi-Job Service (MMJS)

A single MMJS is used to manage all multi-jobs for all users. Each Managed Multi-Job Resource (MMJR) enables the MMJS to publish information about the individual multi-job the MMJR represents. This information can be accessed by querying the MMJS for the resource properties of a given MMJR, such as the:

- current overall job state

[9] http://www.globus.org/api/c-globus-4.0/globus_c_rendezvous_client_bindings/html/index.html
[10] http://www.globus.org/api/c-globus-4.0/globus_c_rendezvous_client_bindings/html/modules.html
[11] http://www.globus.org/api/c-globus-4.0/globus_scheduler_event_generator/html/index.html
[12] http://www.globus.org/api/c-globus-4.0/globus_scheduler_event_generator/html/modules.html
[13] http://www.globus.org/toolkit/docs/4.0/common/javawscore/index.html

- list of sub-job EPRs

2.2. Operations

There are just two operations defined in the GRAM port types (not counting the Rendezvous port type which is used for MPI job synchronization): "createManagedJob" in the Managed Job Factory port type, and "release" in the Managed Job port type. All other operations (such as canceling/killing the job and querying for resource properties) are provided by the underlying WSRF implementation of the toolkit.

2.2.1. ManagedJobFactoryPortType

- `createManagedJob`: This operation creates either a MEJR or MMJR, subscribes the client for notifications if requested, and replies with one or two endpoint references (EPRs). The input of this operation consists of a *job description*, an optional initial termination time for the job resource, and an optional state notification subscription request.

  The first EPR:

  - is qualified with the identifier to the newly created MEJR or MMJR
  - points to either the MEJS or MMJS.

  The second EPR:

  - is only present if a notification subscription was requested
  - is qualified with the identifier to the newly created subscription resource
  - points to the subscription manager service.

  Using the optional subscription request provides an efficient means of subscribing to the newly created MEJR or MMJR without additional round-trip messages. Clients who subscribe afterwards must check the current status of the job, since the inherent race-condition means some state-changes may have occurred prior to the separate subscription request. In any event, there is a slight risk of lost notifications due to the lack of reliability guarantees in the notification delivery mechanism from WS-BaseNotification.

The ManagedJobFactoryPortType also has all the operations and publishes all the resource properties (via the MJFR) defined in the following WS-ResourceProperties[14] port types:

- `GetResourceProperty`
- `GetMultipleResourceProperties`
- `QueryResourceProperties`

2.2.2. ManagedJobPortType

- `release:` This operation takes no parameters and returns nothing. Its purpose is to release a hold placed on a state through the use of the "holdState" field in the job description. See the domain-specific WS GRAM component documentation[15] for more information on the "holdState" field.

[14] http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf
[15] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html

The ManagedJobPortType also has all the operations and publishes all the resource properties (via the MJFR) defined in the following port types:

WS-ResourceProperties[16] *port types:*

- `GetResourceProperty`
- `GetMultipleResourceProperties`
- `QueryResourceProperties`

WS-ResourceLifetime[17] *port types:*

- `ScheduledResourceTermination`
- `ImmediateResourceTermination`

WS-BaseNotification[18] *port type:*

- `NotificationProducer`

2.2.3. Managed Executable Job Port Type

This port type does not define any new operations. See Section 2.3, “Resource properties”.

2.2.4. Managed Multi-Job Port Type

This port type does not define any new operations. See Section 2.3, “Resource properties”.

2.3. Resource properties

2.3.1. Managed Job Factory Port Type

- `{http://www.globus.org/namespaces/2004/10/gram/job}condorArchitecture` Condor architecture label.
- `{http://www.globus.org/namespaces/2004/10/gram/job}condorOS` Condor OS label.
- `{http://www.globus.org/namespaces/2004/10/gram/job}delegationFactoryEndpoint` The endpoint reference to the delegation factory used to delegated credentials to the job.
- `{http://mds.globus.org/glue/ce/1.1}GLUECE` GLUE data
- `{http://mds.globus.org/glue/ce/1.1}GLUECESummary` GLUE data summary
- `{http://www.globus.org/namespaces/2004/10/gram/job}globusLocation` The location of the Globus Toolkit installation that these services are running under.
- `{http://www.globus.org/namespaces/2004/10/gram/job}hostCPUType` The job host CPU architecture (i686, x86_64, etc...)

[16] http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf
[17] http://www.ibm.com/developerworks/library/ws-resource/ws-resourcelifetime.pdf
[18] ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf

- {http://www.globus.org/namespaces/2004/10/gram/job}hostManufacturer The host manufacturer name. May be "unknown".
- {http://www.globus.org/namespaces/2004/10/gram/job}hostOSName The host OS name (Linux, Solaris, etc...)
- {http://www.globus.org/namespaces/2004/10/gram/job}hostOSVersion The host OS version.
- {http://www.globus.org/namespaces/2004/10/gram/job}localResourceManager The local resource manager type (i.e. Condor[19], Fork, LSF[20], Multi, PBS[21], etc...)
- {http://mds.globus.org/metadata/2005/02}ServiceMetaDataInfo service start time, Globus Toolkit(R) version, service type name
- {http://www.globus.org/namespaces/2004/10/gram/job}scratchBaseDirectory The directory recommended by the system administrator to be used for temporary job data.
- {http://www.globus.org/namespaces/2004/10/gram/job}stagingDelegationFactory-Endpoint The endpoint reference to the delegation factory used to delegated credentials to the staging service (RFT).

2.3.2. Managed Job Port Type

- {http://www.globus.org/namespaces/2004/09/rendezvous}Capacity Used for Rendezvous.
- {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}CurrentTime Time of creation.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://www.globus.org/namespaces/2004/10/gram/job/faults}fault The fault (if generated) indicating the reason for failure of the job to complete.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}holding Indicates whether a hold has been placed on this job.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}localUserId The job owner's local user account name.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RegistrantData Used for Rendezvous.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RendezvousCompleted Used for Rendezvous.
- {http://www.globus.org/namespaces/2005/5/gram/job/description}serviceLevelAgree-ment A wrapper around fields containing the single-job and multi-job descriptions or RSLs[22]. Only one of these sub-fields shall have a non-null value.

[19] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#condor
[20] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#lsf
[21] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#pbs
[22] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#rsl

- {http://www.globus.org/namespaces/2004/10/gram/job/types}state The current state of the job.
- {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}TerminationTime Time when the resource expires.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}Topic Used in notifiation.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}TopicExpressionDialects Used in notifiation.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}userSubject The GSI certificate DN of the job owner.

2.3.3. Managed Executable Job Port Type

- {http://www.globus.org/namespaces/2004/09/rendezvous}Capacity Used for Rendezvous.
- {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}CurrentTime Time of creation.
- {http://www.globus.org/namespaces/2005/09/gram/job/exec}credentialPath The path (relative to the job process) to the file containing the user proxy used by the job to authenticate out to other services.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}exitCode The exit code generated by the job process.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://www.globus.org/namespaces/2004/10/gram/job/faults}fault The fault (if generated) indicating the reason for failure of the job to complete.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}holding Indicates whether a hold has been placed on this job.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}localUserId The job owner's local user account name.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RegistrantData Used for Rendezvous.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RendezvousCompleted Used for Rendezvous.
- {http://www.globus.org/namespaces/2005/5/gram/job/description}serviceLevelAgreement A wrapper around fields containing the single-job and multi-job descriptions or RSLs[23]. Only one of these sub-fields shall have a non-null value.

[23] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#rsl

- {http://www.globus.org/namespaces/2004/10/gram/job/types}state The current state of the job.
- {http://www.globus.org/namespaces/2005/09/gram/job/exec}stderrURL A GridFTP URL to the file generated by the job which contains the stderr.
- {http://www.globus.org/namespaces/2005/09/gram/job/exec}stdoutURL A GridFTP URL to the file generated by the job which contains the stdout.
- {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}TerminationTime Time when the resource expires.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}Topic Used in notifiation.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}TopicExpressionDialects Used in notifiation.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}userSubject The GSI certificate DN of the job owner.

2.3.4. Managed Multi-Job Port Type

- {http://www.globus.org/namespaces/2004/09/rendezvous}Capacity Used for Rendezvous.
- {http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}CurrentTime Time of creation.
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}FixedTopicSet ???
- {http://www.globus.org/namespaces/2004/10/gram/job/faults}fault The fault (if generated) indicating the reason for failure of the job to complete.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}holding Indicates whether a hold has been placed on this job.
- {http://www.globus.org/namespaces/2004/10/gram/job/types}localUserId The job owner's local user account name.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RegistrantData Used for Rendezvous.
- {http://www.globus.org/namespaces/2004/09/rendezvous}RendezvousCompleted Used for Rendezvous.
- {http://www.globus.org/namespaces/2005/5/gram/job/description}serviceLevelAgreement A wrapper around fields containing the single-job and multi-job descriptions or RSLs[24]. Only one of these sub-fields shall have a non-null value.

[24] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#rsl

- `{http://www.globus.org/namespaces/2004/10/gram/job/types}state` The current state of the job.
- `{http://www.globus.org/namespaces/2004/10/gram/job/multi}subJobEndpoint` A set of endpoint references to the sub-jobs created by this multi-job.
- `{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}TerminationTime` Time when the resource expires.
- `{http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}Topic` Used in notifiation.
- `{http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd}TopicExpressionDialects` Used in notifiation.
- `{http://www.globus.org/namespaces/2004/10/gram/job/types}userSubject` The GSI certificate DN of the job owner.

2.4. WSDL and Schema Definition

WSDL links:

- ManagedJobFactoryPortType[25]
- ManagedJobPortType[26]
- ManagedExecutableJobPortType[27]
- ManagedMulti-JobPortType[28]

Schema links:

- CAS Types[29]
- File System Mapping Config Schema[30]
- GLUE Schema
  - Batch Providers[31]
  - Compute Element[32]
  - Metadata[33]

[25] http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/managed_job_factory_port_type.wsdl?rev=1.5&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[26] http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/managed_job_port_type.wsdl?rev=1.6&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[27] http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/managed_executable_job_port_type.wsdl?rev=1.4&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[28] http://viewcvs.globus.org/viewcvs.cgi/ws-gram/common/schema/gram/managed_multi_job_port_type.wsdl?rev=1.3&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[29] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/cas_types.html
[30] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_fs_map.html
[31] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/batchproviders.html
[32] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/glue_ce.html
[33] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/metadata.html

- Job Description Schema[34]
- Managed Job Faults Schema[35]
- Managed Job Types Schema[36]
- RFT Types Schema[37]
- WS Addressing Schema[38]
- WS Base Faults Schema[39]

3. Command-line tools

Please see the GT 4.0 WS GRAM Command-line Reference.

4. Graphical User Interface

There is no support for this type of interface for WS GRAM.

5. Semantics and syntax of domain-specific interface data

5.1. Single-Job Description

The general form of a *job description* used to start a single job (meant for creating a Managed Executable Job Resource instance) is as follows:

```
<job>
    <!--put additional elements here-->
    <executable><!--put executable pat here--></executable>
    <!--put additional elements here-->
</job>
```

Here is a basic example of a job description for a single-job:

```
<job>
    <executable>bin/echo</executable>
    <argument>Testing</argument>
    <argument>1...2...3</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

[34] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html
[35] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/mj_faults.html
[36] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/mj_types.html
[37] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/rft_types.html
[38] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/ws_addressing.html
[39] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/ws_base_faults.html

5.2. Multi-Job Description

The general form of a job description used to start a multi-job (meant for creating a Managed Multi Job Resource instance) is as follows:

```
<multiJob>
    <!--Put subjob default elements here.-->
    <job>
        <factoryEndpoint
                xmlns:gram="http://www.globus.org/namespaces/2004/10/gram/job"
                xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing">
            <wsa:Address>
                <!--put ManagedJobFactoryService address here-->
            </wsa:Address>
            <wsa:ReferenceProperties>
                <gram:ResourceID><!--put scheduler type here--></gram:ResourceID>
            </wsa:ReferenceProperties>
        </factoryEndpoint>
        <executable><!--put executable path here--></executable>
    </job>
    <!--put additional job elements here-->
</multiJob>
```

Here is a basic example of a job description for a multi-job:

```
<multiJob>
    <executable>/bin/echo</executable>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
    <job>
        <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://mymachine.mydomain.com:8443/wsrf/services/ManagedJobFactoryService
            </wsa:Address>
            <wsa:ReferenceProperties>
                <gram:ResourceID>Pbs</gram:ResourceID>
            </wsa:ReferenceProperties>
        </factoryEndpoint>
        <argument>Testing</argument>
        <argument>1...2...3</argument>
    <job>
    <job>
        <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://myothermachine.myotherdomain.org:8443/wsrf/services/ManagedJobFact
            </wsa:Address>
            <wsa:ReferenceProperties>
```

```
                <gram:ResourceID>Pbs</gram:ResourceID>
            </wsa:ReferenceProperties>
        </factoryEndpoint>
        <argument>Hi There!</argument>
        <argument>Dear John!</argument>
    </job>
</multiJob>
```

5.3. Staging Directives

The WS-GRAM job description schema imports types from the RFT job description schema for specifying staging directives (i.e. `fileStageIn`, `fileStageOut`, and `fileCleanUp`). See the RFT domain-specific interface documentation[40] for details on these imported types.

Since `fileStageIn` and `fileStageOut` are of type TransferRequestType[41] and `fileCleanUp` is of type DeleteRequestType[42], mentally replace "transferRequest" with "fileStageIn" or "fileStageOut", and "deleteRequest" with "fileCleanUp" in the RFT domain-specific interface documentation. The Request Options[43] section is of particular usefullness.

5.4. Job Description Schema Reference

Please see the Job Description Schema documentation[44] for details about the job description elements and substitution variables used to define GRAM jobs.

6. Configuration interface

Please see the Configuring WS GRAM.

7. Environment variable interface

There is no support for this type of interface for WS GRAM.

[40] http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Public_Interfaces.html#s-rft-Public-Interfaces-domain
[41] http://www.globus.org/toolkit/docs/4.0/data/rft/rft_job_description.html#type_TransferRequestType
[42] http://www.globus.org/toolkit/docs/4.0/data/rft/rft_job_description.html#type_DeleteRequestType
[43] http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Public_Interfaces.html#s-domain-specific-interface-options
[44] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html

Chapter 14. GT 4.0 WS GRAM: Quality Profile

1. Test coverage reports

- 4.0.0 Testing Status Report[1]
- 4.0.0 Clover Test Coverage Report[2]

2. Code analysis reports

- No code analysis reports have been generated at this time.

3. Outstanding bugs

- Current NEW, ASSIGNED, and REOPENED Bugzilla entries[3] for WS-GRAM.

4. Bug Fixes

- The following link lists all of the bugs resolved for WS GRAM since GT 3.2: Resolved Bugs[4]

5. Performance reports

- 4.0.0 Throughput Statistics[5]
- 32,000 jobs achieved in Max Concurrency test[6]

[1] reports/tests/
[2] reports/coverage/
[3] http://bugzilla.globus.org/bugzilla/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=GRAM&component=wsrf+discovery+interface&component=wsrf+gram+clients&component=wsrf+managed+execution+job+service&component=wsrf+managed+job+factory+service&component=wsrf+managed+multi+job+service&component=wsrf+rendezvous&component=wsrf+scheduler+interface&component=wsrf+tests&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=
[4] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=GRAM&component=wsrf+discovery+interface&component=wsrf+gram+clients&component=wsrf+managed+execution+job+service&component=wsrf+managed+job+factory+service&component=wsrf+managed+multi+job+service&component=wsrf+rendezvous&component=wsrf+scheduler+interface&component=wsrf+tests&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=RESOLVED&bug_status=VERIFIED&bug_status=CLOSED&resolution=FIXED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfield=resolution&chfieldfrom=2004-09-01&chfieldto=2005-04-22&chfieldvalue=&namedcmd=4.0+gram&newqueryname=&field0-0-0=noop&type0-0-0=noop&value0-0-0=
[5] reports/throughput/
[6] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3090

Chapter 15. GT 4.0 Samples for WS GRAM

1. [name of sample]

[enter info about this sample]

Chapter 16. GT 4.0 Migrating Guide for WS GRAM

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

1. Migrating from GT2

1.1. WS GRAM - GT2 Migration Guide

1.1.1. Installation / Deployment Differences

In pre-WS GRAM, 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 WS GRAM, 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.

1.1.2. Security Differences

1.1.2.1. Proxies and Delegation

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

1.1.2.2. Network Communication

In pre-WS gram, 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 WS GRAM, 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.

1.1.2.3. 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 WS GRAM, 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[1]) are started via sudo[2]. Authorization is done via the globus-gridmap-and-execute program.

1.1.3. Scheduler Interaction Differences

In pre-WS gram, 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 WS GRAM 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

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#scheduler
[2] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#sudo

- stage_out: rft
- remote_io_file_create: rft or resource property queries
- proxy_relocate: delegation service
- proxy_update: delegation service

1.1.4. Local Node Impact

In pre-WS gram, 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 WS GRAM, 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[3] program, monitoring a log file for job state changes.
Migration from pre-WS GRAM to WS-GRAM

1.2. User - Migration Guide

1.2.1. Command Line Tools

Typical interactions with the pre-WS gram 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[4] in RSL[5] format, and the globus-job-submit and globus-job-run scripts would construct that based on command line options.

In WS GRAM, the globusrun-ws[6] command implements the functionality of globusrun using the XML Job Description language in place of the RSL format job description of pre-WS GRAM. 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.

[3] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#seg
[4] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#job-description
[5] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#rsl
[6] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#globusrun-ws

Table 16.1. Command Line Option Comparison

Description	pre-WS GRAM globusrun option	WS GRAM 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 pre-WS GRAM, streaming is done using https connections from the job manager to a GASS server embedded in the globusrun program. In WS GRAM, 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.

1.3. Developer - API and RSL Migration Guide

This table describes the migration path for applications which use the C language interface to pre-WS gram. This table covers the globus_gram_client API.

Table 16.2. 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/2004/10/gram/job/types}state
globus_gram_client_register_job_status()	ManagedExecutableJobPortType_GetResourceProperty_epr_register() with the property name {http://www.globus.org/namespaces/2004/10/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_unregistration()	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 Prews GRAM to ws-GRAM

Pre-WS GRAM uses a custom language for specifying a job description. WS GRAM uses an xml based language for this same purpose. In pre-WS GRAM, relations (such as count=5) can occur in any order within the RSL; in WS GRAM, 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 16.3. 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</name> ... <name>ENV_VAR_N</name> <value>ENV_VAR_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 = JOBTYPE)	<jobType>JOBTYPE</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 WS GRAM are multiple (the default), single, mpi, and condor.

Note 4: The WS GRAM 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 WS GRAM:

- 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 WS GRAM, 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 WS GRAM service does not use GASS, so there is no equivalent to this.
- restart: There is no equivalent.
- rsl_substitution: The WS GRAM 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 WS GRAM job description document for description of RSL variable syntax, values, and attributes where they may be used.
- save_state: All WS GRAM 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 WS GRAM 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 WS GRAM service, so there is no equivalent element for restarting output streaming at a specific point.

Here are some examples of converting some pre-WS GRAM RSLs to WS GRAM.

Table 16.4. RSL Migration Examples

pre-WS GRAM RSL	WS GRAM Job Descriptio
(* Simple Job Request With Arguments *) &(executable = /bin/echo) (arguments = Hello, Grid)	<?xml v <!-- Si <job xm <ns1: <ns1: <ns1: </job>

pre-WS GRAM RSL	WS GRAM Job Descriptio
(* 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-WS GRAM RSL	WS GRAM Job Descriptio
	<?xml v <!-- Mu <multiJ <!- xml <!- xml <fa < < < < < </f <jo < < < < < < < < < </j <jo

pre-WS GRAM RSL	WS GRAM Job Descriptio
	< < < < < < < < < < </j </multi

2. Migrating from GT3

Migrating to GT 4.0 from GT version 3.2:

- The 4.0 protocol has been changed to be WSRF compliant. There is no backward compatibility between 4.0 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[7]. 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[8] has replaced managed-job-globusrun as the WS GRAM client submission program. The main reason was performance. The cost of JVM startup for each job submission through managed-job-globusrun was

[7] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#uuid
[8] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#globusrun-ws

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[9] 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 WS GRAM, this feature is no longer present. In GT 4.0 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[10]. It can then be canceled once the StageIn-Hold state notification is received.
- remoteIoUrl has been removed. This was a hack for pre-ws GRAM 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.

[9] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#rsl
[10] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Glossary.html#scheduler

- 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.

GT 4.0 WS GRAM Command-line Reference

Name

globusrun-ws -- Official job submission client for WS GRAM

`globusrun-ws -submit` [-batch] [-quiet] [-no-cleanup] [-streaming] [-streaming-out *`filename`*] [-streaming-err *`filename`*] [-host-authz] [-self-authz] [-subject-authz *`subject name`*] [-private] [-http-timeout *`milliseconds`*] [-debug] [-allow-ipv6] [-passive] [-nodcau] [[-factory-epr-file *`filename`*] [[-factory *`contact`*] | [-factory-type *`type`*]]] [[-submission-id *`uuid`*] | [-submission-id-file *`filename`*]] [-submission-id-output-file *`filename`*] [-job-epr-output-file *`filename`*] [-job-delegate] [-staging-delegate] [-job-credential-file *`filename`*] [-staging-credential-file *`filename`*] [-transfer-credential-file *`filename`*] [-termination [`+HH:MMmm/dd/yyyy HH:MM`]] [[-job-description-file *`filename`*] | [-job-command [--] *`program arg ...`*]]
`globusrun-ws -validate` -job-description-file *`filename`*
`globusrun-ws -monitor` -job-epr-file *`filename`* [-quiet] [-no-cleanup] [-streaming] [-streaming-out *`filename`*] [-streaming-err *`filename`*] [-host-authz] [-self-authz] [-subject-authz *`subject name`*] [-private] [-http-timeout *`milliseconds`*] [-debug] [-allow-ipv6] [-passive] [-nodcau]
`globusrun-ws -status` -job-epr-file *`filename`* [-host-authz] [-self-authz] [-subject-authz *`subject name`*] [-private] [-http-timeout *`milliseconds`*] [-debug]
`globusrun-ws -kill` -job-epr-file *`filename`* [-host-authz] [-self-authz] [-subject-authz *`subject name`*] [-private] [-http-timeout *`milliseconds`*] [-debug]
`globusrun-ws -help`
`globusrun-ws -usage` [-submit] [-validate] [-monitor] [-status] [-kill]
`globusrun-ws -version(s)`

Description

globusrun-ws (WS GRAMclient) is a program for submitting and managing jobs to a local or remote job host. WS GRAMprovides secure job submission to many types of *job scheduler* for users who have the right to access a job hosting resource in a Grid environment. All WS GRAMsubmission options are supported transparently through the embedded request document input. `globusrun-ws` offers additional features to fetch job output files incrementally during the run as well as to automatically delegate credentials needed for certain optional WS GRAMfeatures. Online and batch submission modes are supported with reattachment (recovery) for jobs whether they were started with this client or another WS GRAMclient application.

Command options

Quiet mode

A variety of protocol status messages, warning messages, and output data may be printed to standard output and error under multiple command modes. The *quiet mode* suppresses all but fatal standard error messages in order to have clean outputs for use in scripting or with the *streaming output mode* where application output is retrieved and output.

-q, -quiet If supplied, all non-fatal status and protocol-related messages are suppressed.

Debug mode

-dbg, -debug If supplied, all soap messages and ftp control messages will be displayed on stderr.

Protocol Options

Service authorization

Usually, secure communication includes mutual authentication. In addition to the service authorizing the client for the requested operation(s), an authorization decision is made by the client to determine whether the remote service is the one intended.

-host, -host-authz — The GSI "host authorization" rule is used to verify that the service is using a host credential appropriate for the underlying service address information. This is the default.

-self, -self-authz — The GSI "self authorization" rule is used to verify that the service is using a (proxy) credential derived from the same identity as the client's.

-subject, -subject-authz subject name — The service must be using a credential with the exact subject name provided by this option.

Security Protocol

The client uses secure transport for all https endpoints and secure message for http. Secure conversation is currently unsupported.

-p, -private — If supplied, privacy-protection is enabled between globusrun-ws and WS GRAMor GridFTP services. It is a fatal error to select privacy protection if it is not available due to build options or other security settings. **Note:** Currently only supported with https endpoints.

Timeouts

-T, -http-timeout milliseconds — Set timeout for HTTP socket, in milliseconds, for all Web services interactions. The default value is 120000 (2 minutes).

Signal handling

-n, -no-cleanup — If supplied, the default behavior of trapping interrupts (SIG_INTR) and cancelling the job is disabled. Instead, the interrupt simply causes the tool to exit without affecting the ManagedJob resource.

Submit options

-submit — The -submit command submits (or *resubmits*) a job to a job host using an XML-based job description[1] document. The -submit command can submit jobs in one of three output modes: batch, interactive, or interactive-streaming.

Output Mode

The user can select several tool behaviors following submission. In *batch mode*, the tool prints the resulting ManagedJob EPR as the sole standard output (unless in *quiet mode*) and exits. In *interactive mode*, the tool keeps running in order to monitor job status. Interactive mode is qualitatively equivalent to a batch-mode submission immediately followed a second invocation of globusrun-ws using the -monitor command. In interactive mode, an optional *streaming mode* where job output files are fetched and output from globusrun-ws.

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html#s-wsgram-Public_Interfaces-domain

-b, -batch — If supplied, the batch mode is enabled. The default is interactive mode. The tool prints the resulting ManagedJob EPR as the sole standard output (unless in quiet mode) and exits.

-s, -streaming — The standard output and standard error files of the job are monitored and data is written to the corresponding output of globusrun-ws. The standard output will contain ONLY job output data, while the standard error may be a mixture of job error output as well as globusrun-ws messages, unless the *quiet mode* is also enabled.

Streaming output depends on the ability to access job outputs via GridFTP. If -streaming mode is selected and the *job description* does not already specify output file redirection for the job host, then globusrun-ws adds unique output file name redirections and automatic cleanup directives to the job description.

If you are using -batch mode, but intend to use -streaming with -monitor, you may want to still include -streaming. -streaming always introduces a 'CleanUp Hold' state which ensures that all the data is streamed before the files are destroyed. If you do use -streaming with -batch, you **must** come back with -monitor so the hold can be released.

This option implies -staging-delegate if the stdout and stderr entries are not specified in the job description.

-so, -stdout-file filename — append stdout out stream to the specified file instead of to stdout.

-se, -stderr-file filename — append stderr out stream to the specified file instead of to stderr.

Streaming Options

Streaming makes use of GridFTP client calls to retrieve user data. The following options apply to such transfers.

-ipv6, -allow-ipv6 — Allow streaming transfers to use IPV6.

-passive — Force streaming transfers to use MODE S to allow for passive mode transfers. (Useful if you're behind a firewall, but expensive because there is no connection caching).

-nodcau — Disable data channel authentication on streaming transfers

Factory information

Addressing information for the ManagedJobFactory target of this submission must be provided. If neither option is specified, and no EPR is supplied in the job description, then "-factory localhost -factory-type fork" is assumed.

-Ff, -factory-epr-file filename — If supplied, this option causes the EPR for the ManagedJobFactory to be read from the given file. This EPR is used as the service endpoint for submission of the job.

-F, -factory contact — If supplied, this option causes an EPR to be constructed using ad-hoc methods that depend on GT implementation details. For interoperability to other implementations of WS_GRAM, the -factory-epr-file option should be used instead.

[protocol://]{hostname|hostaddr}[:port][/service]

Default values form the following contact information if not overridden:

https://localhost:8443/wsrf/services/ManagedJobFactoryService

-Ft, -factory-type type — In the absence of -factory-epr-file, this option refines the behavior of the -factory option to select a specific type of scheduler. The default is "Fork" for single jobs and "Multi" for *multijobs*.

Job description

A description of the job to be submitted must be provided with the -submit command, either using the WS GRAMXML description syntax or a simpler Unix command and argument list.

-f, -job-description-file filename — If supplied, this option causes the job description to be read from the given file. This description is modified according to the other options and passed in the WS GRAMsubmission messages. The root element of this file must be 'job' for a single job or 'multiJob' for a multijob.

-c, -job-command [--] prog [arg ...] — If supplied, this option take all remaining globusrun-ws arguments as its arguments; therefore it must appear last among globusrun-ws options. This option causes globusrun-ws to generate a simple job description with the named program and arguments.

Submission ID

A submission ID may be used in the WS GRAMprotocol for robust reliability in the face of message faults or other transient errors to ensure that at most one instance of a job is executed, i.e. to prevent accidental duplication of jobs under rare circumstances with client retry on failure. The globusrun-ws tool always uses this feature, requiring either a submission ID to be passed in as input or a new unique ID to be created by the tool itself. If a new ID is created, it should be captured by the user who wishes to exploit this reliability interface. The ID in use, whether created or passed as input, will be written to the optional output file when provided, as well as to the standard error output unless the *quiet mode* is in effect.

If a user is unsure whether a job was submitted successfully, he should resubmit using the same ID as was used for the previous attempt.

-I, -submission-id ID — If supplied, this option causes the job to be submitted using the given ID in the reliability protocol.

-If, -submission-id-file filename — If supplied, this option causes the ID to be read from the given file. It is an error to use both mechanisms to provide an input ID.

-Io, -submission-id-output-file filename — If supplied, the ID in use is written to the given file, whether this ID was provided by the user or given by one of the above input options.

Job EPR output

A successful submission will create a new ManagedJob resource with its own unique EPR for messaging. The globusrun-ws tool will output this EPR to a file when requested and as the sole standard output when running in batch mode. When running in streaming output mode, it is possible that the EPR will not be output and the user's only recourse is to submit again with the same submission ID and job request in order to reattach to the existing job.

-o, -job-epr-output-file filename — If supplied, the created ManagedJob EPR will be written to the given file following successful submission. The file will not be written if the submission fails.

Delegation

The job description supports the optional identification of delegated credentials for use by the WS GRAMservices. These features are passed through globusrun-ws without modification. However, globusrun-ws can also perform del-

egation and construct these optional request elements before submitting it to the service. The only delegation performed by default (if an endpoint does not already exist) is the multijob level jobCredential.

-J, -job-delegate
: If supplied AND the job description does not already provide a jobCredential element, globusrun-ws will delegate the client credential to WS GRAMand introduce the corresponding element to the submission input.

-S, -staging-delegate
: If supplied AND the job description does include staging or cleanup directives AND the job description does not already provide the necessary stagingCredential or transferCredential element(s), globusrun-ws will delegate the client credential to WS GRAM and RFT, and introduce the corresponding elements to the submission input.

  This option is implied by -streaming

-Jf, -job-credential-file filename:
: If supplied AND the job description does not already provide a jobCredential element, globusrun-ws will copy the supplied epr into the job description. This should be an epr returned from the DelegationFactoryService intended for use by the job (or, in the case of a multijob, for authenticating to the subjobs).

  note: for multijob descriptions, only the top level jobCredential will be copied into.

-Sf, -staging-credential-file filename:
: If supplied AND the job description does not already provide a stagingCredential element, globusrun-ws will copy the supplied epr into the job description. This should be an epr returned from the DelegationFactoryService intended for use with the RFT service associated with the ManagedJobService.

  note: this option is ignored for multijobs.

-Tf, -transfer-credential-file filename:
: If supplied, globusrun-ws will copy the epr into each of the stage in, stage out, and cleanup elements that do not already contain a transferCredential element. This should be an epr returned from the DelegationFactoryService intended for use by RFT to authenticate with the target gridftp server.

  note: this option is ignored for multijobs.

Lifetime

The ManagedJob resource supports lifetime management in the form of a scheduled destruction. The default lifetime requested by the client is infinite, subject to server policies.

-term, -termination mm/dd/yyyy HH:MM
: Set an absolute termination time.

-term, -termination +HH:MM
: Set a termination time relative to the successful creation of the job. The default is +24:00

Validate options

-validate
: The -validate command checks the job description for syntax errors and a subset of semantic errors without making any service requests.

Job description

-f, -job-description-file filename This option causes the job description to be read from the given file. This description is checked for validity.

Monitor options

-monitor The -monitor command attaches to an existing job in interactive or interactive-streaming output modes.

Job

Addressing information for the ManagedJob target of this command must be provided.

-j, -job-epr-file filename If supplied, this option causes the EPR for the ManagedJob to be read from the given file. This EPR is used as the endpoint for service requests.

Output mode

In the default *interactive mode*, the tool keeps running in order to monitor job status. In the optional *interactive-streaming mode*, the job output files are fetched and output from globusrun-ws as well.

-s, -streaming See Output mode under Submit Options above for details on streaming.

Status options

-status The -status command reports the current state of the job and exits. See the External States of the Managed Job Services[2] section of the developer guid for information on valid job states.

See the Job options for the -monitor command.

Kill options

-kill The -kill command requests the immediate cancellation of the job and exits.

Help options

-help Outputs an overview of the commands and features of the command.

Usage options

-usage Outputs brief usage information for the command.

Version options

-version Outputs version information for the command.

Job Handling

For every job that globusrun-ws delegates a credential, globusrun will augment the user's job description, adding annotations that will later tell globusrun-ws to destroy the credential after the job has been destroyed. Below are 2 job annotation examples. globusrun-ws only delegated the job cred...

[2] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/developer-index.html#s-wsgram-developer-archdes-states-external

```
<extensions>
<globusrunAnnotation>
<automaticJobDelegation>true</automaticJobDelegation>
<automaticStagingDelegation>false</automaticStagingDelegation>
<automaticStageInDelegation>false</automaticStageInDelegation>
<automaticStageOutDelegation>false</automaticStageOutDelegation>
<automaticCleanUpDelegation>false</automaticCleanUpDelegation>
</globusrunAnnotation>
</extensions>
```

globusrun-ws delegated the job, staging and stage in cred...

```
<extensions>
<globusrunAnnotation>
<automaticJobDelegation>true</automaticJobDelegation>
<automaticStagingDelegation>true</automaticStagingDelegation>
<automaticStageInDelegation>true</automaticStageInDelegation>
<automaticStageOutDelegation>false</automaticStageOutDelegation>
<automaticCleanUpDelegation>false</automaticCleanUpDelegation>
</globusrunAnnotation>
</extensions>
```

Environment

X509_USER_PROXY Overrides the default selection of user credentials when using GSI security.

Exit Codes

The client returns negative error codes for client errors, 0 for success, and positive error codes from the submitted job (where possible)

Name

managed-job-globusrun -- (DEPRECATED) Java-based job submission client for GRAM

`managed-job-globusrun`

Tool Description

Warning

This tool has been deprecated in this version and is only documented here because it may be useful for some testing purposes only.

Use globusrun-ws instead.

`managed-job-globusrun` is a Java-based job submission tool for the [WS] GRAM (i.e. it is a program for submitting jobs to a local or remote host and managing those jobs via the GRAM services). GRAM services provide secure job submission to many types of job schedulers for users who have the right to access a job hosting resource in a Grid environment. All GRAM job submission options are supported transparently through the embedded request document input. In fact, the job startup is done by submitting a client-side provided job description (*RSL* document in GT2). to the GRAM services.

In addition to starting jobs, it is possible to delegate credentials needed for certain optional GRAM features, query the state of a previously started job and parse a job description file without making any submission. Online and batch submission modes are supported with reattachment (recovery) for jobs whether they were started with this client or another GRAM client application.

Note: the existence of a valid proxy is required for essentially all supported operations but job description file parsing (**-p**). In order to generate a valid proxy file, use the grid-proxy-init tool available under $GLOBUS_LOCATION/bin.

Command Syntax

Arguments

```
managed-job-globusrun [options] [<factory>] <job description>
                managed-job-globusrun -p -file <job description filename>
                managed-job-globusrun (-state | -release | -kill) <job handle>
                managed-job-globusrun -help | -usage | -version
```

with

```
<job description>     = -file <job description filename> | <command line>
                <factory> = -factory <contact> [-type <type>]
                <contact> = [<protocol>://]<host>[:[port]][/<service>]
                [options] = [-q] [-n]
                [-b] [-duration] [-terminate-at]
                [-auth <auth>] [-xmlsec <sec>] [-personal]
                [-submission-id <ID>]
```

Options

Table 14. Options for managed-job-globusrun

Help options	
-help	Displays help information about the command.
-usage	Displays usage of the command.
-v, -version	Displays version of the command.
Job Factory Contact options	
-factory <contact>	Specifies the URL of the Job Factory Service to contact when submitting or listing jobs. A factory contact string can be specified in the following ways: • host • host: • host:port • host:port/service • host/service • host:/service It is also possible to specify the protocol by prepending **protocol://** to each of the previous possibilities, bringing the total number of supported syntaxes to 12. For those factory contacts which omit the protocol, port or service field, the following default values are used, as the following table explains: URL part default value port 8080 protocol http service /wsrf/services/ManagedJobFactoryService Omitting altogether the -factory option is equivalent to specifying the local host as the contact string (with the implied default protocol, port and service).
-type <factory type>	Specifies the type of factory resource to use. This is the name of the local resource manager. The default value is *Fork*.
Job Specification options	
<command line>	Creates a simple job description that only consists of a command line of the form: `'executable (argument)*'` Quotes must be used if there is one or more arguments.
-file <job description filename>	Reads job description from the local file <job description filename>. The job description must be a single job request.
-p	This option only parses the job description, and then prints either a success message or a parser failure. No job will be submitted to any factory service. The job description must be a single job request.

Batch Operations options	
-b, -batch	Do not wait for started job to complete (and do not destroy started job service on exit.) The handle of the job service will be printed on the standard output. This option is incompatible with multi-request jobs. Implies -quiet.
-state <handle>	Print out the state of the specified job. For a list of valid states, see the GRAM documentation [need link]; the current valid states are Pending, Active, Done, Suspended, and Failed. The handle may need to be quoted.
-r, -release <handle>	release the specified job from hold. The handle may need to be quoted.
-k, -kill <handle>	Kill the specified job. The handle may need to be quoted. Note: The <handle> argument is printed out when executing in batch mode or when using the -list option.
Job Resource Lifetime options	
-duration <duration>	Specify the duration of the job resource. The job resource will destroy itself automatically after the specified duration starting from service creation. • Format: HH:mm • Default: 24 hours. Incompatible with -terminate-at. Useful with -batch.
-terminate-at <date>	Specify the termination date/time of the job resource. Same as -duration but with an absolute date/time value. • Format: MM/dd/yyyy HH:mm • Default: see -duration. The date expression may need to be quoted, as in: `-terminate-at '08/15/2005 11:30'` Incompatible with -duration. Useful with -batch.
Security options	

-auth <auth>	Set authorization type. Usually, secure communication includes mutual authentication. In addition to the service authorizing the client for the requested operation(s), an authorization decision is made by the client to determine whether the remote service is the one intended. Depending on the configured authorization type of the GRAM services (which by default is 'host'), the user must select a corresponding client-side authorization type <auth>. <auth> can be: • *host* for host authorization (default): the GSI "host authorization" rule is used to verify that the service is using a host credential appropriate for the underlying service address information. This is the default. • *self* for self authorization: the GSI "self authorization" rule is used to verify that the service is using a (proxy) credential derived from the same identity as the client's. • an <id> for identity authorization: the service must be using a credential with the exact subject name provided.
-xmlsec <sec>	Set message protection level. <sec> can be: • *sig* for XML Signature (default) • *enc* for XML Encryption.
-personal	Shortcut for -auth self.
-proxy <proxy file>	Use <proxy file> instead of the default proxy credential file.
-deleg <deleg>	Set delegation type. <deleg> can be: • *limited* for limited delegation (default). • *full* for full delegation • *none* for no delegation
Miscellaneous options	
-q, -quiet	Switch quiet mode on, i.e. do not print diagnostic messages when job state changes, in non-batch mode. Disabled by default.
-n, -no-interrupt	Disable interrupt handling. By default, interrupt signals (typically generated by *Ctrl + C*) cause the program to terminate the currently submitted job. This flag disables that behavior.
-timeout <integer>	Set timeout for HTTP socket, in milliseconds. Applies to job submission only. The default value is 120000.

-submission-id <ID>	Set the submission ID of a previous job submission for which no server response was received. The ID can be used after an attempted job submission in order to recover the handle to the job.
GT2 globusrun options NOT functional (yet)	
-l, -list	NOT IMPLEMENTED ON SERVER SIDE YET. List previously started and not destroyed job services for this user. The output of this command consists of the handles and job description of the submitted jobs. Requires the -factory <URL> argument.
-dryrun	NOT IMPLEMENTED ON SERVER SIDE YET. Augment the job description in order to mark this job as a dry run, if the job description does not already say so. This causes the job manager to stop short of starting the job, but still detect other job description errors (such as bad directory, bad executable, etc). An error message will be displayed if the dry run fails. Otherwise, a message will be displayed indicating that the dryrun was successful.
-authenticate-only	NOT IMPLEMENTED ON SERVER SIDE YET.

New Functionality

Substitution variables

In GT 3.9.2, job description substitution variables had been removed from GRAM. Starting with GT 3.9.5, substitution variables are available again, while preserving the simplicity of the job description XML schema (relative to the GT3.2 job description schema). Substitution variables can be used in any path-like string or URL specified in the job description. They are special strings that are replaced by the GRAM services with actual values that the client-side does not *a priori* know. An example of substitution variable is **${GLOBUS_USER_HOME}**, which represents the path to the HOME directory on the file system visible by the GRAM services of the user on behalf of whom the job is executed.

Details are in job description doc[1]

Submission ID

A submission ID may be used in the GRAM protocol for robust reliability in the face of message faults or other transient errors in order to ensure that at most one instance of a job is executed, i.e. to prevent accidental duplication of jobs under rare circumstances with client retry on failure. The managed-job-globusrun tool always uses this feature, requiring either a submission ID to be passed in as input or a new unique ID to be created by the tool itself. If a new ID is created, it should be captured by the user who wishes to exploit this reliability interface. The ID in use, whether created or passed as input, will be written to the first line of standard output unless the *quiet mode* is in effect.

If a user is unsure whether a job was submitted successfully, he should resubmit using the same ID as was used for the previous attempt.

[1] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html

Job hold and release

It is possible to specify in a job description that the job be put on hold when it reaches a chosen state (see GRAM Approach[2] documentation for more information about the executable job state machine, and the job description XML schema documentation for information about how to specify a held state[3] This is useful for instance when a GRAM client wishes to directly access output files written by the job (as opposed to waiting for the stage-out step to transfer files from the job host). The client would request that the file cleanup process be held until released, giving the client an opportunity to fetch all remaining/buffered data after the job completes but *before* the output files are deleted.

Note that the hold feature of the GRAM service interface is not exploited by the current Java version of the client tool, but will be in the C client in order to implement client-side streaming of remote stdout/err.

The current client tool does however

- automatically release a job remotely in interactive mode if the job is being held at any given state
- offer an option (**-release**) for the user to release a job previously submitted in batch mode.

MultiJobs

The new job description XML schema allows for specification of a MultiJob i.e. a job that is itself composed of several executable jobs (those jobs cannot be multijobs, so the structure is not recursive). This is useful in order to bundle a group of jobs together and submit them as a whole to a remote GRAM installation.

Note that there is no specification of relationships between the executable jobs, which we will refer to as "subjobs". The subjobs are submitted to job factory services in their order of appearance in the multijob description.

Job and process rendezvous

This version of GRAM offers a mechanism to perform synchronization between job processes in a multiprocess job and between subjobs in a multijob. The job application can in fact register binary information, for instance process information or subjob information, and get notified when all the other processes or subjobs have registered their own information. This is for instance useful for parallel jobs which need to rendezvous at a "barrier" before proceeding with computations, in the case when no native application API is available to help do the rendezvous.

Limitations

With the porting of existing GRAM functionality from OGSI to WSRF, this new version of the job submission tool suffers from a few limitations comparatively to previous versions of the tool. These limitations will be dealt with in the next version of the tool, which will be implemented in C and thus will be better performing.

No more file staging using GASS

The GASS server is not being used anymore by GRAM, so the options -server and -write have been removed. Instead, file staging is done in a reliable fashion via RFT and GridFTP servers. file staging in GT 4.0 GRAM[4]

No standard output redirection yet

Unlike the GT3.2 managed-job-globusrun used with the option -output, this version of the tool does not offer any streamed redirection of the standard streams. This is because the GASS server is not used anymore by GRAM. Instead,

[2] ../WS_GRAM_Approach.html
[3] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/mj_types.html
[4] #specifyingstaging

a future version of the tool will allow for streaming of any server-side file (including the standard streams of the job execution) using GridFTP "tailing" of remote files.

No listing of submitted jobs yet

The -list option, which made the 3.2 tool print the identifiers of the jobs submitted by the user on the standard output, is not available in this version of the tool.

Tool behavior for some features

Tool-triggered automatic job resource destruction

Execution errors and user interrupt events are handled by automatically destroying the requested job service(s), unless the -batch option is on the command-line. The -batch option prevents the tool from listening to job state changes and from waiting for the job to finish. If -batch is selected, the command will return as soon as the remote job has been submitted.

The behavior of the tool with respect to job service destruction will vary in response to several kinds of events:

- The command exits normally after the job(s) finish(es), and destroys the job service(s) it requested. In batch mode, the requested job is never destroyed.
- The command is terminated in response to a user interrupt, such as typing *Ctrl + C*, or a system-wide event, such as user logoff or system shutdown. If the -no-interrupt option is on the command-line, and the command-line has been successfully parsed when the interrupt occurs, the tool does not destroy any job service(s) it requested. Otherwise the tool destroys the requested job service(s).
- In case of any error of execution, the command will exit and destroy the job(s) it successfully requested.

If the Java virtual machine of the tool aborts, that is, stops running without shutting down cleanly, for instance because it received a SIGKILL signal on Unix, then no guarantee can be made about whether or not the job service(s) will be destroyed.

Note: the shutdown behavior explained above cannot be guaranteed if the JVM option -Xrs is entered. The recommended way to disable service destruction is to specify the -batch option on the command-line.

Credential delegation

Single job submission

managed-job-globusrun inserts references to newly delegated credentials in the job description before submitting it. In order to do so, it obtains endpoint references to resources representing delegated credentials by passing a proxy credential (user supplied or default) to the Globus delegation services. The resulting EPRs are then inserted in the job description before submission. The possible elements where the EPR are added are: as the value of **jobCredentialEndpoint** and **stagingCredentialEndpoint**, in order to secure calls to the GRAM and RFT factories, and inside each individual RFT directive, i.e. inside the **fileStageIn**, **fileStageOut** and **fileCleanUp** elements. See the job description doc[5] for details about these attributes. The Managed Executable Job uses the endpoints in the job description to fetch the credentials from the Delegation services and use them as needed on behalf of the job.

[5] schemas/gram_job_description.html

MultiJob submission

managed-job-globusrun delegates full credentials to the delegation service for the multijob, then processes each single job as stated in the single job submission case.

If several subjobs are to use the same delegation service, then only one credential will be delegated to that delegation service, i.e. the same credential will be used for several jobs.

How to do common job submission tasks

Submitting a job in interactive mode

A very simple command-line can be used to submit a job. For instance, the following command-line submits a job to the GRAM services hosted on the same machine (assuming a Globus container is running of course):

```
% bin/managed-job-globusrun "/bin/echo Testing 1...2...3"
```

The output should look like:

```
Submission ID: uuid:661AA7F0-2573-11D9-99B2-D4755757F903
WAITING FOR JOB TO FINISH
========== State Notification ==========
Job State: Active
========================================
========== State Notification ==========
Job State: CleanUp
========================================
========== State Notification ==========
Job State: Done
========================================
Exit Code: 0
DESTROYING SERVICE
SERVICE DESTROYED
```

Note: the job state notifications are printed in the order of arrival, but they may arrive at the client-side in *any order*.

In this example the job description specifies the standard output stream path of the job to be: ${GLOBUS_USER_HOME}/stdout. The GRAM services replace the substitution variable ${GLOBUS_USER_HOME} with the path to the Home directory of the submitting user as seen by the machine were the invoked GRAM services are hosted. You can thus verify the output of the job with the following command:

```
% cat ~/stdout
```

which will display the string:

```
12 abc 34 pdscaex_instr_GrADS_grads23_28919.cfg pgwynnel was here
```

Submitting a job in batch mode, checking its status and destroying the resource

To submit a job without having the client wait for job completion, specify the option **-batch** (or **-b**) on the command-line:

```
% bin/managed-job-globusrun -batch "/bin/echo Testing 1...2...3"
Warning: Will not wait for job completion, and will not destroy job servic
Submission ID: uuid:9C715240-26C7-11D9-850A-ABE2020F9ED6
CREATED MANAGED JOB SERVICE WITH HANDLE:
http://127.0.0.1:8080/wsrf/services/ManagedExecutableJobService?9C715240-2
```

To check the status of the job, use the -state option:

```
% bin/managed-job-globusrun
-state 'http://127.0.0.1:8080/wsrf/services/ManagedExecutableJobService?9C
Job State: Done
```

To destroy the job resource created on the server side, use the -kill option:

```
% bin/managed-job-globusrun
-kill 'http://127.0.0.1:8080/wsrf/services/ManagedExecutableJobService?9C7
DESTROYING SERVICE
SERVICE DESTROYED
```

Finding which schedulers are interfaced by the WS GRAM installation

Unfortunately there is no option yet to print the list of local resource managers supported by a given WS-GRAM service installation. But there is a way to check, whether WS-GRAM supports a certain local resource manager or not. The following command gives an example of how a client could check if Condor is available at the remote site:

```
wsrf-query \
    -s https://<hostname>:<port>/wsrf/services/ManagedJobFactoryService
    -key {http://www.globus.org/namespaces/2004/10/gram/job}ResourceID Condor \
    "//*[local-name()='version']"
```

Replace host and port settings with the values you need. If Condor is available on the server-side, the output should look something like the following:

```
 <ns1:version xmlns:ns1="http://mds.globus.org/metadata/2005/02">4.0.3</ns1:version>
```

In this example the output indicates, that a GT is listening on the server-side, that Condor is available and that the GT version is 4.0.3. If no GT at all is running at the specified host and/or port or if the specified local resource manager is not available on the server-side, the output will be an error message.

On the server-side the *GRAM name*of local resource managers for which GRAM support has been installed can be obtained by looking at the GRAM configuration on the GRAM, as explained here[6] The GRAM name of the local resource manager can be used with the `-type` option to specify which factory resource to use when submitting a job. For instance:

[6] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html#s-wsgram-Interface_Config_Frag-managerconfig

```
% bin/managed-job-globusrun -type Fork /bin/true
```

will submit a `/bin/true` job to the **Fork** local resource manager (i.e. the command-line `/bin/true` will simply be executed as a newly spawn process)

```
% bin/managed-job-globusrun -type LSF /bin/true
```

will submit a `/bin/true` job to the *LSF* scheduler (if installed).

```
% bin/managed-job-globusrun -type Multi -file simple_multi_job.xml
```

where simple_multi_job.xml contains the description of a multijob[7] will submit a multi job to the **Multi ManagedJob-Factory** resource.

Specifying file staging in the job description

In order to do file staging one must add specific elements to the job description. The file transfer directives follow the RFT syntax RFT syntax[8] ,which enables third-party transfers. Each file transfer must therefore specify a source URL and a destination URL. URLs are specified as GridFTP URLs (for remote files) or as file URLs (for local files).

For instance, in the case of staging a file *in*, the source URL would be a GridFTP URL (for instance **gsiftp://job.submitting.host:2888/tmp/mySourceFile**) resolving to a source document accessible on the file system of the job submission machine (for instance **/tmp/mySourceFile**). At run-time the Reliable File Transfer service used by the GRAM service on the remote machine would fetch the remote file using the GridFTP protocol and write it reliably to the specified local file (for instance **file:///${GLOBUS_USER_HOME}/my_transfered_file**, which resolves to **~/my_transfered_file**). Here is how the stage-in directive would look like:

```
<fileStageIn>
<transfer>
<sourceUrl>gsiftp://job.submitting.host:2888/tmp/mySourceFile</sourceU
<destinationUrl>file:///${GLOBUS_USER_HOME}/my_transfered_file</destin
</transfer>
</fileStageIn>
```

Note: additional RFT-defined quality of service requirements can be specified for each transfer. See the RFT documentation for more information.

Here is an example job description with file stage-in and stage-out:

```
<job>
<executable>my_echo</executable>
<directory>${GLOBUS_USER_HOME}</directory>
<argument>Hello</argument>
```

[7] #specifyingmultijob
[8] schemas/rft_types.html

```
<argument>World!</argument>
<stdout>${GLOBUS_USER_HOME}/stdout</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr</stderr>
<fileStageIn>
<transfer>
<sourceUrl>gsiftp://job.submitting.host:2888/bin/echo</sourceUrl>
<destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
</transfer>
</fileStageIn>
<fileStageOut>
<transfer>
<sourceUrl>file://${GLOBUS_USER_HOME}/stdout</sourceUrl>
<destinationUrl>gsiftp://job.submitting.host:2888/tmp/stdout</destinat
</transfer>
</fileStageOut>
<fileCleanUp>
<deletion>
<file>file://${GLOBUS_USER_HOME}/my_echo</file>
</deletion>
</fileCleanUp>
</job>
```

The submission of this job to the GRAM services causes the following sequence of actions:

1. The **/bin/echo** executable is transfered from the submission machine to the GRAM host file system. The destination location is the HOME directory of the user on behalf of whom the job is executed by the GRAM services (see <**fileStageIn**>).

2. The transfered executable is used to print a test string (see <**executable**>, <**directory**> and the <**argument**> elements) on the standard output, which is redirected to a local file (see <**stdout**>).

3. The standard output file is transfered to the submission machine (see <**fileStageOut**>).

4. The file that was initially transfered during the stage-in phase is removed from the file system of the GRAM installation (see <**fileCleanup**>).

Specifying and submitting a MultiJob

Within the multijob description, each subjob description must come along with an endpoint for the factory to submit the subjob to. This enables the at-once submission of several jobs to different hosts. The factory to which the multijob is submitted acts as an intermediary tier between the client and the eventual executable job factories. See the job description schema documentation[9] for more information about multijob specification.

A multijob must be submitted to a **Multi** job factory resource:

```
% bin/managed-job-globusrun -type Multi -file myMultiJob.xml
```

A multijob resource is created by the factory and exposes a set of WSRF resource properties different than the resource properties of an executable job. The state machine of a multijob is also different since the multijob represents the overall execution of all the executable jobs it is composed of.

[9] schemas/gram_job_description.html

Troubleshooting

Job Execution Errors

fault types[10]

Common issues

Expired credentials

Symptom: the client output shows an error related to expired credentials, as in:

```
Error: error submitting job request: ; nested exception is:
javax.xml.rpc.soap.SOAPFaultException: Expired credentials
(O=Grid,OU=GlobusTest,OU=simpleCA.foo.bar.com,OU=bar.com,CN=John Doe,C
```

Solution: use the $GLOBUS_LOCATION/bin/grid-proxy-init tool to create a new proxy file:

```
% bin/grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-foo.bar.com/OU=bar.co
Enter GRID pass phrase for this identity:
Creating proxy ............................... Done
Your proxy is valid until: Tue Oct 26 01:33:42 2004
```

Socket timeout error

Symptom: the client output shows a timeout error when waiting for the response from the GRAM service(s):

```
Error: error submitting job request: ; nested exception is:
java.net.SocketTimeoutException: Read timed out
```

Solution: re-submit the job with a higher delay before HTTP socket timeout than the default. Use the `-timeout` option of `managed-job-globusrun`, as in:

```
% bin/managed-job-globusrun -timeout 240000 -f myJob.xml
```

Connection refused to postmaster

Symptom: the server log and client output show exception stack traces with the following message:

```
Unable to create RFT Resource; nested exception is:
org.apache.commons.dbcp.DbcpException: Connection refused. Check that
```

[10] schemas/mj_faults.html

```
and port are correct and that the postmaster is accepting TCP/IP conne
```

This error indicates a lack of configuration for RFT. *Solution:* See RFT Configuration Docs[11]

Lack of authorization for the user's Distinguished Name

Symptom: the server log and client output show exception stack traces with the following message:

```
Error: error submitting job request:
org.globus.wsrf.impl.security.authorization.exceptions.AuthorizationEx
(pdp08) "/O=Grid/OU=GlobusTest/OU=simpleCA-foo.bar.com/OU=foo.bar.com/
is not authorized to use operation:
{http://properties.impl.wsrf.globus.org}getMultipleResourceProperties
```

This error indicates a lack of authorization for the Distinguished Name (DN) reported in the error message. This means that according to the gridmap configuration for the toolkit, this user has not been authorized to call the operation reported in the error message.

Solution: Add an entry for the user's DN to the gridmap file. See the GRAM configuration documentation[12]

File(s) Not Found warnings

Symptom: the server LOG displays messages at WARN severity such as:

```
[Thread-3] WARN  factory.ManagedJobFactoryResource [getRestartTimestam
java.io.FileNotFoundException: /software/globus/gt4/rc4.0.0/var/globus
[Thread-3] WARN  factory.ManagedJobFactoryResource [getRestartTimestam
java.io.FileNotFoundException: /software/globus/gt4/rc4.0.0/var/globus
[Thread-2] WARN  utils.XmlPersistenceHelper [load:185] [CORE] File
/nfs/v5/alain/.globus/persisted/128.9.72.67/ManagedExecutableJobResour
for resource {http://www.globus.org/namespaces/2004/10/gram/job}Resour
was not found
```

Solution: the log messages above are harmless and are not indicative of any problem in the behavior of the GRAM service. They can be ignored.

Known problems

Client Hanging Forever

Symptom: in interactive (i.e. non-batch) mode, the managed-job-globusrun client seems to be stuck waiting for additional job state notifications.

Solution: This is a known problem which can happen sometimes.

Possible solution: remove the timestamp files in $GLOBUS_LOCATION/var:

[11] ../../data/rft/admin-index.html#s-rft-admin-configuring
[12] admin-index.html#s-wsgram-admin-nondefaultgridmap

```
% rm var/globus-jsm-*.stamp
```

Restart the container.

If you decide to report the issue, please provide the job description and submission command-line as well as a full server-side GRAM log so we can determine the cause of the problem:

1. Edit $GLOBUS_LOCATION/log4j.properties to add **exec=DEBUG**.
2. Restart container and execute the same job submission command-line.
3. Submit full GRAM server LOG to support list.

NotRegisteredException ERROR log message

Symptom: the following message appears in the server log:

```
[Thread-7] ERROR jobmanager.JobManager [unsubscribeForNotifications:17
org.globus.exec.monitoring.NotRegisteredException
at org.globus.exec.monitoring.JobStateMonitor.unregisterJobID(JobState
at org.globus.exec.service.job.jobmanager.JobManager.unsubscribeForNot
at org.globus.exec.service.job.jobmanager.JobManager.processState(JobM
at org.globus.exec.service.job.jobmanager.RunQueue.run(RunQueue.java:7
```

Solution: this is typically harmless and can be ignored.

Name

globus-job-run-ws -- Interactive job submission script for WS GRAM (requires globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[1])

`globus-job-run-ws` GLOBAL OPTIONS
`GLOBAL OPTIONS:`
[-help|-usage] [-version] [-verbose] [-dumpcmd] [-dumpjdd] [-o|-job-epr-output-file *`filename`*] [-contact *`factory contact`*] [-factory-type *`ResourceID`*] [-z|-authorization *`authz`*] [-delegation-epr-file *`EPR file`*] [-gridftp *`URL`*] [-jobid *`job ID`*] [-np|-count *`N`*] [-host-count *`N`*] [-max-time *`minutes`*] [-max-memory *`MB`*] [-min-memory *`MB`*] [-project *`project ID`*] [-queue *`queue ID`*] [-dir *`directory`*] [-job-type *`job type`*] [-stdin *`file`*] [-stdout *`file`*] [-stderr *`file`*] [-library-path *`path`*...] [-env *`name=value[ name=value]...`*] [-args *`arg[ arg]...`*] {[HOST CLAUSE] | [-: *`OPTIONS`* HOST CLAUSE[-: *`OPTIONS`* HOST CLAUSE].....]}
`OPTIONS:`
[-factory-type *`ResourceID`*] [-np|-count *`N`*] [-host-count *`N`*] [-max-time *`minutes`*] [-max-memory *`MB`*] [-min-memory *`MB`*] [-project *`project ID`*] [-queue *`queue ID`*] [-dir *`directory`*] [-job-type *`job type`*] [-stdin *`file`*] [-stdout *`file`*] [-stderr *`file`*] [-env *`name=value[ name=value]...`*]
`HOST CLAUSE:`
contact [[-l] | [-s]] executable *`arg[ arg]`*...

Description

Important

This package requires the globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[2].

`globus-job-run-ws` is a wrapper script around `globusrun-ws`. It makes it easier to submit interactive jobs with streaming output. Interactive jobs are defined as those that are submitted and actively monitored by the client for state changes. The client will not exit until the job is finished and the output has been streamed back. The job is also automatically destroyed and it's scratch space cleaned up before the client exits.

Command options

Global Options

The global options are used to 1) affect the script execution, 2) control the job submission, and 3) provide defaults for subjobs when using the multijob syntax.

-help|-usage	Prints the command usage and exits.
-version	Prints the version of the client program and exits.
-verbose	Prints extra script execution debug messages.
-dumpcmd	Prints the actual `globusrun-ws` command that would have been executed and exits.
-dumpjdd	Prints the actual Job Description Document that would have been passed to `globusrun-ws` and exits.

[1] http://www.globus.org/toolkit/downloads/development/
[2] http://www.globus.org/toolkit/downloads/development/

-o|-job-epr-output-file	Causes the job's EPR to be written to the specified file.
-contact	Specifies the factory contact string for where to submit the multijob. This argument is ignored if there is only one single job host clause (i.e. no '-:' arguments).
-factory-type	Specifies the factory resource ID for the target factory service. Supported values are "Fork", "Multi", "*Condor*", "*LSF*", and "*PBS*". For single jobs the the default is "Fork". For multijobs the default is "Multi".
-z|-authorization	Specifies the type of authorization to do against the service. Valid values are "host", "self", or a certificate subject (i.e. identity authorization). The default is "host" if not specified.
-delegation-epr-file	Specifies a file with an EPR to a delegated credential that is to be used for the job, staging, and transfer credential. If you need more flexibility over these different credentials separately, please use `globusrun-ws` directly.
-gridftp	Specifies the base URL for the GridFTP server used local to the client for staging files to and from the server.
-jobid	Specifies the ID of the job. This overrides the automatically generated *UUID*.
-np|-count	Specifies the number of processes to spawn for this job.
-host-count	Specifies the number of processing elements to employ for running the job process(es).
-max-time	Specifies the maximum amount of time the scheduler should allow for running the job process(es).
-max-memory	Specifies the maximum amount of memory the scheduler should allow for running the job process(es).
-min-memory	Specifies the minimum amount of memory the job needs to run.
-project	Specifies the project ID associated with the job.
-queue	Specifies the target queue that the job should be placed in when the job is submitted to the scheduler.
-dir	Specifies the working directory for the job process(es).
-job-type	Specifies the type of job this is. Valid values are "single", "multiple", "condor", and "mpi". The default value is "multiple".
-stdin	Specifies the stdin input file path local to the job process(es).
-stdout	Specifies the stdout output file path local to the job process(es).
-stderr	Specifies the stderr output file path local to the job process(es).
-library-path	Specifies the a path to be added to the library path of the job. This argument can be repeated.
-env	Specifies name/value pairs separated by '=' to be used as environment variable assignments for the job process(es). The name/value pair values can be repeated

until a dashed argument is specified or there are only two arguments left on the command line (i.e. ending host clause).

-args — Specifies default command arguments for the subjobs. The values can be repeated until a host clause is encountered. Arguments specified with the executable in the host clause are appended to these values.

Options

These options can be specified before each subjob host clause when using the multijob syntax.

-factory-type — Specifies the factory resource ID for the target factory service. Supported values are "Fork", "Condor", "LSF", and "PBS". The default is "Fork". A value of "Multi" is not allowed in the subjob options because the service semantics do not allow this.

-np|-count — Specifies the number of processes to spawn for this job.

-host-count — Specifies the number of processing elements to employ for running the job process(es).

-max-time — Specifies the maximum amount of time the scheduler should allow for running the job process(es).

-max-memory — Specifies the maximum amount of memory the scheduler should allow for running the job process(es).

-min-memory — Specifies the minimum amount of memory the job needs to run.

-project — Specifies the project ID associated with the job.

-queue — Specifies the target queue that the job should be placed in when the job is submitted to the scheduler.

-dir — Specifies the working directory for the job process(es).

-job-type — Specifies the type of job this is. Valid values are "single", "multiple", "condor", and "mpi". The default value is "multiple".

-stdin — Specifies the stdin input file path local to the job process(es).

-stdout — Specifies the stdout output file path local to the job process(es).

-stderr — Specifies the stderr output file path local to the job process(es).

-library-path — Specifies the a path to be added to the library path of the job. This argument can be repeated.

-env — Specifies name/value pairs separated by '=' to be used as environment variable assignments for the job process(es). The name/value pair values can be repeated until a dashed argument is specified or there are only two arguments left on the command line (i.e. ending host clause).

Host Clause

The host clause specifies the service contact string and executable.

contact string — Specifies the target service. Valid forms of the contact string are as follows:

```
hostname
hostname:port
```

```
hostname:port/service
hostname/service
protocol://hostname
protocol://hostname:port
protocol://hostname:port/service
protocol://hostname/service
```

-l|-s	Specifies whether the executable is already local to the execution machine (-l) or should be staged (-s). The default is -l.
executable	The local path to the program or script to be executed.

Notes

Job stdin defaults to /dev/null. Job stdout/stderr defaults to "stdout/err" in the scratch directory. The file modifiers -l and -s specify different filespaces: -l[ocal] file is relative to working directory of job (DEFAULT) -s[tage] file relative to job request is staged to job host The working directory of the submitted job defaults to $HOME. More than one host clause can be specified, in which case the job runs on all of the hosts simultaneously. The host clause begins with '-:' for a multi-request, and the contact string of the remote resource for a single request. It allows any of the specified options in any order. The first unrecognized (non-flagged) term is treated as the executable, with arguments following immediately after. In the multi-request case, arguments may be specified specifically for each host-clause by appending them after the name of the executable, or job-wide with the -args flag. In either case, arguments continue until the end of the command-line or until the next '-:' flag signifying a new host-clause. Two '-:' flags in a row are interpreted as an escaped '-:' and the host- clause continues until the next unpaired '-:' flag or end of input. Except for -library-path, if more than one copy of an argument is specified (such as two -stdin definitions), the setting defined last on the command line will be used.

Name

globus-job-submit-ws -- Batch job submission script for WS GRAM (requires globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[1])

`globus-job-submit-ws` GLOBAL OPTIONS
`GLOBAL OPTIONS:`
[-help|-usage] [-version] [-verbose] [-dumpcmd] [-dumpjdd] -o|-job-epr-output-file *`filename`* [-contact *`contact string`*] [-factory-type *`ResourceID`*] [-z|-authorization *`authz`*] [-delegation-epr-file *`EPR file`*] [-gridftp *`URL`*] [-jobid *`job ID`*] [-np|-count *`N`*] [-host-count *`N`*] [-max-time *`minutes`*] [-max-memory *`MB`*] [-min-memory *`MB`*] [-project *`project ID`*] [-queue *`queue ID`*] [-dir *`directory`*] [-job-type *`job type`*] [-stdin *`file`*] [-stdout *`file`*] [-stderr *`file`*] [-library-path *`path`*...] [-env *`name=value[ name=value]...`*] [-args *`arg[ arg]...`*] {[HOST CLAUSE] | [-: *`OPTIONS`* HOST CLAUSE[-: *`OPTIONS`* HOST CLAUSE].....]}
`OPTIONS:`
[-np|-count *`N`*] [-host-count *`N`*] [-max-time *`minutes`*] [-max-memory *`MB`*] [-min-memory *`MB`*] [-project *`project ID`*] [-queue *`queue ID`*] [-dir *`directory`*] [-job-type *`job type`*] [-stdin *`file`*] [-stdout *`file`*] [-stderr *`file`*] [-env *`name=value[ name=value]...`*]
`HOST CLAUSE:`
contact string [[-l] | [-s]] executable *`arg[ arg]`*...

Description

Important

This package requires the globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[2].

`globus-job-submit-ws` is a wrapper script around `globusrun-ws`... It makes it easier to submit batch jobs. Batch jobs are defined as those that are submitted but not actively monitored for state changes. One must use `globus-run-ws` to check the job's state. Also, when the job is finished one can use `globus-job-get-output-ws` to fetch the output of the job and `globus-job-cliean-ws` to destroy the job and clean up the job's scratch space.

Command options

Global Options

The global options are used to 1) affect the script execution, 2) control the job submission, and 3) provide defaults for subjobs when using the multijob syntax.

-help|-usage	Prints the command usage and exits.
-version	Prints the version of the client program and exits.
-verbose	Prints extra script execution debug messages.
-dumpcmd	Prints the actual `globusrun-ws` command that would have been executed and exits.
-dumpjdd	Prints the actual Job Description Document that would have been passed to `globusrun-ws` and exits.

[1] http://www.globus.org/toolkit/downloads/development/
[2] http://www.globus.org/toolkit/downloads/development/

-o|-job-epr-output-file	Causes the job's EPR to be written to the specified file. This argument is not optional.
-contact	Specifies the factory contact string for where to submit the multijob. This argument is ignored if there is only one single job host clause (i.e. no '-:' arguments).
-factory-type	Specifies the factory resource ID for the target factory service. Supported values are "Fork", "Multi", "Condor", "LSF", and "PBS". For single jobs the the default is "Fork". For multijobs the default is "Multi".
-z|-authorization	Specifies the type of authorization to do against the service. Valid values are "host", "self", or a certificate subject (i.e. identity authorization). The default is "host" if not specified.
-delegation-epr-file	Specifies a file with an EPR to a delegated credential that is to be used for the job, staging, and transfer credential. If you need more flexibility over these different credentials separately, please use `globusrun-ws` directly.
-gridftp	Specifies the base URL for the GridFTP server used local to the client for staging files to and from the server.
-jobid	Specifies the ID of the job. This overrides the automatically generated UUID.
-np|-count	Specifies the number of processes to spawn for this job.
-host-count	Specifies the number of processing elements to employ for running the job process(es).
-max-time	Specifies the maximum amount of time the scheduler should allow for running the job process(es).
-max-memory	Specifies the maximum amount of memory the scheduler should allow for running the job process(es).
-min-memory	Specifies the minimum amount of memory the job needs to run.
-project	Specifies the project ID associated with the job.
-queue	Specifies the target queue that the job should be placed in when the job is submitted to the scheduler.
-dir	Specifies the working directory for the job process(es).
-job-type	Specifies the type of job this is. Valid values are "single", "multiple", "condor", and "mpi". The default value is "multiple".
-stdin	Specifies the stdin input file path local to the job process(es).
-stdout	Specifies the stdout output file path local to the job process(es).
-stderr	Specifies the stderr output file path local to the job process(es).
-library-path	Specifies the a path to be added to the library path of the job. This argument can be repeated.
-env	Specifies name/value pairs separated by '=' to be used as environment variable assignments for the job process(es). The name/value pair values can be repeated

until a dashed argument is specified or there are only two arguments left on the command line (i.e. ending host clause).

-args — Specifies default command arguments for the subjobs. The values can be repeated until a host clause is encountered. Arguments specified with the executable in the host clause are appended to these values.

Options

These options can be specified before each subjob host clause when using the multijob syntax.

-factory-type — Specifies the factory resource ID for the target factory service. Supported values are "Fork", "Condor", "LSF", and "PBS". The default is "Fork". A value of "Multi" is not allowed in the subjob options because the service semantics do not allow this.

-np|-count — Specifies the number of processes to spawn for this job.

-host-count — Specifies the number of processing elements to employ for running the job process(es).

-max-time — Specifies the maximum amount of time the scheduler should allow for running the job process(es).

-max-memory — Specifies the maximum amount of memory the scheduler should allow for running the job process(es).

-min-memory — Specifies the minimum amount of memory the job needs to run.

-project — Specifies the project ID associated with the job.

-queue — Specifies the target queue that the job should be placed in when the job is submitted to the scheduler.

-dir — Specifies the working directory for the job process(es).

-job-type — Specifies the type of job this is. Valid values are "single", "multiple", "condor", and "mpi". The default value is "multiple".

-stdin — Specifies the stdin input file path local to the job process(es).

-stdout — Specifies the stdout output file path local to the job process(es).

-stderr — Specifies the stderr output file path local to the job process(es).

-library-path — Specifies the a path to be added to the library path of the job. This argument can be repeated.

-env — Specifies name/value pairs separated by '=' to be used as environment variable assignments for the job process(es). The name/value pair values can be repeated until a dashed argument is specified or there are only two arguments left on the command line (i.e. ending host clause).

Host Clause

The host clause specifies the service contact string and executable.

contact string — Specifies the target service. Valid forms of the contact string are as follows:

```
hostname
hostname:port
```

```
hostname:port/service
hostname/service
protocol://hostname
protocol://hostname:port
protocol://hostname:port/service
protocol://hostname/service
```

-l|-s	Specifies whether the executable is already local to the execution machine (-l) or should be staged (-s). The default is -l.
executable	The local path to the program or script to be executed.

Notes

Job stdin defaults to /dev/null. Job stdout/stderr defaults to "stdout/err" in the scratch directory. The file modifiers -l and -s specify different filespaces: -l[ocal] file is relative to working directory of job (DEFAULT) -s[tage] file relative to job request is staged to job host The working directory of the submitted job defaults to $HOME. More than one host clause can be specified, in which case the job runs on all of the hosts simultaneously. The host clause begins with '-:' for a multi-request, and the contact string of the remote resource for a single request. It allows any of the specified options in any order. The first unrecognized (non-flagged) term is treated as the executable, with arguments following immediately after. In the multi-request case, arguments may be specified specifically for each host-clause by appending them after the name of the executable, or job-wide with the -args flag. In either case, arguments continue until the end of the command-line or until the next '-:' flag signifying a new host-clause. Two '-:' flags in a row are interpreted as an escaped '-:' and the host- clause continues until the next unpaired '-:' flag or end of input. Except for -library-path, if more than one copy of an argument is specified (such as two -stdin definitions), the setting defined last on the command line will be used.

Name

globus-job-get-output-ws -- Job output fetch script for WS GRAM (requires globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[1])

`globus-job-get-output-ws` [-help] [-version] [-verbose] [[-out] | [-err]] [-z|authorization *`authz method`*] <EPR file>

Description

Important

This package requires the globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[2].

`globus-job-get-output-ws` is a wrapper around `wsrf-get-property` and `globus-url-copy`. It is intended to fetch either the stdout or stderr of a batch job submitted with `globus-job-submit-ws`.

Command Options

-help	HELP!
-version	The script version.
-verbose	Verbose script execution.
-out, -err	Fetch either the stdout or stderr of the job.
-authorization	Either "host", "self", or a cert subject (identity authz).
EPR file	Specifies a path to a file containg a job EPR. This is usually the file written via the `-job-epr-output-file` argument to `globus-job-submit-ws`.

[1] http://www.globus.org/toolkit/downloads/development/
[2] http://www.globus.org/toolkit/downloads/development/

Name

globus-job-clean-ws -- Destroy and clean up a batch job for WS GRAM (requires globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[1])

`globus-job-clean-ws` [-help] [-version] [-verbose] [-dumpcmd] [-z|authorization *`authz method`*] <EPR file>

Description

Important

This package requires the globus_wsrf_gram_client_tools update package from http://www.globus.org/toolkit/downloads/development/[2].

`globus-job-clean-ws` is a very simple wrapper around the `globusrun-ws -kill` command.

Command Options

-help	HELP!
-version	The script version.
-verbose	Verbose script execution.
-dumpcmd	Dump the actual globusrun-ws command being executed.
-authorization	Either "host", "self", or a cert subject (identity authz).
EPR file	Specifies a path to a file containg a job EPR. This is usually the file written via the `-job-epr-output-file` argument to `globus-job-submit-ws`.

[1] http://www.globus.org/toolkit/downloads/development/
[2] http://www.globus.org/toolkit/downloads/development/

Chapter 17. Submitting a job in Java using WS GRAM

The following is a general scenario for submitting a job using the Java stubs and APIs. Please consult the Java WS Core API[1], Delegation API[2], Reliable File Transfer API[3], and WS-GRAM API[4] documentation for details on classes referenced in the code excerpts.

Also, it will probably be helpful to look at the GramJob class source code[5] as a functioning example.

1. Class imports

The following imports will be needed for these examples:

```
import java.io.File;[6]
import java.io.FileInputStream;[7]
import java.net.URL;[8]
import java.util.LinkedList;[9]
import java.util.List;[10]
import java.util.Vector;[11]
import java.security.cert.X509Certificate;[12]
import javax.xml.rpc.Stub;[13]
import javax.xml.soap.SOAPElement;[14]
import org.apache.axis.components.uuid.UUIDGenFactory;[15]
import org.apache.axis.message.addressing.AttributedURI;[16]
import org.apache.axis.message.addressing.EndpointReferenceType;[17]
import org.globus.delegation.DelegationUtil;[18]
import org.globus.exec.generated.CreateManagedJobInputType;[19]
import org.globus.exec.generated.CreateManagedJobOutputType;[20]
import org.globus.exec.generated.JobDescriptionType;[21]
import org.globus.exec.generated.ManagedJobFactoryPortType;[22]
```

[1] http://www.globus.org/toolkit/docs/4.0/common/javawscore/Java_WS_Core_Public_Interfaces.html#apis
[2] http://www.globus.org/toolkit/docs/4.0/security/delegation/WS_AA_Delegation_Service_Public_Interfaces.html#apis
[3] http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Public_Interfaces.html#apis
[4] http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html#apis
[5] http://viewcvs.globus.org/viewcvs.cgi/ws-gram/client/java/source/src/org/globus/exec/client/GramJob.java?rev=1.129.2.3&only_with_tag=globus_4_0_branch&content-type=text/vnd.viewcvs-markup
[6] http://java.sun.com/j2se/1.4.2/docs/api/java/io/File.html
[7] http://java.sun.com/j2se/1.4.2/docs/api/java/io/FileInputStream.html
[8] http://java.sun.com/j2se/1.4.2/docs/api/java/net/URL.html
[9] http://java.sun.com/j2se/1.4.2/docs/api/java/util/LinkedList.html
[10] http://java.sun.com/j2se/1.4.2/docs/api/java/util/List.html
[11] http://java.sun.com/j2se/1.4.2/docs/api/java/util/Vector.html
[12] http://java.sun.com/j2se/1.4.2/docs/api/java/security/cert/X509Certificate.html
[13] http://java.sun.com/j2ee/1.4/docs/api/javax/xml/rpc/Stub.html
[14] http://java.sun.com/j2ee/1.4/docs/api/javax/xml/soap/SOAPElement.html
[15] http://ws.apache.org/axis/java/apiDocs/org/apache/axis/components/uuid/UUIDGenFactory.html
[16] http://ws.apache.org/addressing/apidocs/org/apache/axis/message/addressing/AttributedURI.html
[17] http://ws.apache.org/addressing/apidocs/org/apache/axis/message/addressing/EndpointReferenceType.html
[18] http://www.globus.org/api/javadoc-4.0/globus_wsrf_delegation_service_java/org/globus/delegation/DelegationUtil.html
[19] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/CreateManagedJobInputType.html
[20] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/CreateManagedJobOutputType.html
[21] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/JobDescriptionType.html
[22] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/ManagedJobFactoryPortType.html

```
import org.globus.exec.generated.ManagedJobPortType;[23]
import org.globus.exec.generated.ReleaseInputType;[24]
import org.globus.exec.utils.ManagedJobConstants;[25]
import org.globus.exec.utils.ManagedJobFactoryConstants;[26]
import org.globus.exec.utils.client.ManagedJobClientHelper;[27]
import org.globus.exec.utils.client.ManagedJobFactoryClientHelper;[28]
import org.globus.exec.utils.rsl.RSLHelper;[29]
import org.globus.wsrf.NotificationConsumerManager;[30]
import org.globus.wsrf.WSNConstants;[31]
import org.globus.wsrf.encoding.ObjectDeserializer;[32]
import org.globus.wsrf.impl.security.authentication.Constants;[33]
import org.globus.wsrf.impl.security.authorization.Authorization;[34]
import org.globus.wsrf.impl.security.authorization.HostAuthorization;[35]
import org.globus.wsrf.impl.security.authorization.IdentityAuthorization;[36]
import org.globus.wsrf.impl.security.authorization.SelfAuthorization;[37]
import org.globus.wsrf.impl.security.descriptor.ClientSecurityDescriptor;[38]
import org.globus.wsrf.impl.security.descriptor.GSISecureMsgAuthMethod;[39]
import org.globus.wsrf.impl.security.descriptor.GSITransportAuthMethod;[40]
import org.globus.wsrf.impl.security.descriptor.ResourceSecurityDescriptor;[41]
import org.gridforum.jgss.ExtendedGSSManager;[42]
import org.ietf.jgss.GSSCredential;[43]
import org.oasis.wsn.Subscribe;[44]
import org.oasis.wsn.SubscribeResponse;[45]
import org.oasis.wsn.SubscriptionManager;[46]
import org.oasis.wsn.TopicExpressionType;[47]
import org.oasis.wsn.WSBaseNotificationServiceAddressingLocator;[48]
import org.oasis.wsrf.lifetime.Destroy;[49]
import org.oasis.wsrf.properties.GetMultipleResourceProperties_Element;[50]
```

[23] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/ManagedJobPortType.html
[24] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_common_java/org/globus/exec/generated/ReleaseInputType.html
[25] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_utils_java/org/globus/exec/utils/ManagedJobConstants.html
[26] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_utils_java/org/globus/exec/utils/ManagedJobFactoryConstants.html
[27] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_utils_java/org/globus/exec/utils/client/ManagedJobClientHelper.html
[28] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_utils_java/org/globus/exec/utils/client/ManagedJobFactoryClientHelper.html
[29] http://www.globus.org/api/javadoc-4.0/globus_wsrf_gram_utils_java/org/globus/exec/utils/rsl/RSLHelper.html
[30] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/notification/ClientNotificationConsumerManager.html
[31] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/WSNConstants.html
[32] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/encoding/ObjectDeserializer.html
[33] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/authentication/Constants.html
[34] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/authorization/Authorization.html
[35] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/authorization/HostAuthorization.html
[36] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/authorization/IdentityAuthorization.html
[37] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/authorization/SelfAuthorization.html
[38] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/descriptor/ClientSecurityDescriptor.html
[39] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/descriptor/GSISecureMsgAuthMethod.html
[40] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/descriptor/GSITransportAuthMethod.html
[41] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/globus/wsrf/impl/security/descriptor/ResourceSecurityDescriptor.html
[42] http://www.cogkit.org/release/4_1_2/api/jglobus/org/gridforum/jgss/ExtendedGSSManager.html
[43] http://java.sun.com/j2se/1.4.2/docs/api/org/ietf/jgss/GSSCredential.html
[44] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/Subscribe.html
[45] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/SubscribeResponse.html
[46] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/SubscriptionManager.html
[47] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/TopicExpressionType.html
[48] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsn/WSBaseNotificationServiceAddressingLocator.html
[49] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrf/lifetime/Destroy.html
[50] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrf/properties/GetMultipleResourceProperties_Element.html

```
import org.oasis.wsrf.properties.GetMultipleResourcePropertiesResponse;[51]
import org.oasis.wsrf.properties.GetResourcePropertyResponse;[52]
```

2. Loading the job description

```
File jobDescriptionFile = new File("myjobdesc.xml");
JobDescriptionType jobDescription = RSLHelper.readRSL(jobDescriptionFile);
```

The object `jobDescription` will be of sub-type MultiJobDescriptionType if the file contents is a multi-job description.

3. Creating the factory service stub

```
URL factoryUrl = ManagedJobFactoryClientHelper.getServiceURL(
    contactString).getURL();
String factoryType
    = ManagedJobFactoryConstants.FACTORY_TYPE.<factory type constant>;
EndpointReferenceType factoryEndpoint
    = ManagedJobFactoryClientHelper.getFactoryEndpoint(factoryUrl, factoryType);
ManagedJobFactoryPortType factoryPort
    = ManagedJobFactoryClientHelper.getPort(factoryEndpoint);
```

The format of `contactString` is *[protocol://]host[:port][/servicepath]*.

4. Loading a proxy from a file

- Default proxy file:

```
ExtendedGSSManager manager =
    (ExtendedGSSManager)ExtendedGSSManager.getInstance();

GSSCredential cred = manager.createCredential(
    GSSCredential.INITIATE_AND_ACCEPT);
```

- Specific proxy file:

```
File proxyFile = new File("proxy_file");
byte[] proxyData = new byte[(int)proxyFile.length)];
FileInputStream inputStream = new FileInputStream(proxyFile);
inputStream.read(proxyData);
inputStream.close();

ExtendedGSSManager manager =
    (ExtendedGSSManager)ExtendedGSSManager.getInstance();
```

[51] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrf/properties/GetMultipleResourcePropertiesResponse.html
[52] http://www.globus.org/api/javadoc-4.0/globus_java_ws_core/org/oasis/wsrf/properties/GetResourcePropertyResponse.html

```
  GSSCredential proxy = manager.createCredential(
      proxyData,
      ExtendedGSSCredential.IMPEXP_OPAQUE,
      GSSCredential.DEFAULT_LIFETIME,
      null,
      GSSCredential.ACCEPT_ONLY);
```

5. Setting stub security parameters

```
ClientSecurityDescriptor secDesc = new ClientSecurityDescriptor();
secDesc.setGSITransport(Constants.<protection level constant>);
secDesc.setAuthz(<Authorization sub-class instance>);
if (proxy != null) {
    secDesc.setGSSCredential(proxy);
}
((Stub) port)._setProperty(Constants.CLIENT_DESCRIPTOR, secDesc);
```

Use setGSISecureMsg() for GSI Secure Message.

6. Querying for factory resource properties

6.1. One at a time

```
GetResourcePropertyResponse response
    = factoryport.getResourceProperty(ManagedJobConstants.<RP constant>);

SOAPElement[] any = response.get_any();

... = ObjectDeserializer.toObject(any[0], <RP type>.class);
```

6.2. Many at a time

```
GetMultipleResourceProperties_Element rpRequest
    = new GetMultipleResourceProperties_Element();
rpRequest.setResourceProperty(new QName[] {
    ManagedJobFactoryConstants.<RP constant #1>,
    ManagedJobFactoryConstants.<RP constant #2>,
    ManagedJobFactoryConstants.<RP constant #N>
});
GetMultipleResourcePropertiesResponse response
    = factoryPort.getMultipleResourceProperties(rpRequest);

SOAPElement[] any = response.get_any();

... = ObjectDeserializer.toObject(any[0], <RP #1 type>.class);
... = ObjectDeserializer.toObject(any[0], <RP #2 type>.class);
... = ObjectDeserializer.toObject(any[0], <RP #N type>.class);
```

7. Delegating credentials (if needed)

```
X509Certificate certToSign = DelegationUtil.getCertificateChainRP(
    delegFactoryEndpoint,   //EndpointReferenceType
    secDesc,                //ClientSecurityDescriptor
)[0];   //first element in the returned array
EndpointReferenceType credentialEndpoint = DelegationUtil.delegate(
    delegFactoryurl,        //String
    credential,             //GlobusCredential
    certToSign,             //X509Certificate
    lifetime,               //int (seconds)
    fullDelegation,         //boolean
    secDesc);               //ClientSecurityDescriptor
```

There are three types of delegated credentials:

1. Credential used by the job to generate user-owned proxy:

   ```
   jobDescription.setJobCredential(credentialEndpoint);
   ```

2. Credential used to contact RFT for staging and file clean up:

   ```
   jobDescription.setStagingCredentialEndpoint(credentialEndpoint);
   ```

3. Credential used by RFT to contact GridFTP servers:

   ```
   TransferRequestType stageOut = jobDescription.getFileStageOut();
            stageOut.setTransferCredential(credentialEndpoint);
   ```

   Do the same for fileStageIn and fileCleanUp.

8. Creating the job resource

```
CreateManagedJobInputType jobInput = new CreateManagedJobInputType();
jobInput.setJobID(new AttributeURI("uuid: " +  UUIDGenFactory.getUUIDGen().nextUUID()));
jobInput.setInitialTerminationTime(<Calendar instance>);
if (multiJob) jobInput.setMultiJob(jobDescription) else jobInput.setJob(jobDescription);
if (subscribeOnCreate) jobInput.setSubscribe(subscriptionReq);
CreateManagedJobOutputType createResponse
    = factoryPort.createManagedJob(jobInput);
EndpointReferenceType jobEndpoint = createResponse.getManagedJobEndpoint();
```

9. Creating the job service stub

```
ManagedJobPortType jobPort = ManagedJobClientHelper.getPort(jobEndpoint);
```

You must set the appropriate security parameters for the job service stub (`jobPort`) as well.

10. Subscribing for job state notifications

```
NotificationConsumerManager notifConsumerManager
    = NotificationConsumerManager.getInstance();

notifConsumerManager.startListening();
List topicPath = new LinkedList();
topicPath.add(ManagedJobConstants.RP_STATE);

ResourceSecurityDescriptor resourceSecDesc = new ResourceSecurityDescriptor();
resourceSecDesc.setAuthz(Authorization.<authz type constant>);

Vector authMethods = new Vector();
authMethods.add(GSITransportAuthMethod.BOTH);
resourceSecDesc.setAuthMethods(authMethods);

EndpointReferenceType notificationConsumerEndpoint
    = notifConsumerManager.createNotificationConsumer(
        topicPath,
        this,
        resourceSecDesc);


Subscribe subscriptionReq = new Subscribe();
subscriptionReq.setConsumerReference(
    notificationConsumerEndpoint);

TopicExpressionType topicExpression = new TopicExpressionType(
    WSNConstants.SIMPLE_TOPIC_DIALECT,
    ManagedJobConstants.RP_STATE);
subscriptionReq.setTopicExpression(topicExpression);

EndpointReferenceType subscriptionEndpoint;
```

- Subscribe on creation

  ```
  jobInput.setSubscribe(subscriptionReq);
  ```

- Subscribe after creation

  ```
  SubscribeResponse subscribeResponse
          = jobPort.subscribe(subscriptionRequest);
       subscriptionEndpoint = subscribeResponse.getSubscriptionReference();
  ```

11. Releasing any state holds (if necessary)

```
jobPort.release(new ReleaseInputType());
```

12. Destroying resources

```
/*destroy subscription resource*/
SubscriptionManager subscriptionManagerPort
    = new WSBaseNotificationServiceAddressingLocator()
    .getSubscriptionManagerPort(subscriptionEndpoint);

//set stub security parameters on subscriptionManagerPort

subscriptionManagerPort.destroy(new Destroy());


/*destroy the job resource*/
jobPort.destroy(new Destroy());
```

Chapter 18. Submitting a job in C using WS GRAM

The following is a general scenario for submitting a job using the C stubs and APIs. Please consult the C WS Core API[1], WS-GRAM API[2] documentation for details on the APIs used in this example.

1. Loading the job description

```
const char *                        file = "job.xml";
globus_soap_message_handle_t        message;
wsgram_CreateManagedJobInputType    input;

globus_soap_message_handle_init_from_file(&message, file);

globus_soap_message_deserialize_element_unknown(message, &element);

if(strcmp(element.local, "job") == 0)
{
    wsgram_JobDescriptionType *     jd;

    input.choice_value.type = wsgram_CreateManagedJobInputType_job;
    jd = &input.choice_value.value.job;

    wsgram_JobDescriptionType_deserialize(&element, jd, message, 0);
}
else if(strcmp(element.local, "multiJob") == 0)
{
    wsgram_MultiJobDescriptionType *       mjd;

    input.choice_value.type = wsgram_CreateManagedJobInputType_multiJob;
    mjd = &input.choice_value.value.multiJob;

    wsgram_MultiJobDescriptionType_deserialize(&element, mjd, message, 0);
}
xsd_QName_destroy_contents(&element);
globus_soap_message_handle_destroy(message);
```

This code sets the choice value of the wsgram_CreateManagedJobInputType to be the appropriate type depending on whether the *job description* is a job or *multijob* request.

2. Setting the security attributes

```
globus_soap_message_attr_t              message_attr;

globus_soap_message_attr_init(&message_attr);
```

[1] ../../../common/cwscore/C_WS_Core_Public_Interfaces.html#apis
[2] ../WS_GRAM_Public_Interfaces.html#apis

```
/*
 * Set authentication mode to host authorization: other possibilities are
 * GLOBUS_SOAP_MESSAGE_AUTHZ_HOST_IDENTITY or
 * GLOBUS_SOAP_MESSAGE_AUTHZ_HOST_SELF.
 */
globus_soap_message_attr_set(
    message_attr,
    GLOBUS_SOAP_MESSAGE_AUTHZ_METHOD_KEY,
    NULL,
    NULL,
    (void *) GLOBUS_SOAP_MESSAGE_AUTHZ_HOST);

/*
 * Set message protection level. GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_PRIVACY
 * for encryption.
 */
globus_soap_message_attr_set(
    message_attr,
    GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_KEY,
    NULL,
    NULL,
    (void *) GLOBUS_SOAP_MESSAGE_AUTH_PROTECTION_PRIVACY);
```

3. Creating the factory client handle

```
ManagedJobFactoryService_client_handle_t    factory_handle;

result = ManagedJobFactoryService_client_init(
    &factory_handle,
    message_attr,
    NULL);
```

4. Querying for factory resource properties

4.1. One at a time

```
/*
 * localResourceManager, or other resource property names as defined in the
 * WSDL
 */
xsd_QName                                 property_name =
{
    "http://www.globus.org/namespaces/2004/10/gram/job",
    "localResourceManager"
};
wsrp_GetResourcePropertyResponseType *    property_response;
int                                       fault_type;
xsd_any *                                 fault;
```

```
ManagedJobFactoryPortType_GetResourceProperty(
    factory_handle,
    endpoint,
    &property_name,
    &property_response,
    (ManagedJobFactoryPortType_GetResourceProperty_fault_t *) &fault_type,
    &fault);
```

If this is successful, then `property_response`'s `any` field will contain the deserialized data in the `value` field of the first element in the array.

```
xsd_string * localResourceManager = property_response->any.elements[0].value;

printf("local resource manager is %s\n", *localResourceManager);
```

5. Creating the notification consumer

The notification consumer can be either passed in as part of the `wsgram_CreateManagedJobInputType` or through a separate invocation of `ManagedJobPortType_Subscribe_epr()`.

```
    globus_service_engine_t             engine;
    wsa_EndpointReferenceType           consumer_reference;

    globus_service_engine_init(&engine, NULL, NULL, NULL, NULL, NULL);

    globus_notification_create_consumer(
            &consumer_reference,
            engine,
            notify_callback,
            NULL);
```

6. Creating the job resource

First, prepare the other parts of the `wsgram_CreateManagedJobInputType` structure.

```
/*
 * You can set input.InitialTerminationTime to be a timeout if interested.
 * The xsd_dateTime type is a struct tm pointer.
 */
time_t                                  term_time = time(NULL);
globus_uuid_t                           uuid;
wsa_AttributedURI *                     job_id;
wsa_EndpointReferenceType *             factory_epr;
xsd_any *                               reference_property;
wsgram_CreateManagedJobOutputType *     output = NULL;
xsd_QName                               factory_reference_id_qname =
{
    "http://www.globus.org/namespaces/2004/10/gram/job",
    "ResourceID"
```

```
};

term_time += 60 * 60; /* 1 hour later */
xsd_dateTime_copy(&input.InitialTerminationTime, gmtime(&term_time));

/*
 * Set unique JobID. This is used to reliably create jobs and check for status.
 */
globus_uuid_create(&uuid);
wsa_AttributedURI_init(&job_id);
job_id->base_value = globus_common_create_string("uuid:%s", uuid.text);

/* To subscribe to notifications at create time, add the consumer's EPR to
 * the input message. Otherwise, use the EPR created above in a
 * call to
 */
wsnt_SubscribeType_init(&input.Subscribe);
wsa_EndpointReferenceType_copy_contents(
        &input.Subscribe.ConsumerReference,
        &consumer_reference);

xsd_any_init(&input.Subscribe->TopicExpression.any);
&input.Subscribe->TopicExpression.any->any_info =
        &xsd_QName_contents_info;
xsd_QName_copy(
    (xsd_QName **) &input.Subscribe->TopicExpression.any->any.value,
    &ManagedJobPortType_state_rp_qname);

xsd_anyURI_copy_cstr(
    &input.Subscribe->TopicExpression._Dialect,
    "http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple");
xsd_boolean_init(&input.Subscribe->UseNotify);

*(&input.Subscribe->UseNotify) = GLOBUS_TRUE;

/* Construct the EPR of the job factory */
wsa_EndpointReferenceType_init(&factory_epr);
wsa_AttributedURI_init_contents(&factory_epr->Address);
xsd_anyURI_init_contents_cstr(&factory_epr->Address.base_value,
        globus_common_create_string(
                "https://%s:%hu/wsrf/services/%s",
                factory_host,
                factory_port,
                MANAGEDJOBFACTORYSERVICE_BASE_PATH);
wsa_ReferencePropertiesTypeinit(&factory_epr->ReferenceProperties);
reference_property = xsd_any_array_push(
        &factory_epr->ReferenceProperties.any);
reference_property->any_info = &xsd_string_info;
xsd_QName_copy(
        &reference_property->element,
        &factory_reference_id_qname);

xsd_string_copy_cstr(
        (xsd_string **) &reference_property->value,
```

```
        "Fork");

/* Submit the request to the service container */
ManagedJobFactoryPortType_createManagedJob_epr(
        factory_handle,
        factory_epr,
        input,
        &output,
        (ManagedJobFactoryPortType_createManagedJob_fault_t *) &fault_type,
        &fault);
```

If this is successful, then the `output` structure will be initialized with the results of the operation. Of particular interest is the `managedJobEndpoint` which contains the reference to the newly-created job resource.

7. Subscribing for job state notifications

In order to subscribe for job state change notifications to an existing job resource, initialize the `subscribe_input` used below in the same way as `input.Subscribe` was initialized above.

```
    ManagedJobService_client_handle_t   job_handle;
    wsnt_SubscribeType                  subscribe_input;
    wsnt_SubscribeResponseType *        subscribe_response;

ManagedJobService_client_init(
    &job_handle,
    message_attr,
    NULL);

ManagedJobPortType_Subscribe_epr(
    job_handle,
    output->managedJobEndpoint,
    subscribe_input,
    &subscribe_response,
    (ManagedJobPortType_Subscribe_fault_t *) &fault_type,
    &fault);
```

8. Releasing any state holds (if necessary)

```
    wsgram_ReleaseInputType             release;
    wsgram_ReleaseOutputType *          release_response = NULL;

    wsgram_ReleaseInputType_init_contents(&release);

ManagedJobPortType_release_epr(
    job_handle,
    output->managedJobEndpoint,
    &release,
    &release_response,
    (ManagedJobPortType_release_fault_t *) &fault_type,
    &fault);
```

9. Destroying resources

```
/* destroy subscription resource */
    SubscriptionManagerService_client_init    subscription_handle;
    wsnt_DestroyType                          destroy;
    wsnt_DestroyResponseType *                destroy_response = NULL;

    SubscriptionManagerService_client_init(
        &subscription_handle,
        message_attr,
        NULL);
    /* if subscription done at job creation time, use
     * output->subscriptionEndpoint in place of
     * subscribe_response->SubscriptionReference,
     */
    SubscriptionManager_Destroy_epr(
        subscription_handle,
        subscribe_response->SubscriptionReference,
        &destroy,
        &destroy_response,
        (SubscriptionManager_Destroy_fault_t *) &fault_type,
        &fault);

    /* destroy the job resource */
    jobPort.destroy(new Destroy());
    ManagedJobPortType_Destroy_epr(
        job_handle,
        output->managedJobEndpoint,
        &destroy,
        &destroy_response,
        (ManagedJobPortType_Destroy_fault_t *) &fault_type,
        &fault);
```

10. Building a client

In order to build a client application, certain flags must be passed to the compiler and linker to enable them to be able to locate headers and libraries. The easiest way to do so is to generate a *makefile header*, which is a fragment of a Makefile which includes all of the necessary flags needed to build the application. To do this, issue the command:

```
% globus-makefile-header --flavor=gcc32dbg globus_c_gram_client_bindings > Makefile.inc
```

Then, write your makefile to include this file and use the GLOBUS_CC, GLOBUS_LD, GLOBUS_CFLAGS, GLOBUS_LDFLAGS, and GLOBUS_PKG_LIBS macros. For example:

```
GLOBUS_FLAVOR_NAME=gcc32dbg

include Makefile.inc

CC = $(GLOBUS_CC)
```

```
LD = $(GLOBUS_LD)

CFLAGS = $(GLOBUS_CFLAGS)
LDFLAGS = $(GLOBUS_LDFLAGS) $(GLOBUS_PKG_LIBS)

client: client.c
```

GT 4.0 Execution Management Glossary

B

batch scheduler — See scheduler.

C

Condor — A job scheduler mechanism supported by GRAM. See http://www.cs.wisc.edu/condor/.

G

globusrun-ws — A command line program used to submit jobs to a WS GRAM service. See the commandline reference page at: http://www.globus.org/toolkit/docs/4.0/execution/wsgram/rn01re01.html

Grid Resource allocation and Management (GRAM) — Web Services Grid Resource Allocation and Management (WS GRAM) component comprises a set of WSRF-compliant Web services to locate, submit, monitor, and cancel jobs on Grid computing resources. WS GRAM is not a *job scheduler*, but rather a set of services and clients for communicating with a range of different batch/cluster job schedulers using a common protocol. WS GRAM is meant to address a range of jobs where reliable operation, stateful monitoring, credential management, and file staging are important.

J

job description — Term used to describe a WS GRAM job for GT4.

job scheduler — See scheduler.

L

LSF — A job scheduler mechanism supported by GRAM.

http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/

M

meta-scheduler — A program that typically operates at a higher level than a job scheduler (typically, above the GRAM level). It schedules and submits jobs to GRAM services.

Managed Executable Job Service (MEJS) — [FIXME]

Managed Job Factory Service (MJFS) — [FIXME]

Managed Multi Job Service (MMJS)	[FIXME]
MMJS subjob	One of the executable jobs in a multijob rendezvous.
multijob	A job that is itself composed of several executable jobs; these are processed by the MMJS subjob. See Also MMJS subjob.
multijob rendezvous	A mechanism used by GRAM to synchronize between job processes in a multiprocess job and between.

P

Portable Batch System (PBS)	A job scheduler mechanism supported by GRAM. http://www.openpbs.org

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 adapter	The interface used by GRAM to communicate/interact with a job scheduler mechanism. In GT4 this is both the perl submission scripts and the SEG program. See Also scheduler.
Scheduler Event Generator (SEG)	[FIXME]
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. http://www.courtesan.com/sudo/

U

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