
Audit Logging in GRAM2

Table of Contents

1. Overview	1
2. Audit and Accounting Records	2
3. GJID	2
4. Accessing Audit and Accounting Information	2
5. For More Information	3
6. Functionality	3
7. Configuration to enable audit logging	3

1. Overview

GRAM2 includes mechanisms to provide access to audit and accounting information associated with jobs that GRAM2 submits to a local resource manager (LRM) such as PBS, LSF, or Condor.



Note

Remember, 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 general information about the usage of the underlying LRM, such as:

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

The following three use cases give a better overview of the meaning and purpose of auditing and accounting:

1. **Group Access.** A grid resource provider allows a remote service (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 stores 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 a job is stored at the end of the processing cycle of a job - either when it is completely processed or failed.

2. Audit and Accounting Records

While audit and accounting records may be generated and stored by different entities in different contexts, we make the following assumptions in this chapter:

	Audit Records	Accounting Records
Generated by:	GRAM service	LRM to which the GRAM service submits jobs
Stored in:	Database, indexed by GJID	LRM, indexed by JID
Data that is stored:	See list below.	May include all information about the duration and resource-usage of a job

The audit record of each job contains the following data:

- **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

3. GJID

The GRAM2 service returns a "job contact" that is used to control the job. The job contact, by default, is already in an acceptable GJID format; therefore, the GRAM2 client and service do not need to convert it in any way.

4. Accessing Audit and Accounting Information

To connect the two sets of records, both audit and accounting, we require that GRAM records the JID in each audit record that it generates. It is then straightforward for an audit service to respond to requests such as "Give me the charge of the job with JID x" by:

1. first selecting matching record(s) from the audit table,
2. then using the local JID(s) to join to the accounting table of the LRM and access relevant accounting record(s).

We propose a Web Service interface for accessing audit and accounting information. [OGSA-DAI](#)¹ is a WSRF service that can create a single virtual database from two or more remote databases. In the future, other per-job information such as job performance data could be stored using the GJID or local JID as an index, and then made available in the same virtual database.

5. For More Information

The rest of this chapter focuses on how to configure GRAM2 to enable Audit Logging. A case study for TeraGrid can be read [here](#)², which also includes more information about how to use this data to get accounting information of a job, query the audit database for information via a Web Services interface, etc.

6. Functionality

Audit logging in GRAM2 is realized in the following way:

1. The job manager writes a file to disk for each job. This file contains the audit record. The format of an audit record file that is logged to the database is a comma-separated list of double-quoted strings.
2. The audit records are not inserted into the GRAM audit database directly. The job manager will, at final job termination (FAILED or DONE state), write a record to a unique file in a directory specified by a configuration file. These audit files must be uploaded by a program which can be called manually or be run periodically as a cron job. The program is a perl script and is located in `${GLOBUS_LOCATION}/libexec/globus-gram-audit` and creates audit records in the configured database from the user audit files. Once the record is uploaded, the program will remove the audit file.

Here's an example on how a crontab entry must look like in order to run the script every 15 minutes:

```
0,15,30,45 * * * * ${GLOBUS_LOCATION}/libexec/globus-gram-audit
```

The script gets the necessary parameters to connect to the database from the configuration file `${GLOBUS_LOCATION}/etc/globus-job-manager-audit.conf`, which is described below.

You may notice that this method is different than that used for GRAM4. GRAM4 writes audit records directly to the audit database. This is done because only a single account (the container account) may be given access to the DB. The container account is already trusted, so this is reasonable for GRAM4.

In GRAM2, the Job Manager process writes the audit records. However, this process runs under the user's account; opening up access to the audit database for each user is *not* acceptable.

Therefore, the Job Manager writes the audit record to a file. Then a single database upload program, running under the same GRAM4 container account (e.g. globus), can first verify that the owner of the GRAM2 audit file and the username field in the audit record match. This prevents users from interfering with other users' audit records.

7. Configuration to enable audit logging

Audit logging is turned off by default. To turn on Audit Logging, follow these steps:

¹ <http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/>

² http://www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit

7.1. Configure Audit Record Directory

Add the following line to `$GLOBUS_LOCATION/etc/globus-job-manager.conf`:

```
-audit-directory <your desired audit record directory>
```

7.2. Create Audit Record Directory

Create the audit record directory specified in the above configuration file with the following permissions:

```
rws-wsrwx
```

7.3. Database configuration

Edit `$GLOBUS_LOCATION/etc/globus-job-manager-audit.conf` to include the correct database connection parameters. For example::

```
DRIVERCLASS:com.mysql.jdbc.Driver
USERNAME:john
PASSWORD:foo
URL:jdbc:mysql://myhost/auditDatabase
AUDITVERSION:1
```

We support 3 database systems: MySQL, PostgreSQL, Derby. The following table gives an overview which values must be used for the parameters `url` and `driverClassName` in the above JNDI configuration for the various db systems. Derby is configured as the default DB system.

DB system	driverClassName	url
MySQL	com.mysql.jdbc.Driver	<code>jdbc:mysql://HOST[:PORT]/auditDatabase</code>
PostgreSQL	org.postgresql.Driver	<code>jdbc:mysql://HOST[:PORT]/auditDatabase</code>
Derby	org.apache.derby.jdbc.Embedded-Driver	<code>jdbc:derby:directory:PATH_TO_GLOBUS_LOCATION/var/gram/auditDatabase</code>

Don't change the parameter `AUDITVERSION` for now. In future releases we'll support more than one version.

7.4. Creating the Audit Database

Audit records are stored in a database which must be set up once.

7.4.1. MySQL

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

1. Create a database inside of MySQL
2. Grant necessary privileges to the account that will be used to upload the audit records in the audit. Typically the "globus" account.
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/globus_wsrf_gra
```

Enter password:

```
host:~ feller$
```

7.4.2. PostgreSQL

The following describes how to set up the audit database in PostgreSQL:

1. Create a database inside of PostgreSQL
2. Grant necessary privileges to the account that will be used to upload the audit records in the audit. Typically the "globus" account.
3. Use the schema to create the table:

```
# Connect as postgres admin
create database gt4audit\g
create user gt4auditload with encrypted password '<password1>'\g
create user gt4auditview with encrypted password '<password2>'\g
\c gt4audit
\i gram_audit_schema_postgres-8.0.sql
grant insert on gram_audit_table to gt4auditload\g
grant select on gram_audit_table to gt4auditview\g
\q
```

You must also update `pg_hba.conf` to allow connections from container host (`pg_hba.conf` configures client authentication and is stored in the database cluster's data directory):

```
hostssl  gt4audit  gt4auditload  <containerhostip> 255.255.255.255 md5
host     gt4audit  gt4auditload  <containerhostip> 255.255.255.255 md5
hostssl  gt4audit  gt4auditview  <containerhostip> 255.255.255.255 md5
host     gt4audit  gt4auditview  <containerhostip> 255.255.255.255 md5
```

7.4.3. Derby

During GT installation the Derby audit database is already created. Its location is `${GLOBUS_LOCATION}/var/gram/auditDatabase`. If you ever have to create it manually, make sure that this directory does not exist and then call `${GLOBUS_LOCATION}/setup/globus/setup-gram-service-database`. The user and password information can be found in `${GLOBUS_LOCATION}/share/globus_wsrf_gram/gram_audit_v1_schema_derby.sql`.