

GT 4.2.1 RLS: System Administrator's Guide

GT 4.2.1 RLS: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with RLS. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the [Installing GT 4.2.1](#). Read through this guide before continuing!

Table of Contents

1. Building and Installing	1
1. Prerequisites	1
2. Installation of RLS	5
3. Database Setup	6
2. Configuring RLS	11
1. Configuration overview	11
2. Server configuration file (globus-rls-server.conf)	11
3. Basic configuration	11
4. Host key and certificate configuration	12
5. Configuring LRC to RLI updates	13
6. Configuring the RLS Server for the WS MDS Index Service	14
7. Configuring the RLS Server for the MDS2 GRIS	15
8. Complete RLS Server settings (globus-rls-server.conf)	15
3. Starting the RLS Server	21
4. Stopping the RLS Server	22
5. Testing	23
1. Manual Tests	23
2. Automated Tests	23
6. Security Considerations	24
1. Replica Location Service (RLS) Security Considerations	24
7. Troubleshooting	25
1. Errors	25
2. I need help installing MySQL DBMS	26
3. How can I check if MySQL is installed correctly?	26
4. I need help installing PostgreSQL DBMS	27
5. How can I check if PostgreSQL is installed correctly?	27
6. I cannot connect to PostgreSQL	27
7. Performance of RLS with PostgreSQL has degraded over time	28
8. I need help installing SQLite embedded database	28
9. How can I check if SQLite is installed correctly?	28
10. I need help installing iODBC	28
11. I need help installing unixODBC	29
12. Building unixODBC failed on OS X	29
13. I need help installing MySQL Connector/ODBC	30
14. I need help installing psqlODBC	32
15. I need help installing SQLite ODBC Driver	33
16. Which libraries do you recommend to new users?	34
17. I need help configuring my DSNs	34
18. How can I tell which odbc.ini is being used?	35
19. Testing your ODBC configuration	35
20. Important notes on RLS initialization	37
21. Linux kernel and glibc incompatibility	37
8. Usage statistics collection by the Globus Alliance	39
1. RLS-specific usage statistics	39
Glossary	41
Index	42

List of Tables

1.1. Database & ODBC Observations	4
2.1. Complete RLS Server settings (globus-rls-server.conf)	16
7.1. Replica Locator Service (RLS) Errors	25

Chapter 1. Building and Installing

RLS is built and installed as part of the Globus Toolkit installation. This section provides instructions specific to building and installing RLS. Most of this section concerns the optional setup and configuration of a database management system (DBMS) and database connectivity (ODBC) software for the RLS server. By default, the RLS is installed and automatically configured to use an embedded database. No additional steps are required unless you wish to configure RLS to use your own ODBC-compliant database management system.

1. Prerequisites

By default, the RLS is installed and automatically configured to use an embedded database. If you wish to use the default configuration, you may skip this section and proceed to [Section 2, “Installation of RLS”](#). If you wish to use your own ODBC-compliant database management system, please review this section carefully.

The RLS depends on DBMS and ODBC software. If you are not familiar with DBMS and ODBC software, you may need to familiarize yourself with these tools before proceeding with RLS installation. Most of the difficulty users experience when installing RLS is actually due to DBMS and ODBC issues, which are often beyond our control.

The following list summarizes the prerequisite database software:

- [Section 1.1, “Database Management System”](#): for persistent storage of replica location information
- [Section 1.2, “ODBC Manager”](#): for vendor-neutral connectivity to DBMS
- [Section 1.3, “ODBC Driver”](#): for vendor-specific connectivity to DBMS from ODBC Managers

When using this guide, please consult the vendor and official project sites for authoritative information on DBMS and ODBC software. Any additional information provided on this site is intended as helpful advice and only reflects our limited experiences as fellow users of the software.

1.1. Database Management System

The RLS depends on a DBMS for persistent storage of replica location information. The RLS is designed to work with MySQL, PostgreSQL, SQLite, and Oracle DBMS software. In the case of MySQL, it is important to use the transaction-capable InnoDB database engine. For complete information on installation and configuration of third-party DBMS software, please refer to:

- [MySQL](#)¹
- [PostgreSQL](#)²
- [SQLite](#)³
- [Oracle](#)⁴

We provide the following tips from the troubleshooting section of this guide:

- [Section 4, “I need help installing PostgreSQL DBMS”](#)

¹ <http://www.mysql.com>

² <http://www.postgresql.org>

³ <http://www.sqlite.org>

⁴ <http://www.oracle.com>

- [Section 5, “How can I check if PostgreSQL is installed correctly?”](#)
- [Section 6, “I cannot connect to PostgreSQL”](#)
- [Section 7, “Performance of RLS with PostgreSQL has degraded over time”](#)
- [Section 2, “I need help installing MySQL DBMS”](#)
- [Section 3, “How can I check if MySQL is installed correctly?”](#)
- [Section 8, “I need help installing SQLite embedded database”](#)
- [Section 9, “How can I check if SQLite is installed correctly?”](#)

1.2. ODBC Manager

The RLS depends on an ODBC manager for standardised database connectivity. The unixODBC or iODBC libraries may be used for this purpose. Please confirm that one of these libraries is installed and configured before proceeding with RLS installation. If you need more information on the ODBC managers, consult their respective project home pages.

- [unixODBC](#)⁵
- [iODBC](#)⁶

You may also find that unixODBC or (less likely) iODBC are included along with a vendor's odbc driver. This appears to be the case with MySQL Connector/ODBC drivers at the time of writing this document.

We provide the following tips from the troubleshooting section of this guide:

- [Section 11, “I need help installing unixODBC”](#)
- [Section 10, “I need help installing iODBC”](#)



Note

If you are installing RLS from a Globus Toolkit binary bundle, you do not need to install the unixODBC or iODBC software. The binary installation of RLS is linked to iODBC version 3.51.2 included with the GT bundle.

1.3. ODBC Driver

The RLS depends on an ODBC driver for vendor-specific database connectivity via the vendor neutral ODBC manager. For each vendor you will find a related ODBC driver. You may find relevant information at the following sites:

- [psqlODBC](#)⁷ the official PostgreSQL ODBC Driver
- [MySQL Connector/ODBC](#)⁸ offered directly by MySQL (note: use the 3.51 branch)
- [SQLite ODBC Driver](#)⁹ a freely available ODBC driver for SQLite

⁵ <http://www.unixodbc.org>

⁶ <http://www.iodbc.org>

⁷ <http://pgfoundry.org/projects/psqlodbc/>

⁸ <http://www.mysql.com/products/connector/odbc/index.html>

⁹ <http://www.ch-werner.de/sqliteodbc/>

- Consult Oracle¹⁰ for details on ODBC Driver for Oracle DBMS

We provide the following tips from the troubleshooting section of this guide:

- Section 14, “I need help installing `psqlODBC`”
- Section 13, “I need help installing MySQL Connector/ODBC”
- Section 15, “I need help installing SQLite ODBC Driver”

1.4. Choosing the right DBMS & ODBC software

We have used or know of users who have used the following combinations of software packages on the following platforms. What we provide in this section is a small subset of available options when choosing DBMS & ODBC software.

¹⁰ <http://www.oracle.com>

Table 1.1. Database & ODBC Observations

Architecture	OS	Distribution	ODBC Manager	ODBC Driver	DBMS
PPC	OS X	Apple	unixODBC-2.2.11*	TBD	TBD
SPARC, 32-bit	Solaris	Sun	libiodbc-3.51.2	MyODBC-3.51.06	mysql-standard-4.0.18
SPARC, 32-bit	Solaris	Sun	libiodbc-3.51.2	MyODBC-3.51.12	mysql-4.0.16
SPARC, 32-bit	Solaris	Sun	unixODBC-2.2.11	MyODBC-3.51.12	mysql-4.0.16
SPARC, 32-bit	Solaris	Sun	unixODBC-2.2.11	TBD	TBD
x86	Linux	RedHat	libiodbc-3.51.1	MyODBC-3.51.06	mysql-standard-4.0.18
x86	Linux	RedHat	libiodbc-3.51.2	MyODBC-3.51.06	mysql-standard-4.0.18
x86	Linux	RedHat	libiodbc-3.52.4	MyODBC-3.51.06	mysql-standard-4.0.18
x86	Linux	RedHat	libiodbc-3.52.4	psqlodbc-7.2.5	postgresql-7.4.6
x86	Linux	RedHat	libiodbc-3.51.2	psqlodbc-08.00.0102	postgresql-7.4.6
x86	Linux	RedHat	libiodbc-3.51.2	sqliteodbc-0.69	sqlite-3.3.6
x86	Linux	RedHat	unixODBC-2.2.11	MyODBC-3.51.06	mysql-standard-4.0.18
x86	Linux	RedHat	unixODBC-2.2.11	mysql-connector-odbc-3.51.12	mysql-standard-4.0.18
x86	Linux	RedHat	unixODBC-2.2.11	mysql-connector-odbc-3.51.12	mysql-standard-5.0.21
x86	Linux	RedHat	unixODBC-2.2.11	psqlodbc-7.2.5	postgresql-7.4.6
x86	Linux	RedHat	unixODBC-2.2.11	psqlodbc-08.00.0102	postgresql-7.4.6
x86	OS X	Apple	TBD	TBD	TBD
x86_64	Linux	SuSE	unixODBC-2.2.8-35	MyODBC-unixODBC-3.51.06-121	mysql-client-4.0.18-25
x86_64	Linux	SuSE	unixODBC-2.2.8-35	psqlodbc-08.00.0102	postgresql-7.4.9

(*) See troubleshooting tip [Section 12, “Building unixODBC failed on OS X”](#)

If you build from source, be aware that some versions of a vendor's drivers do not build against some versions of the same vendor's clients. For instance, MySQL Connector/ODBC 3.51.06 does not appear to build against MySQL 4.1.x client libraries, instead it requires MySQL 4.0.x client libraries. Also, we have noticed that psqlodbc-07.03.x will build against pgsq-7.4.x but when using the driver it fails with little or no information to indicate why it failed. We have found that psqlodbc-08.00.x works well with postgres-7.4.6. In all cases, consult the vendor's documentation.

Support RLS development by sending your working RLS + ODBC Manager + ODBC Driver + RDBMS configuration to the rls-user@globus.org list! We cannot test on all platforms with all combinations so we rely on our user community to help advise new users. Your support in this matter is appreciated.

We provide the following tips from the troubleshooting section of this guide:

- [Section 16, “Which libraries do you recommend to new users?”](#)

2. Installation of RLS

The following section describes various options available to install RLS given that you have satisfied the prerequisites. The options include:

- [Section 2.1, “Installing RLS using Globus Toolkit Binary Bundle”](#)
- [Section 2.2, “Installing RLS using Globus Toolkit Source Bundle”](#)
- [Section 2.3, “Updating RLS using Globus Toolkit Source Bundle”](#)



Note

If you set the ODBCINI environment variable to the path of the `odbc.ini` file prior to installation, the `globus-rls-server.conf` will be seeded with the corresponding value. If you are not already familiar with `odbc.ini` it will be explained in [Section 3.5, “Specify Data Source Names \(DSNs\)”](#). The setting is optional, since the `globus-rls-server.conf` can be changed at any time.

2.1. Installing RLS using Globus Toolkit Binary Bundle

Follow instructions provided by [Installing GT 4.2.1](#) and [Installing GT](#) in order to install the RLS from one of the available binary bundles of the Globus Toolkit. As noted in [Section 1.2, “ODBC Manager”](#), the `globus-rls-server` found in the binary bundles is linked to the default ODBC Manager.

Once you have unpackaged the binary bundle, the following commands may be used to install RLS (the `--enable-rls` option is not required):

```
% ./configure --prefix=$GLOBUS_LOCATION --enable-rls
% make
% make install
```

2.2. Installing RLS using Globus Toolkit Source Bundle

Follow instructions provided by [Installing GT 4.2.1](#) and [Installing GT](#) in order to install the RLS from the available source bundle of the Globus Toolkit. In addition to the general requirements, you may optionally set environment variables pertaining to the include and lib path for your ODBC Manager installation. The following environment variables are not required to install RLS.

If you wish to use `unixODBC`, set the following environment variables.

```
% setenv GLOBUS_UNIXODBC_INCLUDES /path/to/unixODBC/include
% setenv GLOBUS_UNIXODBC_LIBS /path/to/unixODBC/lib
```

If you wish to use `iODBC`, set the following environment variables.

```
% setenv GLOBUS_IODBC_INCLUDES /path/to/libiodbc/include
```

```
% setenv GLOBUS_IODBC_LIBS /path/to/libiodbc/lib
```



Note

Your shell format for setting environment variables may be `export VAR=VAL` instead of `setenv VAR VAL`.

Alternatively, instead of setting the above environment variables you may choose to specify the includes and libs path for your ODBC Manager using the Globus Toolkit `configure` script's options, which include `--with-unixodbc-includes=DIR` and `--with-unixodbc-libs=DIR` or `--with-iodbc-includes=DIR` and `--with-iodbc-libs=DIR`.

Once you have unpackaged the source bundle, the following commands may be used to install RLS (the `--enable-rls` option is not required):

```
% ./configure --prefix=$GLOBUS_LOCATION --enable-rls
% make
% make install
```

2.3. Updating RLS using Globus Toolkit Source Bundle

In some situations, you may wish to update an RLS installation using RLS source from the Globus Toolkit Source Bundle. For instance, the RLS found in the Globus Toolkit binary bundles is linked against an ODBC Manager. If you would like to link to a different ODBC Manager, it may be easier to install the toolkit from the binary bundle and just update the RLS server package from source. This may save considerable time compared to building the entire Globus Toolkit from source.

To do this, you must first follow the complete instructions found in [Section 2.1, “Installing RLS using Globus Toolkit Binary Bundle”](#) then set the environment according to the instructions found in [Section 2.2, “Installing RLS using Globus Toolkit Source Bundle”](#). If you have not already set the `GPT_LOCATION` then set it to the `GLOBUS_LOCATION` (i.e., `setenv GPT_LOCATION $GLOBUS_LOCATION`).

Once you have unpackaged the source bundle, the following commands may be used to install RLS:

```
% cd source-trees-thr/replica/rls/server/
% $GPT_LOCATION/sbin/gpt-build -verbose -force gcc32dbgpthr
...
% $GPT_LOCATION/sbin/gpt-postinstall
...
```

You may select a flavor other than `gcc32dbgpthr` that is suitable to your platform and existing installation environment. You may omit the `-verbose` flag.

3. Database Setup

By default, the RLS embedded database and related settings are automatically created and configured. Unless you chose to setup your own database management system for use with the RLS, you may skip this section and proceed to [Chapter 2. Configuring RLS](#).

The following section describes the procedures necessary to complete the database setup for your RLS installation. Topics addressed include:

- [Section 3.1, “Choose databases for RLS Server”](#)
- [Section 3.2, “Configure database related RLS settings”](#)
- [Section 3.3, “Create a database user account”](#)
- [Section 3.4, “Create databases”](#)
- [Section 3.5, “Specify Data Source Names \(DSNs\)”](#)



Note

The rest of this section assumes that the DBMS has been started.

3.1. Choose databases for RLS Server

Decide which database(s) the RLS server will use:

- If the RLS server is a Local Replica Catalog (LRC) server you, will need to create the LRC database.
- If the server is a Replica Location Index (RLI) server, you may need to create a RLI database.

An RLI server can receive updates from LRC servers in one of two forms, as LFN lists (in which case the RLI database must be created) or as highly compressed Bloom filters. Since Bloom filters are relatively small, they are kept in memory and no database is required (though if you plan to use the hierarchical RLI updates feature, you will still need the RLI database). An RLS server can be configured as both an LRC and RLI server.

3.2. Configure database related RLS settings

The RLS Server configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) contains a few settings related to your database setup. Throughout this guide we refer to the username as *dbuser* and the password as *dbpassword* and to the LRC database as *lrc1000* and the RLI database as *rli1000*. Substitute your chosen username, password, and database names where applicable. Also, you may change the `odbcini` setting to reflect the location of your ODBC Manager's `odbc.ini` file. You should be familiar with the location of this file from the installation of your ODBC Manager. As of the GT 4.1.x release, the RLS setup includes a default `odbc.ini` file installed to `$GLOBUS_LOCATION/var/odbc.ini` which is the initial default value of the `odbcini` configuration parameter. We discuss more about `odbc.ini` in [Section 3.5, “Specify Data Source Names \(DSNs\)”](#).

Edit the following fields in `$GLOBUS_LOCATION/etc/globus-rls-server.conf`:

```
# Database connection options
db_user          dbuser
db_pwd          dbpasswordgoeshere
odbcini         /path/to/var/odbc.ini
...
# LRC options
lrc_server      true
lrc_dbname      lrc1000 # optional (if LRC server)
...
#RLI options
```

```
rli_server          true
rli_dbname          rli1000 # optional (if RLI server)
...
rli_bloomfilter     false
...
```

3.3. Create a database user account

You or your system administrator must create a DBMS user account to be used by the RLS Server to log on to the DBMS. SQLite users may safely skip this step.

For MySQL users, the command may look something like this:

```
% /path/to/mysql/bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 71 to server version: 4.0.18-standard-log

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

mysql> use mysql;
Database changed
mysql> grant all on lrc1000.* to dbuser@localhost identified by 'dbpassword';
mysql> grant all on rli1000.* to dbuser@localhost identified by 'dbpassword';
```

For PostgreSQL user, the command may look something like this:

```
% /path/to/pgsql/bin/createuser -P dbuser -W
Enter password for new user:
Enter it again:
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
Password:
CREATE USER
```

3.4. Create databases

We provide SQL Data Definition (DDL) scripts to create the databases required by the RLS Server. Configure the script(s) for the database(s) you will create. The SQL scripts for the LRC and RLI databases are installed in \$GLOBUS_LOCATION/setup/globus.

For PostgreSQL, use:

- globus-rls-lrc-postgres.sql
- globus-rls-rli-postgres.sql

For MySQL, use:

- `globus-rls-lrc-mysql.sql`
- `globus-rls-rli-mysql.sql`

For SQLite, use:

- `globus-rls-lrc-sqlite.sql`
- `globus-rls-rli-sqlite.sql`

For Oracle DBMS, use:

- `globus-rls-lrc-oracle.sql`
- `globus-rls-rli-oracle.sql`

Edit these files to set the name of the database user you created for RLS and the names of the databases configured in `$GLOBUS_LOCATION/etc/globus-rls-server.conf`.

Create the database(s) using the appropriate tool for your chosen DBMS.

For PostgreSQL, the commands should look like this:

```
% createdb -O dbuser -U dbuser -W lrc1000
% createdb -O dbuser -U dbuser -W rli1000
% psql -W -U dbuser -d lrc1000 -f $GLOBUS_LOCATION/setup/globus/globus-rls-lrc-postgres.
% psql -W -U dbuser -d rli1000 -f $GLOBUS_LOCATION/setup/globus/globus-rls-rli-postgres.
```

For MySQL, the commands should look like this:

```
% mysql -p -u dbuser < $GLOBUS_LOCATION/setup/globus/globus-rls-lrc-mysql.sql
% mysql -p -u dbuser < $GLOBUS_LOCATION/setup/globus/globus-rls-rli-mysql.sql
```

For SQLite, the commands should look like this:

```
% sqlite3 $GLOBUS_LOCATION/var/globus-rls-lrc1000.db < $GLOBUS_LOCATION/setup/globus/glob
% sqlite3 $GLOBUS_LOCATION/var/globus-rls-rli1000.db < $GLOBUS_LOCATION/setup/globus/glob
```

3.5. Specify Data Source Names (DSNs)

Data Source Names (DSNs) are used by ODBC software (and the applications that depend on ODBC software) to define connection settings between the database client and the DBMS. The RLS depends on the data source names configured in [Section 3.2, “Configure database related RLS settings”](#) for the `lrc_dbname` and `rli_dbname` settings.

DSNs are configured in a file named `odbc.ini`. Most ODBC Managers find this file in the following search order.

- The `ODBCINI` environment variable with value of `/path/to/odbc.ini`
- The user `odbc.ini` found at `~/odbc.ini`

- The system `odbc.ini` found at `/etc/odbc.ini` or similar location

DSNs may be configured in a variety of ways, but they essentially contain the same information which includes:

- *DSN* such as `lrc1000` or `rli1000`
- *Driver*: the path to the ODBC driver (such as `psqlodbc.so`, `libmyodbc3.so`, or `libsqlite3odbc.so`)
- *Database*: the name of the database (in the case of SQLite, this parameter's value will indicate the path to the database file)
- *Server name*: the hostname of the DBMS server
- *Port*: the port number of the DBMS server
- *Socket*: in some cases a local socket file is used if the DBMS and RLS are installed on the same server

As of the Globus Toolkit 4.1.x release, the RLS setup includes a default `odbc.ini` file installed to `$GLOBUS_LOCATION/var/odbc.ini`. The settings of the included `odbc.ini` are appropriate for usage with the `psqlODBC` driver and reference `psqlODBC` libraries installed to `$GLOBUS_LOCATION/lib/psqlodbc.so`, which is the location of the libraries as installed by the Globus Toolkit binary installer and the source installer by using the `make globus_database_psqlodbc` target. When using the installed `odbc.ini` file, you may still need to set your `ODBCINI` environment variable to the `odbc.ini` file's location.

We provide the following tips from the troubleshooting section of this guide:

- [Section 17, "I need help configuring my DSNs"](#)
- [Section 18, "How can I tell which odbc.ini is being used?"](#)
- In particular, please see [Section 19, "Testing your ODBC configuration"](#)

Chapter 2. Configuring RLS

1. Configuration overview

RLS configuration involves statically-defined, system settings as defined in the RLS configuration file (see `$GLOBUS_LOCATION/etc/globus-rls-server.conf`), settings changed temporarily at run-time using the RLS Admin tool (see `globus-rls-admin(1) -C option value` command), and finally LRC-to-RLI and RLI-to-RLI updates configured using the RLS Admin tool (see `globus-rls-admin(1) -a, -A, -d` commands).

2. Server configuration file (`globus-rls-server.conf`)

Configuration settings for the RLS are specified in the `globus-rls-server.conf` file. If the configuration file is not specified on the command line (see the `-c` option) then it is looked for in both:

- `$GLOBUS_LOCATION/etc/globus-rls-server.conf`
- `/usr/local/etc/globus-rls-server.conf` if `GLOBUS_LOCATION` is not set



Note

Command line options always override items found in the configuration file.

The configuration file is a sequence of lines consisting of a keyword, whitespace, and a value. Comments begin with `#` and end with a newline.

3. Basic configuration

Review the server configuration file `$GLOBUS_LOCATION/etc/globus-rls-server.conf` and change any options you want. The server man page `globus-rls-server(8)` has complete details on all options. The complete details are also provided later in this section.

A minimal configuration file for both an LRC and RLI server would be:

```
# Configure the database connection info
db_user      dbuser
db_pwd       dbpassword

# If the server is an LRC server
lrc_server    true
lrc_dbname    lrc1000

# If the server is an RLI server
rli_server    true
rli_dbname    rli1000 # Not needed if updated by Bloom filters

# Configure who can make requests of the server
acl .*: all
```

```
# RE matching grid-mapfile users or DNs from x509 certs
...
```

4. Host key and certificate configuration

The server uses a host certificate to identify itself to clients. By default this certificate is located in the files `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem`. Host certificates have a distinguished name of the form `/CN=host/FQDN`. If the host you plan to run the RLS server on does not have a host certificate, you must obtain one from your Certificate Authority. The RLS server must be run as the same user who owns the host certificate files (typically root). The location of the host certificate files may be specified in `$GLOBUS_LOCATION/etc/globus-rls-server.conf`:

```
rlscertfile      path-to-cert-file    # default /etc/grid-security/hostcert.pem
rlskeyfile       path-to-key-file   # default /etc/grid-security/hostkey.pem
```

It is possible to run the RLS server without authentication, by starting it with the `-N` option, and using URL's of the form `rlsn://server` to connect to it. Notice that the URL scheme is `rlsn` as opposed to `rls`.

It is generally recommended to run the server with a user account other than root for added security. In order to do so, you will need to create complimentary key and certificate files owned by a designated user account, `globus` for instance.

1. Begin by copying the `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem` to `/etc/grid-security/containercert.pem` and `/etc/grid-security/containerkey.pem`. Note that we use the prefix "container" to conform with the recommended naming scheme for other services distributed with the Globus Toolkit.

```
% cp /etc/grid-security/hostcert.pem /etc/grid-security/containercert.pem
% cp /etc/grid-security/hostkey.pem /etc/grid-security/containerkey.pem
```

2. Then change ownership of the files to the designated user account, `globus` in our example.

```
% chown globus /etc/grid-security/containercert.pem
% chown globus /etc/grid-security/containerkey.pem
```

3. Change the `rlskeyfile` and `rlscertfile` settings in the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) to reflect the appropriate filenames.

```
rlscertfile      /etc/grid-security/containercert.pem
rlskeyfile       /etc/grid-security/containerkey.pem
```

4. Finally, bear in mind that your certificate and key files must always have file permissions 644 and 400 respectively.

```
% ls -l /etc/grid-security/*.pem
-rw-r--r-- 1 globus gridstaff 818 Dec 8 2005 /etc/grid-security/containercer
-r----- 1 globus gridstaff 887 Dec 8 2005 /etc/grid-security/containerkey
-rw-r--r-- 1 root root 818 Dec 8 2005 /etc/grid-security/hostcert.pem
-r----- 1 root root 887 Dec 8 2005 /etc/grid-security/hostkey.pem
```

If authentication is enabled, RLI servers must include `acl` configuration options that match the identities of LRC servers that update it and that grant the `rli_update` permission to the LRCs.

5. Configuring LRC to RLI updates

One of the key benefits to using the RLS for managing replica location information is its distributed architecture. In a distributed deployment, one or more Local Replica Catalog (LRC) services will send updates of its contents to one or more Replica Location Index (RLI) services.

By default the installed LRC is not configured to send updates to any RLI, even the local RLI co-located with the local LRC. Use the [globus-rls-admin\(1\)](#) tool to configure the LRC to send updates to one or more RLI services.

- To configure the LRC to send uncompressed lists of its logical names to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -a rls://rli_host rls://lrc_host
```

- To configure the LRC to send compressed bitmaps (using Bloom filters) of its logical names to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -A rls://rli_host rls://lrc_host
```

- To configure the LRC to stop sending updates to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -d rls://rli_host rls://lrc_host
```



Note

While any given LRC is capable of sending uncompressed or compressed updates to any RLI. The RLI service must be configured to accept either uncompressed or compressed updates but not both. See the `rli_bloom-filter` setting of the RLS configuration file for more details.

There are tradeoffs between using uncompressed and compressed updates in your configuration. The advantage of using compressed updates, not surprisingly, is a significant reduction in network overhead and memory usage. As replica location mappings grow into the 10's of millions or more, the savings of using compressed updates becomes important. On the other hand, due to the compressed nature of the Bloom filter bitmap used to represent the logical names in the LRC, the wildcard query at the RLI cannot be supported when update compression is used.

6. Configuring the RLS Server for the WS MDS Index Service

The server package includes a script `$GLOBUS_LOCATION/libexec/aggrexec/globus-rls-aggregator-source.pl` that may be used as an Execution Aggregator Source by WS MDS. See [GT 4.2.1 Index Services](#) for more information on setting up and using the Execution Aggregator Source scripts in WS MDS. The script may be invoked as follows and will generate output in the format as depicted.

```
% $GLOBUS_LOCATION/libexec/aggrexec/globus-rls-aggregator-source.pl rls://mysite
<?xml version="1.0" encoding="UTF-8"?>
<rlsStats>
  <site>rls://mysite</site>
  <version>4.0</version>
  <uptime>03:08:15</uptime>
  <serviceList>
    <service>lrc</service>
    <service>rli</service>
  </serviceList>
  <lrc>
    <updateMethodList>
      <updateMethod>lfnlist</updateMethod>
      <updateMethod>bloomfilter</updateMethod>
    </updateMethodList>
    <updatesList>
      <updates>
        <site>rls://myothersite:39281</site>
        <method>bloomfilter</method>
        <date>08/01/05</date>
        <time>16:16:38</time>
      </updates>
    </updatesList>
    <numlfn>283902</numlfn>
    <numpfn>593022</numpfn>
    <nummap>593022</nummap>
  </lrc>
  <rli>
    <updatedViaList>
      <updatedVia>bloomfilters</updatedVia>
    </updatedViaList>
    <updatedByList>
      <updatedBy>
        <site>rls://myothersite:39281</site>
        <date>08/01/05</date>
        <time>10:03:21</time>
      </updatedBy>
    </updatedByList>
  </rli>
</rlsStats>
```

Important

Be sure to configure the security context of the container running the MDS, and be sure that the security configuration on the RLS host recognizes the MDS security context.

When following the instructions provided by the [GT 4.2.1 Index Services](#), you will need to consider the security context used by the MDS to invoke the Execution Aggregator Source script provided by RLS. Most deployments of RLS run the service with security enabled. Therefore any client connections, including administrative status operations, require authentication and authorization. In order for MDS to use the provided script to check RLS status, it must invoke the script with a valid user proxy or user certificate and key. The RLS must recognize the DN from the user certificate (i.e., the DN should be in the gridmap file).

One way to configure the MDS security context for use with RLS monitoring is to set the environment variables `X509_USER_CERT` and `X509_USER_KEY` to point to the container certificate and key. Run the MDS with these environment settings. Also, add the DN from the container certificate to the gridmap file on the host running the RLS.

Alternatively, you could modify the provided script so that it sets the environment variables to another user certificate and key (or proxy) as desired before calling the RLS.

7. Configuring the RLS Server for the MDS2 GRIS

The server package includes a program called `globus-rls-reporter` that will report information about an RLS server to the MDS2 GRIS. Use this procedure to enable this program:

1. To enable Index Service reporting, add the contents of the file `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf` to the MDS2 GRIS configuration file `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf`.
2. If necessary, set your virtual organization (VO) name in `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf`. The default value is `local`. The VO name is referenced twice, on the lines beginning `dn:` and `args:`.
3. You must restart your MDS (GRIS) server after modifying `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf`. You can use the following commands to do so:

```
$GLOBUS_LOCATION/sbin/SXXgris stop
$GLOBUS_LOCATION/sbin/SXXgris start
```

8. Complete RLS Server settings (globus-rls-server.conf)

This section describes the complete details of the RLS Server configuration settings.

Table 2.1. Complete RLS Server settings (globus-rls-server.conf)

<pre>acl user: permission [permission]</pre>	<p>acl entries may be a combination of DNs and local usernames. If a DN is not found in the gridmap file then the file is used to search the acl list.</p> <p>A gridmap file may also be used to map DNs to local usernames, which in turn are matched against the regular expressions in the acl list to determine the user's permissions.</p> <p>user is a regular expression matching distinguished names (or local usernames if a gridmap file is used) of users allowed to make calls to the server.</p> <p>There may be multiple acl entries, with the first match found used to determine a user's privileges.</p> <p>[permission] is one or more of the following values:</p> <ul style="list-style-type: none"> • lrc_read Allows client to read an <u>LRC</u>. • lrc_update Allows client to update an LRC. • rli_read Allows client to read an <u>RLI</u>. • rli_update Allows client to update an RLI. • admin Allows client to update an LRC's list of RLIs to send updates to. • stats Allows client to read performance statistics. • all Allows client to do all of the above.
<pre>authentication true false</pre>	<p>Enable or disable GSI authentication.</p> <p>The default value is true.</p> <p>If authentication is enabled (true), clients should use the URL schema rls: to connect to the server.</p> <p>If authentication is <i>not</i> enabled (false), clients should use the URL schema rlsn:.</p>
<pre>db_pwd password</pre>	<p>Password to use to connect to the database server.</p> <p>The default value is changethis.</p>
<pre>db_user data-baseuser</pre>	<p>Username to use to connect to database server.</p> <p>The default value is dbperson.</p>
<pre>idletimeout seconds</pre>	<p>Seconds after which idle connections close.</p> <p>The default value is 900.</p>
<pre>loglevel N</pre>	<p>Sets loglevel to N (default is 0). Higher levels mean more verbosity.</p>

lrc_bloomfilter_numhash N	<p>Number of hash functions to use in <i>Bloom filters</i>.</p> <p>The default value is 3.</p> <p>Possible values are 1 through 8.</p> <p>This value, in conjunction with <code>lrc_bloomfilter_ratio</code>, will determine the number of false positives that may be expected when querying an RLI that is updated via Bloom filters.</p> <p><i>Note:</i> The default values of 3 and 10 give a false positive rate of approximately 1%.</p>
lrc_bloomfilter_ratio N	<p>Sets ratio of bloom filter size (in bits) to number of <i>LFNs</i> in the LRC catalog (in other words, size of the Bloom filter as a multiple of the number of LFNs in the LRC database.) This is only meaningful if Bloom filters are used to update an RLI. Too small a value will generate too many false positives, while too large a value wastes memory and network bandwidth.</p> <p>The default value is 10.</p> <p><i>Note:</i> The default values of 3 and 10 give a false positive rate of approximately 1%.</p>
lrc_buffer_time N	<p>LRC to RLI updates are buffered until either the buffer is full or this much time in seconds has elapsed since the last update.</p> <p>The default value is 30.</p>
lrc_dbname	<p>Name of LRC database.</p> <p>The default value is <code>lrcdb</code>.</p>
lrc_server true false	<p>If LRC server, the value should be <code>true</code>.</p> <p>The default value is <code>false</code>.</p>
lrc_update_bf seconds	<p>Interval in seconds between LRC to RLI updates when the RLI is updated by Bloom filters. In other words, how often an LRC server does a Bloom filter soft state update.</p> <p>This can be much smaller than the interval between updates without using Bloom filters (<code>lrc_update_ll</code>).</p> <p>The default value is 300.</p>
lrc_update_factor N	<p>If <code>lrc_update_immediate</code> mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (<code>lrc_update_ll</code>) is multiplied by the value of this option.</p>
lrc_update_immediate true false	<p>Turns LRC to RLI immediate mode updates on (<code>true</code>) or off (<code>false</code>).</p> <p>The default value is <code>false</code>.</p>
lrc_update_ll seconds	<p>Number of seconds before an LRC server does an LFN list soft state update.</p> <p>The default value is 86400.</p>

<code>lrc_update_retry seconds</code>	Seconds to wait before an LRC server will retry to connect to an RLI server that it needs to update. The default value is 300.
<code>maxbackoff seconds</code>	Maximum seconds to wait before re-trying listen in the event of an I/O error. The default value is 300 .
<code>maxfreethreads N</code>	Maximum number of idle threads. Excess threads are killed. The default value is 5 .
<code>maxconnections N</code>	Maximum number of simultaneous connections. The default value is 100.
<code>maxthreads N</code>	Maximum number of threads running at one time. The default value is 30.
<code>myurl URL</code>	URL of server. The default value is <code>rls://<hostname>:port</code>
<code>odbcini filename</code>	Sets environment variable ODBCINI. If not specified, and ODBCINI is not already set, then the default value is <code>\$GLOBUS_LOCATION/var/odbc.ini</code> .
<code>pidfile filename</code>	Filename where pid file should be written. The default value is <code>\$GLOBUS_LOCATION/var/<programname>.pid</code> .
<code>port N</code>	Port the server listens on. The default value is 39281 .
<code>result_limit limit</code>	Sets the maximum number of results returned by a query. The default value is 0 (zero), which means no limit. If a query request includes a limit greater than this value, an error (<code>GLOBUS_RLS_BADARG</code>) is returned. If the query request has no limit specified, then at most <code>result_limit</code> records are returned by a query.
<code>rli_bloomfilter true false</code>	RLI servers must have this set to accept Bloom filter updates. If <code>true</code> , then only Bloom filter updates are accepted from LRCs. If <code>false</code> , full LFN lists are accepted. <i>Note:</i> If Bloom filters are enabled, then the RLI does <i>not</i> support wildcarded queries.

<p><code>rli_bloomfilter_dir</code> <code>none</code> <code>default</code> <code>path-</code> <code>name</code></p>	<p>If an RLI is configured to accept bloom filters (<code>rli_bloomfilter true</code>), then Bloom filters may be saved to this directory after updates.</p> <p>This directory is scanned when an RLI server starts up and is used to initialize Bloom filters for each LRC that updated the RLI.</p> <p>This option is useful when you want the RLI to recover its data immediately after a restart rather than wait for LRCs to send another update.</p> <p>If the LRCs are updating frequently, this option is unnecessary and may be wasteful in that each Bloom filter is written to disk after each update.</p> <ul style="list-style-type: none"> • <code>none</code> Bloom filters are not saved to disk. This is the default. • <code>default</code> Bloom filters are saved to the default directory: <ul style="list-style-type: none"> • <code>\$GLOBUS_LOCATION/var/rls-bloomfilters</code> if <code>GLOBUS_LOCATION</code> is set • <code>else, /tmp/rls-bloomfilters</code> • <code>pathname</code> Bloom filters are saved to the named directory. Any other string is used as the directory name unchanged. The Bloom filter files in this directory have the name of the URL of the LRC that sent the Bloom filter, with slashes(/) changed to percent signs (%) and ".bf" appended.
<p><code>rli_dbname</code> <code>database</code></p>	<p>Name of the RLI database.</p> <p>The default value is <code>rliadb</code>.</p>
<p><code>rli_expire_int</code> <code>seconds</code></p>	<p>Interval (in seconds) between RLI expirations of stale entries. In other words, how often an RLI server will check for stale entries in its database.</p> <p>The default value is <code>28800</code>.</p>
<p><code>rli_expire_stale</code> <code>seconds</code></p>	<p>Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC).</p> <p>The default value is <code>86400</code>.</p> <p>This value should be no smaller than <code>lrc_update_ll</code>.</p> <p>Stale RLI entries are not returned in queries.</p> <p><i>Note:</i> If the LRC server is responding, this value is not used. Instead the value of <code>lrc_update_ll</code> or <code>lrc_update_bf</code> is retrieved from the LRC server, multiplied by 1.2, and used as the value for this option.</p>

Configuring RLS

<code>rli_server</code> <code>true false</code>	If an RLI server, the value should be <code>true</code> . The default value is <code>false</code> .
<code>rlscertfile filename</code>	Name of the X.509 certificate file identifying the server. This value is set by setting environment variable <code>X509_USER_CERT</code> .
<code>rlskeyfile filename</code>	Name of the X.509 key file for the server. This value is set by setting environment variable <code>X509_USER_KEY</code> .
<code>startthreads N</code>	Number of threads to start initially. The default value is 3.
<code>timeout seconds</code>	Timeout (in seconds) for calls to other RLS servers (e.g., for LRC calls to send an update to an RLI).

Chapter 3. Starting the RLS Server

The RLS Server may be started by directly invoking the executable and passing it command-line arguments or by using the startup script. For complete details on starting the server directly, refer to [globus-rls-server\(1\)](#). We provide a startup script found at `$GLOBUS_LOCATION/sbin/SXXrls`, which you may use in places like `/etc/init.d` for automated startup and shutdown of the server. You may start the RLS Server by using this script as follows:

```
% $GLOBUS_LOCATION/sbin/SXXrls start
```

In addition to the startup script, the RLS Server may be started by using the executable as follows:

```
% $GLOBUS_LOCATION/bin/globus-rls-server [-d] [-L log-level]
```

We provide the following tips from the troubleshooting section of this guide:

- [Section 20, “Important notes on RLS initialization”](#)

Chapter 4. Stopping the RLS Server

The RLS Server may be stopped by directly issuing a KILL signal, by using the RLS Admin client, or by using the startup script.

If you have started the RLS Server directly, using `globus-rls-server(1)`, and you have not detached the process, you may issue a CTRL-C in the terminal window. If the server is running detached or was started using the startup script, you may issue a KILL signal to the `globus-rls-server` process. The RLS Server will catch the KILL signal and safely shutdown.

Using the RLS Admin client (see `globus-rls-admin(1)` for details), you may shutdown the RLS Server using the following command (this is also effective for shutting down remote RLS Servers):

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -q rls[n]://hostname:port
```

Using the RLS Server startup script found at `$GLOBUS_LOCATION/sbin/SXXrls`, you may shutdown the RLS Server using the following command:

```
% $GLOBUS_LOCATION/sbin/SXXrls stop
```

Chapter 5. Testing

1. Manual Tests

You can use the programs `globus-rls-admin` and `globus-rls-cli` to test functionality. See their respective `man` pages for details on their use.

1. Start the server in debug mode with the command:

```
$GLOBUS_LOCATION/bin/globus-rls-server [-N] -d -L 3
```

The `-N` option is helpful: if you do not have a host certificate for the server host, or a user certificate for yourself, it disables authentication.

2. Ping the server using `globus-rls-admin`:

```
$GLOBUS_LOCATION/bin/globus-rls-admin -p rls://serverhost
```

If you disabled authentication (by starting the server with the `-N` option), then use this command:

```
$GLOBUS_LOCATION/bin/globus-rls-admin -p rlsn://serverhost
```

2. Automated Tests

Run the following automated test script to perform an exhaustive unit test of the RLS. If the script indicates failures or errors, open the test output folder (as indicated by the script) to see detailed test results.

```
$GLOBUS_LOCATION/test/globus_rls_test/TESTS.pl
```

Chapter 6. Security Considerations

1. Replica Location Service (RLS) Security Considerations

Security recommendations include:

- *Dedicated User Account:* It is recommended that users create a dedicated user account for installing and running the RLS service (e.g., `globus` as recommended in the general GT installation instructions). This account may be used to install and run other services from the Globus Toolkit.
- *Key and Certificate:* It is recommended that users do not use their `hostkey` and `hostcert` for use by the RLS service. Create a `containerkey` and `containercert` with permissions 400 and 644 respectively and owned by the `globus` user. Change the `rlskeyfile` and `rlscertfile` settings in the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) to reflect the appropriate filenames.
- *LRC and RLI Databases:* Users must ensure security of the RLS data as maintained by their chosen database management system. Appropriate precautions should be made to protect the data and access to the database. Such precautions may include creating a user account specifically for RLS usage, encrypting database users' passwords, etc.
- *RLS Configuration:* It is recommended that the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) be owned by and accessible only by the dedicated user account for RLS (e.g., `globus` account per above recommendations). The file contains the database user account and password used to access the LRC and RLI databases along with important settings which, if tampered with, could adversely affect the RLS service.

Chapter 7. Troubleshooting

General information on troubleshooting can be found in the [FAQ](#)¹. For a list of common errors in GT, see [Error Codes](#).

1. Errors

Table 7.1. Replica Locator Service (RLS) Errors

Error Code	Definition	Possible Solutions
Error with credential: The proxy credential: <credential> with subject: <subject> expired <minutes> minutes ago	Expired proxy credential	Create a new proxy with <code>grid-proxy-init</code> .
Unable to connect to localhost:xxxx	Unable to connect to the local host. This can be due to a variety of reasons, including a wrong address or port number in the RLS connection URL or an issue with a firewall configuration.	<ul style="list-style-type: none">• Double-check the address and port number in the RLS connection URL. parameters are correct.• If a firewall configuration is preventing connections to the target host for a particular port, you may need to consult the system administrator.
"connection timeout"	<p>At times, a client may experience a connection timeout when interacting with the RLS server due to a variety of reasons:</p> <ul style="list-style-type: none">• One reason could simply be due to wide-area network latency or congestion.• Another situation that users eventually encounter is due to scaling of the system. As the RLS server's database of replica location mappings grows in size, some query operations, such as bulk queries involving large quantities of mappings or wildcard queries that result in a large subset of mappings, will begin to take more time both to process the query and to return the large results set to the client over the network.	If timeouts are experienced with increasing frequency, increase the RLS server's timeout configuration parameter found in the <code>\$GLOBUS_LOCATION/var/globus-rls-server.conf</code> file. You may also use the <code>-t</code> timeout option of the <code>globus-rls-cli</code> tool.

¹ http://www.globus.org/toolkit/data/rls/rls_faq.html

2. I need help installing MySQL DBMS

There is simply too much information on DBMS installation for us to provide useful tips. In general, a binary package with a native installer (e.g., RPM, PKG, etc.) is the best option if you have root privileges. A binary package without a native installer is an excellent option if you do not have root privileges. And if all else fails, you can always try source installers. We have found most source installers of the DBMS to be reliable.

We have used RLS with MySQL versions 4.0.x, 4.1.x, and 5.0.x.

3. How can I check if MySQL is installed correctly?

There are probably numerous things you could do, but one simple check is to see if you can log on to the database. Of course, this requires that you have a user account. Assuming you installed the DBMS yourself, you should be able to log on as root:

```
% /path/to/mysql/bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 64 to server version: 4.0.18-standard-log
```

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

```
mysql> show databases;
+-----+
| Database          |
+-----+
| mysql             |
| test              |
+-----+
2 rows in set (0.03 sec)
```

```
mysql> quit
Bye
```

If you did not install the DBMS yourself but were given an account on a remote server, you should be able to log on using your user account:

```
% /path/to/mysql/bin/mysql -h hostname -P 3306 -u dbuser -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 66 to server version: 4.0.18-standard-log
```

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

```
mysql>
```

4. I need help installing PostgreSQL DBMS

There is simply too much information on DBMS installation for us to provide useful tips. In general, a binary package with a native installer (e.g., RPM, PKG, etc.) is the best option if you have root privileges. A binary package without a native installer is an excellent option if you do not have root privileges. And if all else fails, you can always try source installers. We have found most source installers of the DBMS to be reliable.

We have used RLS with PostgreSQL versions 7.4.x and it seems to work well with 8.x also.

5. How can I check if PostgreSQL is installed correctly?

There are probably numerous things you could do, but one simple check is to see if you can log on to the DBMS. Of course, this requires that you have a user account. Assuming you installed the DBMS yourself, you should be able to log on as root:

```
% /path/to/pgsql/bin/psql
Welcome to psql 7.4.6, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

user=#
```

If you did not install the DBMS yourself but were given an account on a remote server, you should be able to log on using your user account:

```
% /path/to/pgsql/bin/psql -h hostname -p 5432 -U dbuser -W dbname
Password:
Welcome to psql 7.4.6, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

dbname=>
```

6. I cannot connect to PostgreSQL

If you cannot connect to PostgreSQL but you know that the process is alive, you may have forgotten to use the `-i` when starting the DBMS. This option instructs `postmaster` to listen on TCP/IP socket. Without it, `postmaster` will not accept socket connections from clients.

7. Performance of RLS with PostgreSQL has degraded over time

Be sure to do periodic `vacuum` and `analyze` commands on all your PostgreSQL databases. The PostgreSQL documentation recommends doing this daily from `cron`. Failure to do this can seriously degrade performance, to the point where routine RLS operations (such as LRC to RLI soft state updates) timeout and fail. Please see the PostgreSQL documentation for further details.

8. I need help installing SQLite embedded database

Please be aware that we have relatively limited exposure to using RLS with SQLite than other database management systems. As of the GT 4.2 development branch, we have added SQLite setup scripts to create the RLS schema in a sqlite embedded database. Our testing has been limited to various Linux flavors on 32-bit platforms.

You are encouraged to review the SQLite site for authoritative details on installation of `sqlite3`. The following commands have been used by us to install `sqlite-3.3.6`:

```
% tar zxvf sqlite-3.3.6.tar.gz
% mkdir bld
% cd bld
% ../sqlite-3.3.6/configure --prefix=/path/to/install --enable-threadsafe
% make
% make install
```

9. How can I check if SQLite is installed correctly?

There are probably numerous things you could do, but one simple check is to see if you can open a test database. You should be able to open a test database and perform various SQL operations:

```
% /path/to/sqlite/bin/sqlite3 test.db
SQLite version 3.3.6
Enter ".help" for instructions
sqlite>
```

10. I need help installing iODBC

The `iodbc.org` website contains information on installation of the iODBC (or `libiodbc`) library. We have used the following commands to install iODBC version 3.52.4 in our development and test sites. From the `libiodbc` build directory:

```
% ./configure --prefix=/path/to/libiodbc-3.52.4 \
--with-iodbc-inidir=/path/to/etc --disable-gui --disable-gtktest
```

```
% make
% make install
```

The configure parameters are optional. The defaults enable the features required by RLS, namely pthreads. We prefer to install to a specified directory due to testing against multiple ODBC managers, however, this would not apply to most end users. We also prefer not to install the GUI features.

Keep in mind, if you install the ODBC Manager library to a non-standard location, you will need to update your library path (e.g., LD_LIBRARY_PATH) to reflect your non-standard installation.

11. I need help installing unixODBC

The unixodbc.org website contains information on installation of the unixODBC library. We have used the following commands to install unixODBC version 2.2.11 in our development and test sites. From the unixODBC build directory:

```
% ./configure --prefix=/path/to/unixODBC --disable-gui \
  --disable-drivers
% make
% make install
```

The configure parameters are optional. The defaults enable the features required by RLS, namely threads. We prefer to install to a specified directory due to testing against multiple ODBC managers, however, this would not apply to most end users. We also prefer not to install the GUI features or the supplied drivers.

Keep in mind, if you install the ODBC Manager library to a non-standard location, you will need to update your library path (e.g., LD_LIBRARY_PATH) to reflect your non-standard installation.

12. Building unixODBC failed on OS X

Building unixODBC-2.2.11 on OS X failed for us too. Our unixODBC tarball came from the unixodbc.org project site. It may be that other sources of this tarball may have been ported properly. What we found by doing a brief search was that making a small change to one of the files corrected the problem. In the file `Drivers/txt/SQLTables.c`, we made the following change from:

```
#ifdef OSXHEADER
#include <i386/types.h>
#endif
```

to:

```
/*
#ifdef OSXHEADER
#include <i386/types.h>
#endif
*/
```

13. I need help installing MySQL Connector/ODBC

The MySQL project/company provide several binary packages for several major platforms using a variety of native installers. We recommend using this approach if possible, though it may require root privileges and help from your system administrator. There are also binary packages available that do not use native installers, which can be used without root privileges and provide a better option than source installation.

From what we can tell, the 3.51.12 version of MySQL Connector/ODBC from the binary bundles provided by MySQL will work with unixODBC-2.2.11 but not with libiodbc-3.52.4 or earlier. The last known combinations of drivers from MySQL that worked with iODBC that we know of are:

- MyODBC-3.51.06 and libiodbc-3.51.1
- MyODBC-3.51.06 and libiodbc-3.51.2

Unfortunately, MySQL does not provide links to packages of MyODBC-3.51.06 as far as we can tell. You may find mirrors of the MySQL site that still have this driver available by searching the web for "myodbc 3.51.06". We found this one.²

13.1. MySQL Connector/ODBC 3.51.12 (binary)

With mysql-connector-odbc-3.51.12-linux-i686, we have used the following commands to install the driver.

```
% tar -zxvf /path/tomysql-connector-odbc-3.51.12-linux-i686.tar.gz
```

We have successfully used this driver with unixODBC-2.2.11. However, when testing against libiodbc-3.52.4 using the `iodbctest` utility, the driver failed with the following error.

```
% ./bin/iodbctest "DSN=lrc1000_mysql_3_51_12;UID=dbpassword;PWD:dbpassword"
iODBC Demonstration program
This program shows an interactive SQL processor
Driver Manager: 03.52.0406.0126
1: SQLDriverConnect = [MySQL][ODBC 3.51 Driver]Invalid window handle for
connection completion argument. (0) SQLSTATE=IM008
1: ODBC_Connect = [MySQL][ODBC 3.51 Driver]Invalid window handle for connection
completion argument. (0) SQLSTATE=IM008
```

Have a nice day.

We did not "Have a nice day" after this. There is limited information on this error, though it did appear that the driver loaded and attempted to make a connection.

² <http://sunsite.mff.cuni.cz/MIRRORS/ftp.mysql.com/downloads/api-myodbc-3.51.html>

13.2. MySQL Connector/ODBC 3.51.12 (source)

We attempted building and installing the mysql-connector-odbc-3.51.12 driver from the source package. We attempted building against unixODBC and libiodbc libraries and mysql-4.0.18 and mysql-5.0.21. We used a variety of configure options. In the end, we encountered configure or compiler errors that prevented us from using the source package.

13.3. MyODBC 3.51.06 (binary)

With MyODBC-3.51.06-pc-linux-gnu-i686, we have used the following commands to install the driver.

```
% tar -zxvf /path/to/MyODBC-3.51.06-pc-linux-gnu-i686
```

We have successfully used this driver with unixODBC-2.2.11 and libiodbc-3.52.4.

13.4. MyODBC 3.51.06 (source) and unixODBC

With MyODBC-3.51.06, we have used the following commands to install myodbc in our development and test sites. From the MyODBC-3.51.06 build directory:

```
% ./configure --prefix=/path/to/odbc/Drivers/MyODBC-3.51.06-unixODBC \
--with-unixODBC --with-unixODBC-includes=/path/to/odbc/unixODBC-2.2.11/include \
--with-unixODBC-libs=/path/to/odbc/unixODBC-2.2.11/lib \
--with-odbc-ini=/path/to/odbc/etc/odbc.ini \
--with-mysql-includes=/path/to/mysql/include --with-mysql-libs=/path/to/mysql/lib \
--enable-thread-safe --without-samples
% gmake
% gmake install
```

Note: when building against unixODBC you must use the `--with-unixODBC` flag even if you use the `--with-unixODBC-includes=DIR` and `--with-unixODBC-libs=DIR` parameters.

You must use the `--enable-thread-safe` flag.

Also, be prepared to see a large number of compiler warnings when building this library.

13.5. MyODBC 3.51.06 (source) and iODBC

With MyODBC-3.51.06, we have used the following commands to install myodbc in our development and test sites. From the MyODBC-3.51.06 build directory:

```
% ./configure --prefix=/path/to/odbc/Drivers/MyODBC-3.51.06-iODBC \
--with-iodbc=/path/to/libiodbc-3.51.2 --with-odbc-ini=/path/to/odbc/etc/odbc.ini \
--with-mysql-includes=/path/to/mysql/include --with-mysql-libs=/path/to/mysql/lib \
--enable-thread-safe --without-samples
% gmake
% gmake install
```

Note: when building against iODBC you must use the `--with-iodbc=DIR` parameter. The `--with-iodbc-
includes=DIR` and `--with-iodbc-libs=DIR` parameters DO NOT work.

You must use the `--enable-thread-safe` flag.

Also, be prepared to see a large number of compiler warnings when building this library.

14. I need help installing psqLODBC

The README file or docs directory included with the psqLODBC package contains information on installation of the psqlodbc library. We have noticed some differences between building and installing psqlodbc-7.2.5, psqlodbc-08.00.0102, and psqlodbc-08.01.0200 against iODBC and unixODBC.

Note: In earlier versions of this guide, we warned that the `--with-iodbc` configure option resulted in an unstable psqlodbc library that caused memory corruption and failure (e.g., core dump) when an application (RLS Server) attempted to open the database connection. We have not experienced this problem with the more recent versions of psqlodbc that we have tested and which are documented below.

14.1. psqlodbc-7.2.5 (source) and unixODBC-2.2.11

With psqlodbc-7.2.5, we have used the following commands to install psqlodbc in our development and test sites. From the psqlodbc build directory:

```
% ./configure --prefix=/path/to/odbc/Drivers/psqlodbc-7.2.5-unixODBC \  
--enable-pthreads --with-unixodbc  
% gmake  
% gmake install
```

You may also want to consider setting CPPFLAGS and LDFLAGS if you install your ODBC Manager in a non-standard location.

14.2. psqlodbc-7.2.5 (source) and libiodbc-3.52.4

With psqlodbc-7.2.5, we have used the following commands to install psqlodbc in our development and test sites. From the psqlodbc build directory:

```
% ./configure --prefix=/path/to/odbc/Drivers/psqlodbc-7.2.5-iODBC \  
--enable-pthreads --with-iodbc  
% gmake  
% gmake install
```

You may also want to consider setting CPPFLAGS and LDFLAGS if you install your ODBC Manager in a non-standard location.

14.3. psqlodbc-08.00.0102 (source) and unixODBC-2.2.11

With psqlodbc-08.00.0102, we have used the following commands install psqLODBC with a unixODBC-2.2.11 installation. From the psqlodbc build directory:

```
% ./configure --prefix=/path/to/odbc/Drivers/psqlodbc-08.00.0102-unixODBC \
--enable-pthreads --with-unixodbc LDFLAGS="-L/path/to/odbc/unixODBC-2.2.11/lib" \
CPPFLAGS="-I/path/to/odbc/unixODBC-2.2.11/include"
% gmake
% gmake install
```

14.4. psqlodbc-08.00.0102 (source) and libiodbc-3.52.4

With psqlodbc-08.00.0102, we have used the following commands to configure psqlodbc but it failed to build. We have no advice on resolving this problem other than to suggest using a different combination of ODBC Manager and ODBC Driver.

```
% ./configure --prefix=/path/to/odbc/Drivers/psqlodbc-08.00.0102-iODBC \
--enable-pthreads --with-iodbc LDFLAGS=-L/path/to/odbc/libiodbc-3.52.4/lib \
CPPFLAGS=-I/path/to/odbc/libiodbc-3.52.4/include
% gmake
...ends in errors...
```

14.5. psqlodbc-08.00.0102 (source) and libiodbc-3.51.2

With psqlodbc-08.00.0102, we have used the following commands to configure psqlodbc and to build it successfully. In this case, we used psqlodbc along with libiodbc-3.51.2.

```
% ./configure --prefix=/path/to/odbc/Drivers/psqlodbc-08.00.0102-iODBC \
--enable-pthreads --with-iodbc LDFLAGS=-L/path/to/odbc/libiodbc-3.51.2/lib \
CPPFLAGS=-I/path/to/odbc/libiodbc-3.51.2/include
% gmake
% gmake install
```

14.6. psqlodbc-08.01.0200 (source)

We have NOT succeeded installing psqlodbc-08.01.0200 from source with either iODBC or unixODBC.

15. I need help installing SQLite ODBC Driver

The README file included with the SQLite ODBC Driver package contains information on installation of the sqliteodbc library. Please be aware that the author of the SQLite ODBC Driver states that the package is of experimental quality.

15.1. sqliteodbc-0.69 (source) and iODBC-3.51.2

With sqliteodbc-0.69, we have used the following commands to install sqliteodbc in our development and test sites. From the sqliteodbc build directory:

```
% ./configure --prefix=/path/to/install --disable-winterface --with-sqlite3=/path/to/sqlite3
```

```
% make
% make install
```

Note the distinction between using `--with-sqlite=DIR` and `--with-sqlite3=DIR`. Use the latter for the sqlite3 database library (recommended).

16. Which libraries do you recommend to new users?

The short answer, consider the following options:

- PostgreSQL (recent stable) with `psqlODBC (08.00.0102)` with `unixODBC (2.2.11)`
- MySQL (recent stable) with `MySQL Connector/ODBC (3.51.12, binary package)` with `unixODBC (2.2.11)`

The long answer follows. First of all, if you are a new user and not confident installing system level software, you will save yourself a lot of time and trouble by consulting with your system administrator. If you are an administrator and you still find this troubling, then you are in good company.

As of the writing of this guide, we have experienced a higher success rate using `unixODBC` (current stable release at this time is version 2.2.11) than we have with `iODBC` (current stable release is version 3.52.4).

As of the writing of this guide, we have found PostgreSQL and `psqlODBC` installation from source to be reliable. We would recommend the use of `psqlODBC` version 08.00.0102 with `unixODBC`. If you are an `iODBC` user, you will need to use `psqlODBC 7.2.5`.

As of the writing of this guide, we have found MySQL and `MySQL Connector/ODBC` installation from binary to be reliable. We would recommend the use of `MySQL Connector/ODBC` version 3.51.12 with `unixODBC`. If you are a `iODBC` user, you will need to find a binary of `MyODBC 3.51.06`.

17. I need help configuring my DSNs

There are a variety of ways that you can specify the Data Source Names (DSNs) in the `odbc.ini` file. You may use a GUI or command-line tool provided with your ODBC Manager (`unixODBC` or `iODBC`). We tend to edit the `odbc.ini` file directly when working in our development and test sites, however, you should avoid this practice unless you are experienced with using ODBC tools (a small typo such as an unwanted space character can prevent your configuration from working -- with little in the way of useful error messages).

The following is an example of a typical setup when using PostgreSQL:

```
[lrc1000]
Driver      = /path/to/lib/psqlodbc.so
Database    = lrc1000
Servername  = hostname
Port        = 5432

[rli1000]
Driver      = /path/to/lib/psqlodbc.so
Database    = rli1000
Servername  = hostname
```

Port = 5432

The following is an example of a typical setup when using MySQL:

```
[lrc1000]
Driver      = /path/to/lib/libmyodbc3.so
Database    = lrc1000
SERVER      = hostname
PORT        = 3306
SOCKET      = /path/to/mysql/mysql.sock
```

```
[rli1000]
Driver      = /path/to/lib/libmyodbc3.so
Database    = rli1000
SERVER      = hostname
PORT        = 3306
SOCKET      = /path/to/mysql/mysql.sock
```

The following is an example of a typical setup when using SQLite:

```
[lrc1000]
Driver      = /path/to/lib/libsqlite3odbc.so
Database    = /path/to/globus-rls-lrc1000.db
```

```
[rli1000]
Driver      = /path/to/lib/libsqlite3odbc.so
Database    = /path/to/globus-rls-rli1000.db
```

18. How can I tell which odbc.ini is being used?

The ODBC libraries look for the odbc.ini file along a search path. We are not experts in this matter and it seems that it varies from vendor to vendor over time. But to the best of our knowledge the general search path is in the following order:

- According to environment variable \$ODBCINI
- In the user's home at ~/.odbc.ini
- In the system default /etc/odbc.ini

When we are troubleshooting a system, we tend to use the `strace` utility to see exactly which files it is reading.

19. Testing your ODBC configuration

The commands below demonstrate the `show tables` command with a MySQL database. Use the equivalent of this command if you are using a different DBMS. Alternately, you may simply perform a few `SELECT` queries against the listed tables.

Testing with unixODBC, run:

```
% /path/to/unixODBC-2.2.11/bin/isql lrc1000 dbuser dbpassword
+-----+
| Connected!                               |
| sql-statement                             |
| help [tablename]                         |
| quit                                       |
+-----+
SQL> show tables;
+-----+
| Tables_in_lrc1000|
+-----+
| t_attribute      |
| t_date_attr     |
| tflt_attr       |
| t_int_attr      |
| t_lfn           |
| t_map           |
| t_pfn           |
| t_rli           |
| t_rlipartition  |
| t_str_attr      |
+-----+
SQLRowCount returns 10
10 rows fetched
SQL> quit
```

Testing with iODBC, run:

```
% /path/to/bin/iodbctest "DSN=lrc1000;UID=dbuser;PWD=dbpassword"
iODBC Demonstration program
This program shows an interactive SQL processor
Driver Manager: 03.51.0002.0224
Driver: 03.51.06

SQL>show tables;

Tables_in_lrc1000
-----
t_attribute
t_date_attr
tflt_attr
t_int_attr
t_lfn
t_map
t_pfn
t_rli
t_rlipartition
t_str_attr
```

```
result set 1 returned 10 rows.
```

```
SQL>quit
```

```
Have a nice day.
```

20. Important notes on RLS initialization

Please be advised (and advise other users responsible for bringing up the RLS) that the startup initialization may take a few minutes before the RLS may be accessible. The initialization involves two key operations that may consume significant resources causing the server to appear temporarily unresponsive. Users of RLS may mistakenly assume that RLS failed to startup and may kill the server and start over. Some users may fall into this in a repeated cycle, believing that the RLS is unable to startup properly.

If the RLS is configured to send compressed updates (Bloom filters) to other RLIs, the RLS startup will involve initialization of the Bloom filter representing the current contents of the local replica catalog (LRC). This step is a prerequisite before any additional operations may be allowed, therefore no client connections are permitted until the initialization is complete. In our test environment, we have seen over 30 seconds delay due to creation of the Bloom filter corresponding to 1 million LFN names on a system with Dual 1 GHz CPU and 1.5 GB RAM. You may experience greater delays at larger scales and/or when running RLS with more limited system resources.

If the RLS is configured to send uncompressed updates (LFN lists) to other RLIs, the RLS startup will not involve any additional initialization delay. However, the RLS will spawn an initial full catalog update to all RLIs it updates. Though these updates will take place on separate threads of execution after the initialization of the system, they will consume a great amount of processor activity. Depending on the volume of the local replica catalog (LRC), this processor activity may initially interfere with a client operation. In our test environment, we have seen our initial "globus-rls-admin ping..." operation may suffer a delay and timeout in 30 seconds, the second "ping" may delay for a few seconds but will successfully return, and the third and every subsequent "ping" operation will successfully return immediately throughout the duration of the update. The system exhibits the same behavior for any other client operation, such as a `globus-rls-cli query...` operation.

21. Linux kernel and glibc incompatibility

This note originated based on observations of RLS Server running on RedHat 9 but could also apply to other Linux distributions. There have been occurrences of RLS servers hanging on RedHat 9 systems. The external symptoms are:

- The server does not accept new connections from clients, with an error message similar to:

```
connect(rls://XXXXX): globus_rls_client: IO timeout:
globus_io_tcp_register_connect() timed out after 30 seconds
```

- Often, the server continues to receive and send updates as configured and respond to signals. You can check this by querying other servers that interact with the one that's hung. Under `gdb`: All the server threads are waiting to be signaled on a condition variable. Sometimes, this is in `globus_io` functions, particularly in `globus_io_cancel()`.

21.1. Probable cause

This seems to be due to a problem in the new kernel and thread libraries of RedHat 9. A problem in `pthread_cond_wait()` causes threads not to wake up correctly. This problem has been seen with the following kernels and glibc packages:

- Kernels: 2.4.20-30.9 and 2.4.20-8
- glibc: glibc-2.3.2-27.9.7

21.2. Suggested workaround

The problems don't seem to arise when RLS is linked with older pthread libraries. This can be done as by adding a couple of lines to the RLS startup script in `$GLOBUS_LOCATION/sbin/SXXrls`, as shown:

```
#!/bin/sh

GLOBUS_LOCATION=/opt/gt3.2
MYSQL=/opt/mysql
IODBC=/opt/iodbc

export GLOBUS_LOCATION

#RedHat 9 workaround
LD_ASSUME_KERNEL=2.4.1
export LD_ASSUME_KERNEL
...
```

On i586 systems, set:

```
...
LD_ASSUME_KERNEL=2.2.5
...
```

Chapter 8. Usage statistics collection by the Globus Alliance

1. RLS-specific usage statistics

The following usage statistics are sent by RLS Server by default in a UDP packet:

- Component identifier
- Usage data format identifier
- Time stamp
- Source IP address
- Source hostname (to differentiate between hosts with identical private IP addresses)
- Version number
- Uptime
- *LRC* service indicator
- *RLI* service indicator
- Number of *LFNs*
- Number of *PFNs*
- Number of Mappings
- Number of RLI LFNs
- Number of RLI LRCs
- Number of RLI Senders
- Number of RLI Mappings
- Number of threads
- Number of connections

The RLS sends the usage statistics at server startup, server shutdown, and once every 24 hours when the service is running.

If you wish to disable this feature, you can set the following environment variable before running the RLS:

```
export GLOBUS_USAGE_OPTOUT=1
```

By default, these usage statistics UDP packets are sent to `usage-stats.globus.org:4180` but can be redirected to another host/port or multiple host/ports with the following environment variable:

```
export GLOBUS_USAGE_TARGETS="myhost.mydomain:12345 myhost2.mydomain:54321"
```

You can also dump the usage stats packets to stderr as they are sent (although most of the content is non-ascii). Use the following environment variable for that:

```
export GLOBUS_USAGE_DEBUG=MESSAGES
```

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ ../../Usage_Stats.html

Glossary

B

Bloom filter Compression scheme used by the Replica Location Service (RLS) that is intended to reduce the size of soft state updates between Local Replica Catalogs (LRCs) and Replica Location Index (RLI) servers. A Bloom filter is a bit map that summarizes the contents of a Local Replica Catalog (LRC). An LRC constructs the bit map by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits.

L

Local Replica Catalog (LRC) Stores mappings between logical names for data items and the target names (often the physical locations) of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. Also may associate attributes with logical or target names. Each LRC periodically sends information about its logical name mappings to one or more RLIs.

See also [RLI](#)⁶.

logical file name A unique identifier for the contents of a file.

P

physical file name The address or the location of a copy of a file on a storage system.

R

Replica Location Index (RLI) Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents.

⁶ #rli

Index

E

errors, 25

I

installing

from binary, 5

from source, 5

prerequisites, 1

database, 1

ODBC driver, 2

ODBC manager, 2

recommended software, 3

setting up the database, 6

configure settings, 7

create databases, 8

create user account, 8

DSNs, 9

which database?, 7

S

server operations

starting RLS server, 21

stopping RLS server, 22

T

testing

your installation, 23

troubleshooting

for admin, 25

U

updating

from source, 6

usage statistics, 39

GT 4.2.1 RLS : User's Guide

GT 4.2.1 RLS : User's Guide

Introduction

The Replica Location Service (RLS) maintains and provides access to mapping information from *logical names* for data items to *target names*.

RLS was co-developed by the Globus team and Work Package 2 of the DataGrid project. The distributed RLS is intended to replace the centralized Globus replica catalog available in earlier releases of GT2.x. The distributed RLS provides higher performance, reliability and scalability.

Replication of data items can reduce access latency, improve data locality, and increase robustness, scalability and performance for distributed applications. An RLS typically does not operate in isolation but functions as one component of a data grid architecture (other components include services that provide reliable file transfers, metadata management, reliable replication and workflow management).

The RLS implementation is based on the following mechanisms:

- *Consistent local state maintained in Local Replica Catalogs (LRCs)*. Local catalogs maintain mappings between arbitrary *logical file names (LFNs)* and the *physical file names (PFNs)* associated with those LFNs on its storage system(s).
 - *Collective state with relaxed consistency maintained in Replica Location Indices (RLIs)*. Each *RLI* contains a set of mappings from LFNs to LRCs. A variety of index structures can be defined with different performance characteristics simply by varying the number of RLIs and the amount of redundancy and partitioning among the RLIs.
 - *Soft state maintenance of RLI state*. LRCs send information about their state to RLIs using soft state protocols. State information in RLIs times out and must be periodically refreshed by soft state updates.
 - *Compression of state updates*. This optional compression uses *Bloom filters* to summarize the content of a Local Replica Catalog before sending a soft state update to a Replica Location Index Node.
 - *Membership and partitioning information maintenance*. The current RLS implementation maintains static information about the LRCs and RLIs participating in the distributed system. As new implementations of the RLS are developed, they will use OGSA mechanisms for registration of services and for service lifetime management.
-

Table of Contents

1. Mapping Replica Locations	1
1. Generate a valid proxy	1
2. Ping the server	1
3. Creating replica location mappings	1
4. Adding replica location mappings	1
5. Querying replica location mappings	2
6. Deleting replica location mappings	2
7. Using bulk operations	2
8. Using interactive mode	3
2. Command line tools	4
globus-rls-admin	5
globus-rls-cli	7
globus-rls-server	11
3. Troubleshooting	16
1. Verbose error messages	16
2. Errors	17
4. Usage statistics collection by the Globus Alliance	18
1. RLS-specific usage statistics	18
Glossary	20
Index	21

List of Tables

2.1. Options for globus-rls-admin	6
2.2. Options for globus-rls-cli	7
2.3. Commands for globus-rls-cli	9
2.4. Options for globus-rls-server	14
3.1. Replica Locator Service (RLS) Errors	17

Chapter 1. Mapping Replica Locations

This section describes a few key usage scenarios and provides examples of using the RLS command-line tools.

1. Generate a valid proxy

Before using any of the tools, a user must generate a valid user proxy. Use [grid-proxy-init](#).

```
% $GLOBUS_LOCATION/bin/grid-proxy-init
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA.mymachine/OU=mymachine/CN=John Smith
Enter GRID pass phrase for this identity:
```

2. Ping the server

To check whether your server is active you may use the [globus-rls-admin\(1\) ping](#) command.

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -p rls://localhost
ping rls://localhost: 0 seconds
```

3. Creating replica location mappings

When the RLS server is first installed its database of replica location information will be empty, as expected. To create a replica location mapping, use the [globus-rls-cli\(1\) create](#) command. Replica information in RLS is represented as mappings from logical names to target names. Typically, the logical name will be a unique identifier for a given replicated data set and the target name will be a URL identifying a particular replica of the data set.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli create my-logical-name-1 url-for-target-name-1 rls://lo
```



Note

The create command is intended for creating the initial replica mapping entry for a given logical name. If the user attempts to create another entry using an existing logical name, RLS will report a user error. To map additional target names to an existing logical name, see [Section 4, “Adding replica location mappings”](#).

4. Adding replica location mappings

To map additional target names to a logical name created by the previously described create command, use the [globus-rls-cli\(1\) add](#) command.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli add my-logical-name-1 url-for-target-name-2 rls://lo
```

5. Querying replica location mappings

Once your RLS server is populated with replica location mappings, you can query the server for useful information using the `globus-rls-cli(1) query` command.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli query lrc lfn my-logical-name-1 rls://localhost
my-logical-name-1: url-for-target-name-1
my-logical-name-1: url-for-target-name-2
```

6. Deleting replica location mappings

To remove unwanted replica location mappings from your RLS server, use the `globus-rls-cli(1) delete` command. The delete operation works *directly* on the mapping and *indirectly* on the logical and target names. When the delete operation is performed by the RLS server the association between the specified logical name and the specified target name is eliminated. However, there may still be other target names associated with the logical name, and there could still be other logical names associated with the target name, though the latter scenario is less likely. Only when all mapping associations for a given logical name (or a given target name) are eliminated (i.e., the specified logical name has no target names associated with it) will the logical (or target) name be deleted from the RLS server.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli delete my-logical-name-1 url-for-target-name-1 rls://
% $GLOBUS_LOCATION/bin/globus-rls-cli query lrc lfn my-logical-name-1 rls://localhost
my-logical-name-1: url-for-target-name-2
% $GLOBUS_LOCATION/bin/globus-rls-cli delete my-logical-name-1 url-for-target-name-2 rls://
% $GLOBUS_LOCATION/bin/globus-rls-cli query lrc lfn my-logical-name-1 rls://localhost
globus_rls_client: LFN doesn't exist: my-logical-name-1
```

7. Using bulk operations

The `globus-rls-cli(1)` supports a variety of bulk operations that enhance productivity for users and reduce network connection overhead from making multiple, separate invocations of the client. The general pattern for bulk operation support as implemented by the client is a parameter list consisting of `bulk command-name [command-modifiers] param-1 param-2 param-N`, such as `bulk query lrc lfn my-logical-name-1 my-logical-name-2 my-logical-name-3`.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli bulk create my-logical-name-1 url-for-target-name-1-1
% $GLOBUS_LOCATION/bin/globus-rls-cli bulk add my-logical-name-1 url-for-target-name-1-2 m
% $GLOBUS_LOCATION/bin/globus-rls-cli bulk query lrc lfn my-logical-name-1 my-logical-name
my-logical-name-3: LFN doesn't exist
my-logical-name-2: url-for-target-name-2-1
my-logical-name-2: url-for-target-name-2-2
my-logical-name-1: url-for-target-name-1-1
my-logical-name-1: url-for-target-name-1-2
```

8. Using interactive mode

The `globus-rls-cli(1)` supports an interactive mode in addition to the general command-line mode. To enter the interactive mode, simply invoke the client without any command.

```
% $GLOBUS_LOCATION/bin/globus-rls-cli rls://localhost
rls> query lrc lfn my-logical-name-2
my-logical-name-2: url-for-target-name-2-1
my-logical-name-2: url-for-target-name-2-2
rls> query lrc lfn my-logical-name-1
my-logical-name-1: url-for-target-name-1-1
my-logical-name-1: url-for-target-name-1-2
rls> bulk delete my-logical-name-1 url-for-target-name-1-1 my-logical-name-1
url-for-target-name-1-2 my-logical-name-2 url-for-target-name-2-1
my-logical-name-2 url-for-target-name-2-2
rls> bulk query lrc lfn my-logical-name-2 my-logical-name-1
my-logical-name-1: LFN doesn't exist
my-logical-name-2: LFN doesn't exist
rls> exit
```

Chapter 2. Command line tools

Name

globus-rls-admin -- RLS administration tool

globus-rls-admin

Tool description

Performs administrative operations on an RLS server.

Synopsis

-A/-a/-C option value/-c option/-d/-e/-p/-q/-s/-t timeout/-u/-v [rli] [pattern] [server]

Options

Table 2.1. Options for globus-rls-admin

-A	<p>Adds rli to the list of <i>RLI</i> servers updated by an <i>LRC</i> server using <i>Bloom filters</i>.</p> <p><i>Note:</i> Partitions are not supported with Bloom filters. The LRC server maintains one Bloom filter for all <i>LFNs</i> in its database, which is sent to all RLI servers configured to receive Bloom filter updates with this option.</p>
-a	<p>Adds rli and optionally pattern to the list of RLI servers that the LRC server sends updates to (using a list of LFNs).</p> <p>If pattern is specified, then only LFNs matching it will be sent to rli.</p> <p>If rli is added with no patterns, then it is sent all updates. Pattern matching is done using standard Unix file globbing.</p>
-C option value	<p>Sets server option to value.</p> <p><i>Important:</i> This does <i>not</i> update the configuration file. The next time the server is restarted, the configuration change will be <i>lost</i>.</p>
-c option	<p>Retrieves the configuration value for the specified option from the server.</p> <p>If option is set to all, then all options are retrieved.</p>
-d	<p>Removes rli and pattern from the list of RLI servers that the LRC server sends updates to.</p> <p>If pattern is not specified, then all entries for rli are removed.</p> <p><i>Note:</i> If all patterns are removed separately, then rli is sent all updates. To stop any updates from being sent to rli, do <i>not</i> specify pattern.</p>
-e	<p>Clears the LRC database. Removes all lfn, <i>pfn</i> mappings.</p>
-p	<p>Verifies that the server is responding.</p>
-q	<p>Causes the RLS server to exit.</p>
-S	<p>Shows statistics and other information gathered by the RLS server.</p> <p>This is intended to be input into GRIS.</p>
-s	<p>Shows the list of RLI servers and patterns being sent updates by the LRC server.</p> <p>If rli or pattern are not specified, they are considered wildcards.</p>
-t timeout	<p>Sets timeout (in seconds) for RLS server requests.</p> <p>The default value is 30.</p>
-u	<p>Causes the LRC server to immediately start full soft state updates to any RLI servers previously added with the -a option.</p>
-v	<p>Shows the version and exits.</p>

Name

globus-rls-cli -- RLS client tool

globus-rls-cli

Tool description

Provides a command line interface to some of the functions supported by RLS. It also supports an interactive interface (if *command* is not specified). In interactive mode, double quotes may be used to encode an argument that contains white space.

Synopsis

command [*-c*] [*-h*] [*-l reslimit*] [*-s*] [*-t timeout*] [*-u*] [*command*] *rls-server*

Options

The client command tool uses **getopt** for command line parsing.

Note: Some versions will continue scanning for options (works that begin with a hyphen) for the entire command line, which makes it impossible to specify negative integer or floating point value for an *attribute*. The workaround for this problem is to tell **getopt()** that there are no more options by including 2 hyphens. For example, to specify the value **-2** you must enter **-- -2**.

Table 2.2. Options for globus-rls-cli

-c	Sets "clearvalues" flag when deleting an attribute (will remove any attribute value records when an attribute is deleted).
-h	Shows usage.
-l reslimit	<p>Sets an incremental limit on the number of results returned by a wildcard query at a time.</p> <p>Note that <i>all results will be returned</i> by the client. This parameter only limits the number of results <i>incrementally retrieved</i> by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user's perspective the client will simply return all 1000 results.</p> <p>Zero means no limit.</p>
-s	Uses SQL style wildcards (% and _).
-t timeout	Sets timeout (in seconds) for RLS server requests. The default is 30 seconds.
-u	Uses Unix style wildcards (* and ?).
-v	Shows version.

Commands

Table 2.3. Commands for globus-rls-cli

add <lfn> <pfn>	Adds <i>pfn</i> to mappings of <i>lfn</i> in an <i>LRC</i> catalog.
attribute add <object> <attr> <obj-type> <attr-type>	Adds an attribute to an object, where <i>object</i> should be the lfn or pfn name. <i>obj-type</i> should be one of lfn or pfn. <i>attr-type</i> should be one of date, float, int, or string. If <value> is of type date then it should be in the form "YYYY-MM-DD HH:MM:DD".
attribute bulk add <object> <attr> <obj-type>	Bulk adds attribute values.
attribute bulk delete <object> <attr> <obj-type>	Bulk deletes attributes.
attribute bulk query <attr> <obj-type> <object>	Bulk queries attributes.
attribute define <attr> <obj-type> <attr-type>	Defines a new attribute.
attribute delete <object> <attr> <obj-type>	Removes <i>attribute</i> from <i>object</i> .
attribute modify <object> <attr> <obj-type> <attr-type>	Modifies the value of an attribute.
attribute query <object> <attr> <obj-type>	Retrieves the value of the specified attribute for object .
attribute search <attr> <obj-type> <operator> <attr-type>	Searches for objects which have the specified attribute matching <i>operator</i> and <i>value</i> . <i>operator</i> should be one of =, !=, >, >=, <, or <=.
attribute show <attr> <obj-type>	Shows an attribute definition. If <i>attr</i> is a hyphen (-) then all attributes are shown.
attribute undefine <attr> <obj-type>	Deletes an attribute definition. Will return an error if any objects possess this attribute.
bulk add <lfn> <pfn> [<lfn> <pfn>]	Bulk adds lfn, pfn mappings.
bulk create <lfn> <pfn> [<lfn> <pfn>]	Bulk creates lfn, pfn mappings.
bulk delete <lfn> <pfn> [<lfn> <pfn>]	Bulk deletes lfn, pfn mappings.
bulk query lrc lfn [<lfn> ...]	Bulk queries the LRC for lfns.
bulk query lrc pfn [<pfn> ...]	Bulk queries the LRC for pfns.
bulk query rli lfn [<lfn> ...]	Bulk queries the <i>RLI</i> for lfns.
create <lfn> <pfn>	Creates a new <i>lfn</i> , <i>pfn</i> mapping in an LRC catalog.
delete <lfn> <pfn>	Deletes a <i>lfn</i> , <i>pfn</i> mapping from an LRC catalog.
exit	Exits the interactive session.
help	Prints a help message.
query lrc lfn <lfn>	Queries an LRC server for mappings of <i>lfn</i> .
query lrc pfn <pfn>	Queries an LRC server for mappings to <i>pfn</i> .
query rli lfn <lfn>	Queries an RLI server for mappings of <i>lfn</i> .

query wildcard lrc lfn <lfn-pattern>	Performs a wildcarded query of an LRC server for mappings of <i>lfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
query wildcard lrc pfn <pfn-pattern>	Queries an LRC server for mappings to <i>pfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
query wildcard rli lfn <lfn-pattern>	Queries an RLI server for mappings of <i>lfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
set reslimit <limit>	<p>Sets an incremental limit on the number of results returned by a wildcard query at a time.</p> <p>Note that <i>all results will be returned</i> by the client. This parameter only limits the number of results <i>incrementally retrieved</i> by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user's perspective the client will simply return all 1000 results.</p>
set timeout <timeout>	<p>Sets the timeout (in seconds) on calls to the RLS server.</p> <p>The default value is 30.</p>
version	Shows the version and exits.

Name

globus-rls-server -- RLS server tool

globus-rls-server

Tool description

The RLS server (**globus-rls-server**) can be configured as either one or both of the following:

- *Location Replica Catalog (LRC)* server, which manages *Logical FileName (LFN)* to *Physical FileName (PFN)* mappings in a database. *Note*: If **globus-rls-server** is configured as an LRC server, the *RLI* servers that it sends updates to should be added to the database using **globus-rls-admin**.
- *Replica Location Index (RLI)* server, which manages mappings of LFNs to LRC servers.

Clients wishing to locate one or more physical filenames associated with a logical filename should first contact an RLI server, which will return a list of LRCs that may know about the LFN. The LRC servers are then contacted in turn to find the physical filenames.

Note: RLI information may be out of date, so clients should be prepared to get a negative response when contacting an LRC (or no response at all if the LRC server is unavailable).

Synopsis

```
[ -B lrc_update_bf ] [ -b maxbackoff ] [ -C rlscertfile ] [ -c conffile ] [ -d ] [ -e rli_expire_int ] [ -F lrc_update_factor ] [ -f maxfreethreads ] [ -I true/false ] [ -i idletimeout ] [ -K rlskeyfile ] [ -L loglevel ] [ -l true/false ] [ -M maxconnections ] [ -m maxthreads ] [ -N ] [ -o lrc_buffer_time ] [ -p pidfiledir ] [ -r true/false ] [ -S rli_expire_stale ] [ -s starthreads ] [ -t timeout ] [ -U myurl ] [ -u lrc_update_ll ] [ -v ]
```

LRC to RLI Updates

Two methods exist for LRC servers to inform RLI servers of their LFNs.

- By default, the LFNs are sent from the LRC to the RLI. This can be time consuming if the number of LFNs is large, but it does give the RLI an exact list of the LFNs known to the LRC, and it allows wildcard searching of the RLI.
- Alternatively, *Bloom filters* may be sent, which are highly compressed summaries of the LFNs. However, they do not allow wildcard searching and will generate more "false positives" when querying an RLI.

Please see below for more on Bloom filters.

globus-rls-admin can be used to manage the list of RLIs that an LRC server updates. This includes partitioning LFNs among multiple RLI servers.

A soft state algorithm is used in both update modes: periodically the LRC server sends its state (LFN information) to the RLI servers it updates. The RLI servers add these LFNs to their indexes or update timestamps if the LFNs were already known. RLI servers expire information about LFN, LRC mappings if they haven't been updated for a period longer than the soft state update interval.

The following options in the configuration file control the soft state algorithm when an LRC updates an RLI by sending LFNs:

- **rli_expire_int** (seconds)
- **rli_expire_stale** (seconds)
- **lrc_update_ll** (seconds)
- **lrc_update_bf** (seconds)

Updates to an LRC (new LFNs or deleted LFNs) normally don't propagate to RLI servers until the next soft state update (controlled by options **lrc_update_ll** and **lrc_update_bf**).

However, by enabling "immediate update" mode (set **lrc_update_immediate** to **true**), an LRC will send updates to an RLI within **lrc_buffer_time** seconds.

If updates are done with LFN lists then only the LFNs that have been added or deleted to the LRC are sent. If Bloom filters are used, then the entire Bloom filter is sent.

When immediate updates are enabled, the interval between soft state updates is multiplied by **lrc_update_factor** as long as no updates have failed (LRC and RLI are considered to be in sync). This can greatly reduce the number of soft state updates an LRC needs to send to an RLI.

Incremental updates are buffered by the LRC server until either 200 updates have accumulated (when LFN lists are used), or **lrc_buffer_time** seconds have passed since the last update.

Bloom filter updates

A Bloom filter is an array of bits. Each LFN is hashed multiple times and the corresponding bits in the Bloom filter are set.

Querying an RLI to verify if an LFN exists is done by performing the same hashes and checking if the bits in the filter are on. If not, then the LFN is known not to exist. If they're all on, then all that's known is that the LFN probably exists.

The size of the Bloom filter (as a multiple of the number of LFNs) and the number of hash functions control the false positive rate. The default values of 10 and 3 give a false positive rate of approximately 1%.

The advantage of Bloom filters is their efficiency. For example, if the LRC has 1,000,000 LFNs in its database, with an average length of 20 bytes, then 20,000,000 bytes must be sent to an RLI during a soft state update (assuming no partitioning). The RLI server must perform 1,000,000 updates to its database to create new LFN, LRC mappings or update timestamps on existing entries. With Bloom filters only 1,250,000 bytes are sent (10 x 1,000,000 bits / 8), and there are no database operations on the RLI (Bloom filters are maintained entirely in memory). A comparison of the time to perform a 1,000,000 LFN update: it took 20 minutes sending all the LFNs and less than 1 second using a Bloom filter. However as noted before, Bloom filters do *not* support wild card searches of an RLI.

Note: An LRC server can update some RLIs with Bloom filters and others with LFNs. However, an RLI server can only be updated using one method.

The following options in the [Configuration](#) file control Bloom filter updates:

- **rli_bloomfilter** true/false
- **rli_bloomfilter_dir** none/default/pathname
- **lrc_bloomfilter_numhash** N
- **lrc_bloomfilter_ratio** N

- `irc_update_bf` seconds

Log Messages

globus-rls-server uses syslog to log errors and other information (facility **LOG_DAEMON**) when it's running in normal (daemon) mode.

If the **-d** option (debug) is specified, then log messages are written to stdout.

Signals

The server will reread its configuration file if it receives a **HUP** signal. It will wait for all current requests to complete and shut down cleanly if sent any of the following signals: **INT**, **QUIT** or **TERM**.

Options (globus-rls-server)

The following table describes the command line options available for `globus-rls-server`:

Table 2.4. Options for globus-rls-server

-b maxbackoff	Maximum time (in seconds) that globus-rls-server will attempt to reopen the socket it listens on after an I/O error.
-C rls-certfile	Name of the X.509 certificate file that identifies the server; sets environment variable X509_USER_CERT .
-c conffile	Name of the configuration file for the server. The default is \$GLOBUS_LOCATION/etc/globus-rls-server.conf if the environment variable GLOBUS_LOCATION is set; else, /usr/local/etc/globus-rls-server.conf .
-d	Enables debugging. The server will not detach from the controlling terminal, and log messages will be written to stdout rather than syslog. For additional logging verbosity set the loglevel (see the -L option) to higher values.
-e rli_expire_int	Interval (seconds) at which an RLI server should expire stale entries.
-F lrc_update_factor	If lrc_update_immediate mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (lrc_update_ll) is multiplied by lrc_update_factor .
-f maxfreethreads	Maximum number of idle threads the server will leave running. Excess threads are terminated.
-I true false	Turns LRC to RLI immediate update mode on (true) or off (false). The default value is false .
-i idletimeout	Seconds after which idle client connections are timed out.
-K rls-keyfile	Name of the X.509 key file. Sets environment variable X509_USER_KEY .
-L loglevel	Sets the log level. By default this is 0 , which means only errors will be logged. Higher values mean more verbose logging.
-l true false	Configures whether the server is an LRC server. The default is false .
-M maxconnections	Maximum number of active connections. It should be small enough to prevent the server from running out of open file descriptors. The default value is 100 .
-m maxthreads	Maximum number of threads server will start up to support simultaneous requests.
-N	Disables authentication checking. This option is intended for debugging. Clients should use the URL RLSN://host to disable authentication on the client side.
-o lrc_buffer_time	LRC to RLI updates are buffered until either the buffer is full or this much time (in seconds) has elapsed since the last update. The default value is 30 .
-p pidfiledir	Directory where PID files should be written.

-r	Configures whether the server is an RLI server. The default value is false .
-S rli_expire_stale	Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC). Stale entries are not returned in queries.
-s startthreads	Number of threads to start up initially.
-t timeout	Timeout (in seconds) for calls to other RLS servers (in other words, for LRC calls to send an update to an RLI). A value of 0 disables timeouts. The default value is 30 .
-U myurl	URL for this server.
-u lrc_update_ll	Interval (in seconds) between lfn-list LRC to RLI updates.
-v	Shows version and exits.

Chapter 3. Troubleshooting

The following section described a few troubleshooting tips. Additional information on troubleshooting can be found in the [FAQ](#)¹.

For a list of common errors in GT, see [Error Codes](#).

1. Verbose error messages

When troubleshooting problems encountered during usage of the RLS client or server, verbose error messages may be enabled by setting the `GLOBUS_ERROR_VERBOSE` environment variable. Verbose error messages are particularly helpful when communicating on the *rls-user@globus.org* or *gt-user@globus.org* list or when reporting problems on the *bugzilla.globus.org* site.

¹ http://www.globus.org/toolkit/data/rls/rls_faq.html

2. Errors

Table 3.1. Replica Locator Service (RLS) Errors

Error Code	Definition	Possible Solutions
<p>Error with credential: The proxy credential: <credential> with subject: <subject> expired <minutes> minutes ago</p>	<p>Expired proxy credential</p>	<p>Create a new proxy with <code>grid-proxy-init</code>.</p>
<p>Unable to connect to local-host:xxxx</p>	<p>Unable to connect to the local host. This can be due to a variety of reasons, including a wrong address or port number in the RLS connection URL or an issue with a firewall configuration.</p>	<ul style="list-style-type: none"> • Double-check the address and port number in the RLS connection URL. parameters are correct. • If a firewall configuration is preventing connections to the target host for a particular port, you may need to consult the system administrator.
<p>"connection timeout"</p>	<p>At times, a client may experience a connection timeout when interacting with the RLS server due to a variety of reasons:</p> <ul style="list-style-type: none"> • One reason could simply be due to wide-area network latency or congestion. • Another situation that users eventually encounter is due to scaling of the system. As the RLS server's database of replica location mappings grows in size, some query operations, such as bulk queries involving large quantities of mappings or wildcard queries that result in a large subset of mappings, will begin to take more time both to process the query and to return the large results set to the client over the network. 	<p>If timeouts are experienced with increasing frequency, increase the RLS server's timeout configuration parameter found in the <code>\$GLOBUS_LOCATION/var/globus-rls-server.conf</code> file. You may also use the <code>-t</code> timeout option of the <code>globus-rls-cli</code> tool.</p>

Chapter 4. Usage statistics collection by the Globus Alliance

1. RLS-specific usage statistics

The following usage statistics are sent by RLS Server by default in a UDP packet:

- Component identifier
- Usage data format identifier
- Time stamp
- Source IP address
- Source hostname (to differentiate between hosts with identical private IP addresses)
- Version number
- Uptime
- *LRC* service indicator
- *RLI* service indicator
- Number of *LFNs*
- Number of *PFNs*
- Number of Mappings
- Number of RLI LFNs
- Number of RLI LRCs
- Number of RLI Senders
- Number of RLI Mappings
- Number of threads
- Number of connections

The RLS sends the usage statistics at server startup, server shutdown, and once every 24 hours when the service is running.

If you wish to disable this feature, you can set the following environment variable before running the RLS:

```
export GLOBUS_USAGE_OPTOUT=1
```

By default, these usage statistics UDP packets are sent to `usage-stats.globus.org:4180` but can be redirected to another host/port or multiple host/ports with the following environment variable:

```
export GLOBUS_USAGE_TARGETS="myhost.mydomain:12345 myhost2.mydomain:54321"
```

You can also dump the usage stats packets to stderr as they are sent (although most of the content is non-ascii). Use the following environment variable for that:

```
export GLOBUS_USAGE_DEBUG=MESSAGES
```

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ ../../Usage_Stats.html

Glossary

B

Bloom filter Compression scheme used by the Replica Location Service (RLS) that is intended to reduce the size of soft state updates between Local Replica Catalogs (LRCs) and Replica Location Index (RLI) servers. A Bloom filter is a bit map that summarizes the contents of a Local Replica Catalog (LRC). An LRC constructs the bit map by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits.

L

Local Replica Catalog (LRC) Stores mappings between logical names for data items and the target names (often the physical locations) of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. Also may associate attributes with logical or target names. Each LRC periodically sends information about its logical name mappings to one or more RLIs.

See also [RLI](#)⁶.

logical file name A unique identifier for the contents of a file.

logical name A unique identifier for the contents of a data item.

P

physical file name The address or the location of a copy of a file on a storage system.

R

Replica Location Index (RLI) Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents.

RLS attribute Descriptive information that may be associated with a logical or target name mapping registered in a Local Replica Catalog (LRC). Clients can query the LRC to discover logical names or target names that have specified RLS attributes.

T

target name The address or location of a copy of a data item on a storage system.

⁶ #rli

Index

A

- administrative operations
 - configuring LRC server to stop updating RLI, 5
 - configuring LRC-to-RLI updates
 - compressed (Bloom filters), 5
 - uncompressed, 5
 - configuring settings (runtime only), 5
 - pinging the server, 1, 5
 - stopping RLS server, 5

C

- client operations
 - attributes
 - adding an attribute to an lfn or pfn, 7
 - bulk adding attribute values, 7
 - bulk deleting attribute values, 7
 - bulk querying attributes, 7
 - defining a new attribute, 7
 - deleting an attribute definition, 7
 - modifying the value of an attribute, 7
 - removing an attribute, 7
 - retrieving the value of the specified attribute for an lfn or pfn, 7
 - searching for lfns or pfns which have the specified matching operator and value, 7
 - showing an attribute definition, 7
 - basic
 - adding physical filenames to mappings of logical filenames in a LocalReplicaCatalog, 1, 7
 - creating a new lfn, pfn mapping in an LRC catalog, 1, 7
 - deleting a lfn, pfn mapping from an LRC catalog, 2, 7
 - exiting the interactive session, 7
 - bulk, 2
 - bulk adding lfn, pfn mappings, 7
 - bulk creating lfn, pfn mappings, 7
 - bulk deleting lfn, pfn mappings, 7
 - bulk querying the LRC for lfns, 7
 - bulk querying the LRC for pfns, 7
 - bulk querying the RLI for lfns, 7
 - querying, 2
 - performing a wildcarded query of an LRC server for mappings of lfn-pattern, 7
 - querying an LRC server for mappings of lfn, 7
 - querying an LRC server for mappings of pfn, 7
 - querying an LRC server for mappings to pfn-pattern, 7
 - querying an RLI server for mappings of lfn, 7

- querying an RLI server for mappings to lfn-pattern, 7
- using interactive mode, 3, 7

E

- errors, 17

G

- generating a valid proxy, 1

S

- server operations, 11
 - configuring RLS server as Location Replica Catalog (LRC), 11
 - configuring RLS server as Replica Location Index (RLI), 11

T

- troubleshooting, 16
 - (for end-users), 16
 - verbose error messages, 16

U

- usage statistics, 18

GT 4.2.1 RLS : Developer's Guide

GT 4.2.1 RLS : Developer's Guide

Introduction

This guide contains information of interest to developers working with the RLS. It provides reference information for application developers, including APIs, architecture, procedures for using the APIs and code samples.

Table of Contents

1. Before you begin	1
1. Feature summary	1
2. Tested platforms	1
3. Backward compatibility summary	1
4. Technology dependencies	1
5. Replica Location Service (RLS) Security Considerations	2
2. Usage scenarios	3
1. Java Examples	3
2. Example Code	5
3. Tutorials	7
4. Architecture and design overview	8
5. APIs	9
1. Programming Model Overview	9
2. Component API	9
6. Command line tools	10
globus-rls-admin	11
globus-rls-cli	13
globus-rls-server	17
7. Configuring RLS	22
1. Configuration overview	22
2. Server configuration file (globus-rls-server.conf)	22
3. Basic configuration	22
4. Host key and certificate configuration	23
5. Configuring LRC to RLI updates	24
6. Configuring the RLS Server for the WS MDS Index Service	25
7. Configuring the RLS Server for the MDS2 GRIS	26
8. Complete RLS Server settings (globus-rls-server.conf)	26
8. Debugging	32
9. Troubleshooting	33
1. Errors	33
10. Related Documentation	34
Glossary	35
Index	36

List of Tables

6.1. Options for globus-rls-admin	12
6.2. Options for globus-rls-cli	13
6.3. Commands for globus-rls-cli	15
6.4. Options for globus-rls-server	20
7.1. Complete RLS Server settings (globus-rls-server.conf)	27
9.1. Replica Locator Service (RLS) Errors	33

Chapter 1. Before you begin

1. Feature summary

Features New in GT 4.2.1

- None since GT 4.2.0.

Other Supported Features

- Comprehensive C and Java library for replica registration, replica lookup, replica attributes, index queries, and administrative tasks.
- Command line (`globus-rls-cli`) tool for client operations on catalogs and indexes.
- Command line (`globus-rls-admin`) tool for administrative tasks.

Deprecated Features

- None

2. Tested platforms

Tested platforms for RLS include most 32-bit flavors of Linux and UNIX, including RedHat, Debian, CentOS, SUSE, Solaris/Sparc, Solaris/x86, and others.

3. Backward compatibility summary

Protocol changes since GT 4.0.x

- None

API changes since GT 4.0.x

- None

Exception changes since GT 4.0.x

- None

Schema changes since GT 4.0.x

- None

4. Technology dependencies

RLS depends on the following GT components:

- `globus_core`
- `globus_common`

- `globus_io`
- `globus_gssapi_gsi`
- `globus_usage`

RLS depends on the following 3rd party software:

- RDBMS: SQLite*, MySQL, PostgreSQL, or Oracle
- ODBC manager: iODBC, unixODBC
- ODBC driver: SQLite-ODBC*, MyODBC, psqLODBC, or Oracle

* *The RLS comes installed with and configured to use these components.*

5. Replica Location Service (RLS) Security Considerations

Security recommendations include:

- *Dedicated User Account:* It is recommended that users create a dedicated user account for installing and running the RLS service (e.g., `globus` as recommended in the general GT installation instructions). This account may be used to install and run other services from the Globus Toolkit.
- *Key and Certificate:* It is recommended that users do not use their hostkey and hostcert for use by the RLS service. Create a `containerkey` and `containercert` with permissions 400 and 644 respectively and owned by the `globus` user. Change the `rlskeyfile` and `rlscertfile` settings in the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) to reflect the appropriate filenames.
- *LRC and RLI Databases:* Users must ensure security of the RLS data as maintained by their chosen database management system. Appropriate precautions should be made to protect the data and access to the database. Such precautions may include creating a user account specifically for RLS usage, encrypting database users' passwords, etc.
- *RLS Configuration:* It is recommended that the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) be owned by and accessible only by the dedicated user account for RLS (e.g., `globus` account per above recommendations). The file contains the database user account and password used to access the LRC and RLI databases along with important settings which, if tampered with, could adversely affect the RLS service.

Chapter 2. Usage scenarios

1. Java Examples

This section provides examples of a few basic operations using the new Java client API. Here is an outline of the typical steps used to resolve a replica location.

- Establish a connection to the Replica Location Index service
- Construct a list of the logical names to be used in the query
- Query the index service
- Inspect the return list and construct lists of logical names to be used in queries to the Local Replica Catalog services
- Query the catalog services
- Inspect the results returned by the catalog services

1.1. Create a connection source

A connection source is needed in order to establish connections to the RLS. The connection source may be shared and is a thread safe object. The `SimpleRLSConnectionSource` may be directly instantiated by the client or it may be used as a JNDI object and shared by multiple clients (e.g., in a container that supports JNDI). In this example, the client instantiates the connection source.

```
RLSConnectionSource source = null;
try {
    source = new SimpleRLSConnectionSource();
} catch (RLSException e) {
    // handle exception
}
```

1.2. Create a connection

Use the connection source to establish a connection. If the source has defaults you may use its parameterless `connect()` method, otherwise you must supply the connection URL and credentials.

```
RLSConnection connection = null;
try {
    connection = source.connect(url, credential);
} catch (RLSException e) {
    // handle exception
}
```

1.3. Create a simple index query

Query objects are used to represent different types of RLS queries. There are simple queries, batch queries, and attribute searches. This examples uses a simple query object. We begin by querying the RLS index service which tells us which catalog services to query for a given logical name.

```
IndexQuery indexQuery = new SimpleIndexQuery(
    SimpleIndexQuery.queryMappingsByLogicalName,
    "my-logical-name-123",
    null);
```

1.4. Query the index service

RLS index services keep an index of logical names for each catalog service that sends its index to the given index service. By querying an index service, the client can find out which catalog services may have replica locations for a desired logical name.

```
try {
    ReplicaLocationIndex index = connection.index();
    Results results = index.query(indexQuery);
    if (results.getRC() == RLSStatusCode.RLS_SUCCESS) {
        List batch = results.getBatch();
        Iterator i = batch.iterator();
        while (i.hasNext()) {
            IndexMappingResult result = (IndexMappingResult) i.next();
            if (result.getRC() != RLSStatusCode.RLS_SUCCESS)
                continue;
            String logicalName = result.getLogical();
            String catalogURL = result.getCatalog();
            // At this point, the client will need to create
            // a CatalogQuery object for each distinct catalog
            // URL returned in the results. These URLs indicate
            // the catalog services which have replica locations
            // for the given logical name.
        }
    }
} catch (RLSException e) {
    // handle exception
}
```

1.5. Create a simple catalog query

Based on the results of the index query, a client will create catalog queries.

```
CatalogQuery catalogQuery = new SimpleCatalogQuery(
    SimpleCatalogQuery.queryMappingsByLogicalName,
    "my-logical-name-123",
```

```
    null);
```

1.6. Query the catalog service

RLS catalog services keep a catalog of logical names mapped to target names. The target names are typically used to indicate the URL for a data object (e.g., a gsiftp, http, etc. URL).

```
try {
    LocalReplicaCatalog catalog = connection.catalog();
    Results results = catalog.query(catalogQuery);
    if (results.getRC() == RLSStatusCode.RLS_SUCCESS) {
        List batch = results.getBatch();
        Iterator i = batch.iterator();
        while (i.hasNext()) {
            MappingResult result = (MappingResult) i.next();
            if (result.getRC() != RLSStatusCode.RLS_SUCCESS)
                continue;
            String logicalName = result.getLogical();
            String targetName = result.getTarget();
            // At this point, the client has resolved the
            // target name for the given logical name. Keep in
            // mind that in the RLS, a logical name may be
            // mapped to multiple target names.
        }
    }
} catch (RLSException e) {
    // handle exception
}
```

2. Example Code

This section provides examples illustrating the basic usage of the client interfaces supported by the RLS. Using the client API, developers may create client applications that interact with the RLS server to perform replica location operations.

Developing in C

Client applications developed in C must do *both* of the following:

1. Include the client header file at `$GLOBUS_LOCATION/include/globus_rls_client.h`.
2. Link to the client shared library at `$GLOBUS_LOCATION/lib/libglobus_rls_client_gcc32dbgpthr`.

For [C language example code](#)¹, click [here](#)².

Developing in Java

Client applications developed in Java must do *all* of the following:

¹ test.c

² test.c

1. Include the RLS Jar, `$GLOBUS_LOCATION/lib/globus_rls_client.jar`, in the CLASSPATH.
2. Import the RLS Package `org.globus.replica.rls.*`.

For [Java language example code](#)³, click [here](#)⁴. Note that the examples in this section use the older, *deprecated* API.

³ testrls.java

⁴ testrls.java

Chapter 3. Tutorials

There are no tutorials available at this time.

Chapter 4. Architecture and design overview

The Replica Location Service design consists of two components. *Local Replica Catalogs (LRCs)* maintain consistent information about logical-to-physical mappings on a site or storage system. The *Replica Location Indexes (RLIs)* aggregate state information contained in one or more LRCs and build a global, hierarchical distributed index to support discovery of replicas at multiple sites. LRCs send summaries of their state to RLIs using soft state update protocols. The server consists of a multi-threaded front end server and a back-end relational database, such as MySQL or PostgreSQL. The front end server can be configured to act as an LRC server and/or an RLI server. Clients access the server via a simple string-based RPC protocol. The client APIs support C, Java and Python. The APIs contain operations to create and delete mappings, associate *attributes* with mappings, and perform queries.

Detailed information on the architecture and design can be found in [A Framework for Constructing Scalable Replica Location Services](#)¹ and [Performance and Scalability of a Replica Location Service](#)².

¹ <http://www.isi.edu/~annc/papers/chervenakFinalSC2002.pdf>

² <http://www.isi.edu/~annc/papers/chervenakhpdc13.pdf>

Chapter 5. APIs

1. Programming Model Overview

The RLS provides a Client API for C and Java based clients. The RLS Client C API is provided in the form of a library (e.g., .so file). Any installation of RLS will include the shared library as part of the `$GLOBUS_LOCATION/include` and `$GLOBUS_LOCATION/lib` directories. The RLS Client Java API depends on the `commons-logging` and `cog-jglobus` libraries which typically located in the `$GLOBUS_LOCATION/lib/common` folder. The RLS Java Client jar is named `globus_rls_client.jar` and is typically installed in the `$GLOBUS_LOCATION/lib` folder.

2. Component API

- [RLS Client C API Documentation](#)¹
- [RLS Client Java API Documentation](#)²

¹ <http://www.isi.edu/%7Eannc/rls/doc/client/index.html>

² <http://www.isi.edu/~schuler/globus/4.2/doc/index.html>

Chapter 6. Command line tools

Name

globus-rls-admin -- RLS administration tool

globus-rls-admin

Tool description

Performs administrative operations on an RLS server.

Synopsis

-A/-a/-C option value/-c option/-d/-e/-p/-q/-s/-t timeout/-u/-v [rli] [pattern] [server]

Options

Table 6.1. Options for globus-rls-admin

-A	<p>Adds rli to the list of <i>RLI</i> servers updated by an <i>LRC</i> server using <i>Bloom filters</i>.</p> <p><i>Note:</i> Partitions are not supported with Bloom filters. The LRC server maintains one Bloom filter for all <i>LFNs</i> in its database, which is sent to all RLI servers configured to receive Bloom filter updates with this option.</p>
-a	<p>Adds rli and optionally pattern to the list of RLI servers that the LRC server sends updates to (using a list of LFNs).</p> <p>If pattern is specified, then only LFNs matching it will be sent to rli.</p> <p>If rli is added with no patterns, then it is sent all updates. Pattern matching is done using standard Unix file globbing.</p>
-C option value	<p>Sets server option to value.</p> <p><i>Important:</i> This does <i>not</i> update the configuration file. The next time the server is restarted, the configuration change will be <i>lost</i>.</p>
-c option	<p>Retrieves the configuration value for the specified option from the server.</p> <p>If option is set to all, then all options are retrieved.</p>
-d	<p>Removes rli and pattern from the list of RLI servers that the LRC server sends updates to.</p> <p>If pattern is not specified, then all entries for rli are removed.</p> <p><i>Note:</i> If all patterns are removed separately, then rli is sent all updates. To stop any updates from being sent to rli, do <i>not</i> specify pattern.</p>
-e	<p>Clears the LRC database. Removes all lfn, <i>pfn</i> mappings.</p>
-p	<p>Verifies that the server is responding.</p>
-q	<p>Causes the RLS server to exit.</p>
-S	<p>Shows statistics and other information gathered by the RLS server.</p> <p>This is intended to be input into GRIS.</p>
-s	<p>Shows the list of RLI servers and patterns being sent updates by the LRC server.</p> <p>If rli or pattern are not specified, they are considered wildcards.</p>
-t timeout	<p>Sets timeout (in seconds) for RLS server requests.</p> <p>The default value is 30.</p>
-u	<p>Causes the LRC server to immediately start full soft state updates to any RLI servers previously added with the -a option.</p>
-v	<p>Shows the version and exits.</p>

Name

globus-rls-cli -- RLS client tool

globus-rls-cli

Tool description

Provides a command line interface to some of the functions supported by RLS. It also supports an interactive interface (if *command* is not specified). In interactive mode, double quotes may be used to encode an argument that contains white space.

Synopsis

command [*-c*] [*-h*] [*-l reslimit*] [*-s*] [*-t timeout*] [*-u*] [*command*] *rls-server*

Options

The client command tool uses **getopt** for command line parsing.

Note: Some versions will continue scanning for options (works that begin with a hyphen) for the entire command line, which makes it impossible to specify negative integer or floating point value for an *attribute*. The workaround for this problem is to tell **getopt()** that there are no more options by including 2 hyphens. For example, to specify the value **-2** you must enter **-- -2**.

Table 6.2. Options for globus-rls-cli

-c	Sets "clearvalues" flag when deleting an attribute (will remove any attribute value records when an attribute is deleted).
-h	Shows usage.
-l reslimit	Sets an incremental limit on the number of results returned by a wildcard query at a time. Note that <i>all results will be returned</i> by the client. This parameter only limits the number of results <i>incrementally retrieved</i> by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user's perspective the client will simply return all 1000 results. Zero means no limit.
-s	Uses SQL style wildcards (% and _).
-t timeout	Sets timeout (in seconds) for RLS server requests. The default is 30 seconds.
-u	Uses Unix style wildcards (* and ?).
-v	Shows version.

Commands

Table 6.3. Commands for globus-rls-cli

add <lfn> <pfn>	Adds <i>pfn</i> to mappings of <i>lfn</i> in an <i>LRC</i> catalog.
attribute add <object> <attr> <obj-type> <attr-type>	Adds an attribute to an object, where <i>object</i> should be the lfn or pfn name. <i>obj-type</i> should be one of lfn or pfn. <i>attr-type</i> should be one of date, float, int, or string. If <value> is of type date then it should be in the form "YYYY-MM-DD HH:MM:DD".
attribute bulk add <object> <attr> <obj-type>	Bulk adds attribute values.
attribute bulk delete <object> <attr> <obj-type>	Bulk deletes attributes.
attribute bulk query <attr> <obj-type> <object>	Bulk queries attributes.
attribute define <attr> <obj-type> <attr-type>	Defines a new attribute.
attribute delete <object> <attr> <obj-type>	Removes <i>attribute</i> from <i>object</i> .
attribute modify <object> <attr> <obj-type> <attr-type>	Modifies the value of an attribute.
attribute query <object> <attr> <obj-type>	Retrieves the value of the specified attribute for object .
attribute search <attr> <obj-type> <operator> <attr-type>	Searches for objects which have the specified attribute matching <i>operator</i> and <i>value</i> . <i>operator</i> should be one of =, !=, >, >=, <, or <=.
attribute show <attr> <obj-type>	Shows an attribute definition. If <i>attr</i> is a hyphen (-) then all attributes are shown.
attribute undefine <attr> <obj-type>	Deletes an attribute definition. Will return an error if any objects possess this attribute.
bulk add <lfn> <pfn> [<lfn> <pfn>]	Bulk adds lfn, pfn mappings.
bulk create <lfn> <pfn> [<lfn> <pfn>]	Bulk creates lfn, pfn mappings.
bulk delete <lfn> <pfn> [<lfn> <pfn>]	Bulk deletes lfn, pfn mappings.
bulk query lrc lfn [<lfn> ...]	Bulk queries the LRC for lfns.
bulk query lrc pfn [<pfn> ...]	Bulk queries the LRC for pfns.
bulk query rli lfn [<lfn> ...]	Bulk queries the <i>RLI</i> for lfns.
create <lfn> <pfn>	Creates a new <i>lfn</i> , <i>pfn</i> mapping in an LRC catalog.
delete <lfn> <pfn>	Deletes a <i>lfn</i> , <i>pfn</i> mapping from an LRC catalog.
exit	Exits the interactive session.
help	Prints a help message.
query lrc lfn <lfn>	Queries an LRC server for mappings of <i>lfn</i> .
query lrc pfn <pfn>	Queries an LRC server for mappings to <i>pfn</i> .
query rli lfn <lfn>	Queries an RLI server for mappings of <i>lfn</i> .

query wildcard lrc lfn <lfn-pattern>	Performs a wildcarded query of an LRC server for mappings of <i>lfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
query wildcard lrc pfn <pfn-pattern>	Queries an LRC server for mappings to <i>pfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
query wildcard rli lfn <lfn-pattern>	Queries an RLI server for mappings of <i>lfn-pattern</i> . Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.
set reslimit <limit>	<p>Sets an incremental limit on the number of results returned by a wildcard query at a time.</p> <p>Note that <i>all results will be returned</i> by the client. This parameter only limits the number of results <i>incrementally retrieved</i> by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user's perspective the client will simply return all 1000 results.</p>
set timeout <timeout>	<p>Sets the timeout (in seconds) on calls to the RLS server.</p> <p>The default value is 30.</p>
version	Shows the version and exits.

Name

globus-rls-server -- RLS server tool

globus-rls-server

Tool description

The RLS server (**globus-rls-server**) can be configured as either one or both of the following:

- *Location Replica Catalog (LRC)* server, which manages *Logical FileName (LFN)* to *Physical FileName (PFN)* mappings in a database. *Note:* If **globus-rls-server** is configured as an LRC server, the *RLI* servers that it sends updates to should be added to the database using **globus-rls-admin**.
- *Replica Location Index (RLI)* server, which manages mappings of LFNs to LRC servers.

Clients wishing to locate one or more physical filenames associated with a logical filename should first contact an RLI server, which will return a list of LRCs that may know about the LFN. The LRC servers are then contacted in turn to find the physical filenames.

Note: RLI information may be out of date, so clients should be prepared to get a negative response when contacting an LRC (or no response at all if the LRC server is unavailable).

Synopsis

```
[ -B lrc_update_bf ] [ -b maxbackoff ] [ -C rlsconf ] [ -c conffile ] [ -d ] [ -e rli_expire_int ] [ -F lrc_update_factor ] [ -f maxfreethreads ] [ -I true/false ] [ -i idletimeout ] [ -K rlskeyfile ] [ -L loglevel ] [ -l true/false ] [ -M maxconnections ] [ -m maxthreads ] [ -N ] [ -o lrc_buffer_time ] [ -p pidfiledir ] [ -r true/false ] [ -S rli_expire_stale ] [ -s starthreads ] [ -t timeout ] [ -U myurl ] [ -u lrc_update_ll ] [ -v ]
```

LRC to RLI Updates

Two methods exist for LRC servers to inform RLI servers of their LFNs.

- By default, the LFNs are sent from the LRC to the RLI. This can be time consuming if the number of LFNs is large, but it does give the RLI an exact list of the LFNs known to the LRC, and it allows wildcard searching of the RLI.
- Alternatively, *Bloom filters* may be sent, which are highly compressed summaries of the LFNs. However, they do not allow wildcard searching and will generate more "false positives" when querying an RLI.

Please see below for more on Bloom filters.

globus-rls-admin can be used to manage the list of RLIs that an LRC server updates. This includes partitioning LFNs among multiple RLI servers.

A soft state algorithm is used in both update modes: periodically the LRC server sends its state (LFN information) to the RLI servers it updates. The RLI servers add these LFNs to their indexes or update timestamps if the LFNs were already known. RLI servers expire information about LFN, LRC mappings if they haven't been updated for a period longer than the soft state update interval.

The following options in the configuration file control the soft state algorithm when an LRC updates an RLI by sending LFNs:

- **rli_expire_int** (seconds)
- **rli_expire_stale** (seconds)
- **lrc_update_ll** (seconds)
- **lrc_update_bf** (seconds)

Updates to an LRC (new LFNs or deleted LFNs) normally don't propagate to RLI servers until the next soft state update (controlled by options **lrc_update_ll** and **lrc_update_bf**).

However, by enabling "immediate update" mode (set **lrc_update_immediate** to **true**), an LRC will send updates to an RLI within **lrc_buffer_time** seconds.

If updates are done with LFN lists then only the LFNs that have been added or deleted to the LRC are sent. If Bloom filters are used, then the entire Bloom filter is sent.

When immediate updates are enabled, the interval between soft state updates is multiplied by **lrc_update_factor** as long as no updates have failed (LRC and RLI are considered to be in sync). This can greatly reduce the number of soft state updates an LRC needs to send to an RLI.

Incremental updates are buffered by the LRC server until either 200 updates have accumulated (when LFN lists are used), or **lrc_buffer_time** seconds have passed since the last update.

Bloom filter updates

A Bloom filter is an array of bits. Each LFN is hashed multiple times and the corresponding bits in the Bloom filter are set.

Querying an RLI to verify if an LFN exists is done by performing the same hashes and checking if the bits in the filter are on. If not, then the LFN is known not to exist. If they're all on, then all that's known is that the LFN probably exists.

The size of the Bloom filter (as a multiple of the number of LFNs) and the number of hash functions control the false positive rate. The default values of 10 and 3 give a false positive rate of approximately 1%.

The advantage of Bloom filters is their efficiency. For example, if the LRC has 1,000,000 LFNs in its database, with an average length of 20 bytes, then 20,000,000 bytes must be sent to an RLI during a soft state update (assuming no partitioning). The RLI server must perform 1,000,000 updates to its database to create new LFN, LRC mappings or update timestamps on existing entries. With Bloom filters only 1,250,000 bytes are sent (10 x 1,000,000 bits / 8), and there are no database operations on the RLI (Bloom filters are maintained entirely in memory). A comparison of the time to perform a 1,000,000 LFN update: it took 20 minutes sending all the LFNs and less than 1 second using a Bloom filter. However as noted before, Bloom filters do *not* support wild card searches of an RLI.

Note: An LRC server can update some RLIs with Bloom filters and others with LFNs. However, an RLI server can only be updated using one method.

The following options in the [Configuration](#) file control Bloom filter updates:

- **rli_bloomfilter** true/false
- **rli_bloomfilter_dir** none/default/pathname
- **lrc_bloomfilter_numhash** N
- **lrc_bloomfilter_ratio** N

- `irc_update_bf` seconds

Log Messages

globus-rls-server uses syslog to log errors and other information (facility **LOG_DAEMON**) when it's running in normal (daemon) mode.

If the **-d** option (debug) is specified, then log messages are written to stdout.

Signals

The server will reread its configuration file if it receives a **HUP** signal. It will wait for all current requests to complete and shut down cleanly if sent any of the following signals: **INT**, **QUIT** or **TERM**.

Options (globus-rls-server)

The following table describes the command line options available for **globus-rls-server**:

Table 6.4. Options for globus-rls-server

-b maxbackoff	Maximum time (in seconds) that globus-rls-server will attempt to reopen the socket it listens on after an I/O error.
-C rls-certfile	Name of the X.509 certificate file that identifies the server; sets environment variable X509_USER_CERT .
-c conffile	Name of the configuration file for the server. The default is \$GLOBUS_LOCATION/etc/globus-rls-server.conf if the environment variable GLOBUS_LOCATION is set; else, /usr/local/etc/globus-rls-server.conf .
-d	Enables debugging. The server will not detach from the controlling terminal, and log messages will be written to stdout rather than syslog. For additional logging verbosity set the loglevel (see the -L option) to higher values.
-e rli_expire_int	Interval (seconds) at which an RLI server should expire stale entries.
-F lrc_update_factor	If lrc_update_immediate mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (lrc_update_ll) is multiplied by lrc_update_factor .
-f maxfreethreads	Maximum number of idle threads the server will leave running. Excess threads are terminated.
-I true false	Turns LRC to RLI immediate update mode on (true) or off (false). The default value is false .
-i idletimeout	Seconds after which idle client connections are timed out.
-K rls-keyfile	Name of the X.509 key file. Sets environment variable X509_USER_KEY .
-L loglevel	Sets the log level. By default this is 0 , which means only errors will be logged. Higher values mean more verbose logging.
-l true false	Configures whether the server is an LRC server. The default is false .
-M maxconnections	Maximum number of active connections. It should be small enough to prevent the server from running out of open file descriptors. The default value is 100 .
-m maxthreads	Maximum number of threads server will start up to support simultaneous requests.
-N	Disables authentication checking. This option is intended for debugging. Clients should use the URL RLSN://host to disable authentication on the client side.
-o lrc_buffer_time	LRC to RLI updates are buffered until either the buffer is full or this much time (in seconds) has elapsed since the last update. The default value is 30 .
-p pidfiledir	Directory where PID files should be written.

-r	Configures whether the server is an RLI server. The default value is false .
-S rli_expire_stale	Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC). Stale entries are not returned in queries.
-s startthreads	Number of threads to start up initially.
-t timeout	Timeout (in seconds) for calls to other RLS servers (in other words, for LRC calls to send an update to an RLI). A value of 0 disables timeouts. The default value is 30 .
-U myurl	URL for this server.
-u lrc_update_ll	Interval (in seconds) between lfn-list LRC to RLI updates.
-v	Shows version and exits.

Chapter 7. Configuring RLS

1. Configuration overview

RLS configuration involves statically-defined, system settings as defined in the RLS configuration file (see `$GLOBUS_LOCATION/etc/globus-rls-server.conf`), settings changed temporarily at run-time using the RLS Admin tool (see `globus-rls-admin(1) -C option value` command), and finally LRC-to-RLI and RLI-to-RLI updates configured using the RLS Admin tool (see `globus-rls-admin(1) -a, -A, -d` commands).

2. Server configuration file (`globus-rls-server.conf`)

Configuration settings for the RLS are specified in the `globus-rls-server.conf` file. If the configuration file is not specified on the command line (see the `-c` option) then it is looked for in both:

- `$GLOBUS_LOCATION/etc/globus-rls-server.conf`
- `/usr/local/etc/globus-rls-server.conf` if `GLOBUS_LOCATION` is not set



Note

Command line options always override items found in the configuration file.

The configuration file is a sequence of lines consisting of a keyword, whitespace, and a value. Comments begin with `#` and end with a newline.

3. Basic configuration

Review the server configuration file `$GLOBUS_LOCATION/etc/globus-rls-server.conf` and change any options you want. The server man page `globus-rls-server(8)` has complete details on all options. The complete details are also provided later in this section.

A minimal configuration file for both an LRC and RLI server would be:

```
# Configure the database connection info
db_user      dbuser
db_pwd       dbpassword

# If the server is an LRC server
lrc_server   true
lrc_dbname   lrc1000

# If the server is an RLI server
rli_server   true
rli_dbname   rli1000 # Not needed if updated by Bloom filters

# Configure who can make requests of the server
acl .*: all
```

```
# RE matching grid-mapfile users or DNs from x509 certs
...
```

4. Host key and certificate configuration

The server uses a host certificate to identify itself to clients. By default this certificate is located in the files `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem`. Host certificates have a distinguished name of the form `/CN=host/FQDN`. If the host you plan to run the RLS server on does not have a host certificate, you must obtain one from your Certificate Authority. The RLS server must be run as the same user who owns the host certificate files (typically root). The location of the host certificate files may be specified in `$GLOBUS_LOCATION/etc/globus-rls-server.conf`:

```
rlscertfile      path-to-cert-file    # default /etc/grid-security/hostcert.pem
rlskeyfile       path-to-key-file   # default /etc/grid-security/hostkey.pem
```

It is possible to run the RLS server without authentication, by starting it with the `-N` option, and using URL's of the form `rlsn://server` to connect to it. Notice that the URL scheme is `rlsn` as opposed to `rls`.

It is generally recommended to run the server with a user account other than root for added security. In order to do so, you will need to create complimentary key and certificate files owned by a designated user account, `globus` for instance.

1. Begin by copying the `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem` to `/etc/grid-security/containercert.pem` and `/etc/grid-security/containerkey.pem`. Note that we use the prefix "container" to conform with the recommended naming scheme for other services distributed with the Globus Toolkit.

```
% cp /etc/grid-security/hostcert.pem /etc/grid-security/containercert.pem
% cp /etc/grid-security/hostkey.pem /etc/grid-security/containerkey.pem
```

2. Then change ownership of the files to the designated user account, `globus` in our example.

```
% chown globus /etc/grid-security/containercert.pem
% chown globus /etc/grid-security/containerkey.pem
```

3. Change the `rlskeyfile` and `rlscertfile` settings in the RLS configuration file (`$GLOBUS_LOCATION/etc/globus-rls-server.conf`) to reflect the appropriate filenames.

```
rlscertfile      /etc/grid-security/containercert.pem
rlskeyfile       /etc/grid-security/containerkey.pem
```

4. Finally, bear in mind that your certificate and key files must always have file permissions 644 and 400 respectively.

```
% ls -l /etc/grid-security/*.pem
-rw-r--r--  1 globus  gridstaff    818 Dec  8  2005 /etc/grid-security/containercer
-r-----   1 globus  gridstaff    887 Dec  8  2005 /etc/grid-security/containerkey
-rw-r--r--  1 root    root        818 Dec  8  2005 /etc/grid-security/hostcert.pem
-r-----   1 root    root        887 Dec  8  2005 /etc/grid-security/hostkey.pem
```

If authentication is enabled, RLI servers must include `acl` configuration options that match the identities of LRC servers that update it and that grant the `rli_update` permission to the LRCs.

5. Configuring LRC to RLI updates

One of the key benefits to using the RLS for managing replica location information is its distributed architecture. In a distributed deployment, one or more Local Replica Catalog (LRC) services will send updates of its contents to one or more Replica Location Index (RLI) services.

By default the installed LRC is not configured to send updates to any RLI, even the local RLI co-located with the local LRC. Use the [globus-rls-admin\(1\)](#) tool to configure the LRC to send updates to one or more RLI services.

- To configure the LRC to send uncompressed lists of its logical names to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -a rls://rli_host rls://lrc_host
```

- To configure the LRC to send compressed bitmaps (using Bloom filters) of its logical names to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -A rls://rli_host rls://lrc_host
```

- To configure the LRC to stop sending updates to a RLI, use the following command:

```
% $GLOBUS_LOCATION/bin/globus-rls-admin -d rls://rli_host rls://lrc_host
```



Note

While any given LRC is capable of sending uncompressed or compressed updates to any RLI. The RLI service must be configured to accept either uncompressed or compressed updates but not both. See the `rli_bloom-filter` setting of the RLS configuration file for more details.

There are tradeoffs between using uncompressed and compressed updates in your configuration. The advantage of using compressed updates, not surprisingly, is a significant reduction in network overhead and memory usage. As replica location mappings grow into the 10's of millions or more, the savings of using compressed updates becomes important. On the other hand, due to the compressed nature of the Bloom filter bitmap used to represent the logical names in the LRC, the wildcard query at the RLI cannot be supported when update compression is used.

6. Configuring the RLS Server for the WS MDS Index Service

The server package includes a script `$GLOBUS_LOCATION/libexec/aggrexec/globus-rls-aggregator-source.pl` that may be used as an Execution Aggregator Source by WS MDS. See [GT 4.2.1 Index Services](#) for more information on setting up and using the Execution Aggregator Source scripts in WS MDS. The script may be invoked as follows and will generate output in the format as depicted.

```
% $GLOBUS_LOCATION/libexec/aggrexec/globus-rls-aggregator-source.pl rls://mysite
<?xml version="1.0" encoding="UTF-8"?>
<rlsStats>
  <site>rls://mysite</site>
  <version>4.0</version>
  <uptime>03:08:15</uptime>
  <serviceList>
    <service>lrc</service>
    <service>rli</service>
  </serviceList>
  <lrc>
    <updateMethodList>
      <updateMethod>lfnlist</updateMethod>
      <updateMethod>bloomfilter</updateMethod>
    </updateMethodList>
    <updatesList>
      <updates>
        <site>rls://myothersite:39281</site>
        <method>bloomfilter</method>
        <date>08/01/05</date>
        <time>16:16:38</time>
      </updates>
    </updatesList>
    <numlfn>283902</numlfn>
    <numpfn>593022</numpfn>
    <nummap>593022</nummap>
  </lrc>
  <rli>
    <updatedViaList>
      <updatedVia>bloomfilters</updatedVia>
    </updatedViaList>
    <updatedByList>
      <updatedBy>
        <site>rls://myothersite:39281</site>
        <date>08/01/05</date>
        <time>10:03:21</time>
      </updatedBy>
    </updatedByList>
  </rli>
</rlsStats>
```

Important

Be sure to configure the security context of the container running the MDS, and be sure that the security configuration on the RLS host recognizes the MDS security context.

When following the instructions provided by the [GT 4.2.1 Index Services](#), you will need to consider the security context used by the MDS to invoke the Execution Aggregator Source script provided by RLS. Most deployments of RLS run the service with security enabled. Therefore any client connections, including administrative status operations, require authentication and authorization. In order for MDS to use the provided script to check RLS status, it must invoke the script with a valid user proxy or user certificate and key. The RLS must recognize the DN from the user certificate (i.e., the DN should be in the gridmap file).

One way to configure the MDS security context for use with RLS monitoring is to set the environment variables `X509_USER_CERT` and `X509_USER_KEY` to point to the container certificate and key. Run the MDS with these environment settings. Also, add the DN from the container certificate to the gridmap file on the host running the RLS.

Alternatively, you could modify the provided script so that it sets the environment variables to another user certificate and key (or proxy) as desired before calling the RLS.

7. Configuring the RLS Server for the MDS2 GRIS

The server package includes a program called `globus-rls-reporter` that will report information about an RLS server to the MDS2 GRIS. Use this procedure to enable this program:

1. To enable Index Service reporting, add the contents of the file `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf` to the MDS2 GRIS configuration file `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf`.
2. If necessary, set your virtual organization (VO) name in `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf`. The default value is `local`. The VO name is referenced twice, on the lines beginning `dn:` and `args:`.
3. You must restart your MDS (GRIS) server after modifying `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf`. You can use the following commands to do so:

```
$GLOBUS_LOCATION/sbin/SXXgris stop
$GLOBUS_LOCATION/sbin/SXXgris start
```

8. Complete RLS Server settings (globus-rls-server.conf)

This section describes the complete details of the RLS Server configuration settings.

Table 7.1. Complete RLS Server settings (globus-rls-server.conf)

<pre>acl user: permission [permission]</pre>	<p>acl entries may be a combination of DNs and local usernames. If a DN is not found in the gridmap file then the file is used to search the acl list.</p> <p>A gridmap file may also be used to map DNs to local usernames, which in turn are matched against the regular expressions in the acl list to determine the user's permissions.</p> <p>user is a regular expression matching distinguished names (or local usernames if a gridmap file is used) of users allowed to make calls to the server.</p> <p>There may be multiple acl entries, with the first match found used to determine a user's privileges.</p> <p>[permission] is one or more of the following values:</p> <ul style="list-style-type: none"> • lrc_read Allows client to read an <i>LRC</i>. • lrc_update Allows client to update an LRC. • rli_read Allows client to read an <i>RLI</i>. • rli_update Allows client to update an RLI. • admin Allows client to update an LRC's list of RLIs to send updates to. • stats Allows client to read performance statistics. • all Allows client to do all of the above.
<pre>authentication true false</pre>	<p>Enable or disable GSI authentication.</p> <p>The default value is true.</p> <p>If authentication is enabled (true), clients should use the URL schema rls: to connect to the server.</p> <p>If authentication is <i>not</i> enabled (false), clients should use the URL schema rlsn:.</p>
<pre>db_pwd password</pre>	<p>Password to use to connect to the database server.</p> <p>The default value is changethis.</p>
<pre>db_user data-baseuser</pre>	<p>Username to use to connect to database server.</p> <p>The default value is dbperson.</p>
<pre>idletimeout seconds</pre>	<p>Seconds after which idle connections close.</p> <p>The default value is 900.</p>
<pre>loglevel N</pre>	<p>Sets loglevel to N (default is 0). Higher levels mean more verbosity.</p>

<code>lrc_bloomfilter_numhash N</code>	<p>Number of hash functions to use in <i>Bloom filters</i>.</p> <p>The default value is 3.</p> <p>Possible values are 1 through 8.</p> <p>This value, in conjunction with <code>lrc_bloomfilter_ratio</code>, will determine the number of false positives that may be expected when querying an RLI that is updated via Bloom filters.</p> <p><i>Note:</i> The default values of 3 and 10 give a false positive rate of approximately 1%.</p>
<code>lrc_bloomfilter_ratio N</code>	<p>Sets ratio of bloom filter size (in bits) to number of <i>LFNs</i> in the LRC catalog (in other words, size of the Bloom filter as a multiple of the number of LFNs in the LRC database.) This is only meaningful if Bloom filters are used to update an RLI. Too small a value will generate too many false positives, while too large a value wastes memory and network bandwidth.</p> <p>The default value is 10.</p> <p><i>Note:</i> The default values of 3 and 10 give a false positive rate of approximately 1%.</p>
<code>lrc_buffer_time N</code>	<p>LRC to RLI updates are buffered until either the buffer is full or this much time in seconds has elapsed since the last update.</p> <p>The default value is 30.</p>
<code>lrc_dbname</code>	<p>Name of LRC database.</p> <p>The default value is <code>lrcdb</code>.</p>
<code>lrc_server true false</code>	<p>If LRC server, the value should be <code>true</code>.</p> <p>The default value is <code>false</code>.</p>
<code>lrc_update_bf seconds</code>	<p>Interval in seconds between LRC to RLI updates when the RLI is updated by Bloom filters. In other words, how often an LRC server does a Bloom filter soft state update.</p> <p>This can be much smaller than the interval between updates without using Bloom filters (<code>lrc_update_ll</code>).</p> <p>The default value is 300.</p>
<code>lrc_update_factor N</code>	<p>If <code>lrc_update_immediate</code> mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (<code>lrc_update_ll</code>) is multiplied by the value of this option.</p>
<code>lrc_update_immediate true false</code>	<p>Turns LRC to RLI immediate mode updates on (<code>true</code>) or off (<code>false</code>).</p> <p>The default value is <code>false</code>.</p>
<code>lrc_update_ll seconds</code>	<p>Number of seconds before an LRC server does an LFN list soft state update.</p> <p>The default value is 86400.</p>

<code>lrc_update_retry seconds</code>	Seconds to wait before an LRC server will retry to connect to an RLI server that it needs to update. The default value is 300.
<code>maxbackoff seconds</code>	Maximum seconds to wait before re-trying listen in the event of an I/O error. The default value is 300 .
<code>maxfreethreads N</code>	Maximum number of idle threads. Excess threads are killed. The default value is 5 .
<code>maxconnections N</code>	Maximum number of simultaneous connections. The default value is 100.
<code>maxthreads N</code>	Maximum number of threads running at one time. The default value is 30.
<code>myurl URL</code>	URL of server. The default value is <code>rls://<hostname>:port</code>
<code>odbcini filename</code>	Sets environment variable ODBCINI. If not specified, and ODBCINI is not already set, then the default value is <code>\$GLOBUS_LOCATION/var/odbc.ini</code> .
<code>pidfile filename</code>	Filename where pid file should be written. The default value is <code>\$GLOBUS_LOCATION/var/<programname>.pid</code> .
<code>port N</code>	Port the server listens on. The default value is 39281 .
<code>result_limit limit</code>	Sets the maximum number of results returned by a query. The default value is 0 (zero), which means no limit. If a query request includes a limit greater than this value, an error (GLOBUS_RLS_BADARG) is returned. If the query request has no limit specified, then at most <code>result_limit</code> records are returned by a query.
<code>rli_bloomfilter true false</code>	RLI servers must have this set to accept Bloom filter updates. If <code>true</code> , then only Bloom filter updates are accepted from LRCs. If <code>false</code> , full LFN lists are accepted. <i>Note:</i> If Bloom filters are enabled, then the RLI does <i>not</i> support wildcarded queries.

<p><code>rli_bloomfilter_dir</code> <code>none</code> <code>default</code> <code>path-</code> <code>name</code></p>	<p>If an RLI is configured to accept bloom filters (<code>rli_bloomfilter true</code>), then Bloom filters may be saved to this directory after updates.</p> <p>This directory is scanned when an RLI server starts up and is used to initialize Bloom filters for each LRC that updated the RLI.</p> <p>This option is useful when you want the RLI to recover its data immediately after a restart rather than wait for LRCs to send another update.</p> <p>If the LRCs are updating frequently, this option is unnecessary and may be wasteful in that each Bloom filter is written to disk after each update.</p> <ul style="list-style-type: none"> • <code>none</code> Bloom filters are not saved to disk. This is the default. • <code>default</code> Bloom filters are saved to the default directory: <ul style="list-style-type: none"> • <code>\$GLOBUS_LOCATION/var/rls-bloomfilters</code> if <code>GLOBUS_LOCATION</code> is set • <code>else, /tmp/rls-bloomfilters</code> • <code>pathname</code> Bloom filters are saved to the named directory. Any other string is used as the directory name unchanged. The Bloom filter files in this directory have the name of the URL of the LRC that sent the Bloom filter, with slashes(/) changed to percent signs (%) and ".bf" appended.
<p><code>rli_dbname</code> <code>database</code></p>	<p>Name of the RLI database.</p> <p>The default value is <code>rliadb</code>.</p>
<p><code>rli_expire_int</code> <code>seconds</code></p>	<p>Interval (in seconds) between RLI expirations of stale entries. In other words, how often an RLI server will check for stale entries in its database.</p> <p>The default value is <code>28800</code>.</p>
<p><code>rli_expire_stale</code> <code>seconds</code></p>	<p>Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC).</p> <p>The default value is <code>86400</code>.</p> <p>This value should be no smaller than <code>lrc_update_ll</code>.</p> <p>Stale RLI entries are not returned in queries.</p> <p><i>Note:</i> If the LRC server is responding, this value is not used. Instead the value of <code>lrc_update_ll</code> or <code>lrc_update_bf</code> is retrieved from the LRC server, multiplied by 1.2, and used as the value for this option.</p>

<code>rli_server</code> <code>true false</code>	If an RLI server, the value should be <code>true</code> . The default value is <code>false</code> .
<code>rlscertfile filename</code>	Name of the X.509 certificate file identifying the server. This value is set by setting environment variable <code>X509_USER_CERT</code> .
<code>rlskeyfile filename</code>	Name of the X.509 key file for the server. This value is set by setting environment variable <code>X509_USER_KEY</code> .
<code>startthreads N</code>	Number of threads to start initially. The default value is 3.
<code>timeout seconds</code>	Timeout (in seconds) for calls to other RLS servers (e.g., for LRC calls to send an update to an RLI).

Chapter 8. Debugging

To run the RLS server in debug mode, use the `-d` option along with the `-L num` option (e.g., `$GLOBUS_LOCATION/bin/globus-rls-server -d -L 3`). The `-d` option instructs the RLS server to direct log output to `stdout`, while the `-L num` option sets the log level where a higher `num` results in more detailed output.

Chapter 9. Troubleshooting

Information on troubleshooting can be found in the [FAQ](#)¹. For a list of common errors in GT, see [Error Codes](#).

1. Errors

Table 9.1. Replica Locator Service (RLS) Errors

Error Code	Definition	Possible Solutions
Error with credential: The proxy credential: <credential> with subject: <subject> expired <minutes> minutes ago	Expired proxy credential	Create a new proxy with grid-proxy-init .
Unable to connect to localhost:xxxx	Unable to connect to the local host. This can be due to a variety of reasons, including a wrong address or port number in the RLS connection URL or an issue with a firewall configuration.	<ul style="list-style-type: none">• Double-check the address and port number in the RLS connection URL. parameters are correct.• If a firewall configuration is preventing connections to the target host for a particular port, you may need to consult the system administrator.
"connection timeout"	<p>At times, a client may experience a connection timeout when interacting with the RLS server due to a variety of reasons:</p> <ul style="list-style-type: none">• One reason could simply be due to wide-area network latency or congestion.• Another situation that users eventually encounter is due to scaling of the system. As the RLS server's database of replica location mappings grows in size, some query operations, such as bulk queries involving large quantities of mappings or wildcard queries that result in a large subset of mappings, will begin to take more time both to process the query and to return the large results set to the client over the network.	If timeouts are experienced with increasing frequency, increase the RLS server's timeout configuration parameter found in the <code>\$GLOBUS_LOCATION/var/globus-rls-server.conf</code> file. You may also use the <code>-t</code> timeout option of the globus-rls-cli tool.

¹ http://www.globus.org/toolkit/data/rls/rls_faq.html

Chapter 10. Related Documentation

For additional details, see the [RPC Protocol Description](#)¹.

¹ [rpcprotocol.pdf](#)

Glossary

B

Bloom filter Compression scheme used by the Replica Location Service (RLS) that is intended to reduce the size of soft state updates between Local Replica Catalogs (LRCs) and Replica Location Index (RLI) servers. A Bloom filter is a bit map that summarizes the contents of a Local Replica Catalog (LRC). An LRC constructs the bit map by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits.

L

Local Replica Catalog (LRC) Stores mappings between logical names for data items and the target names (often the physical locations) of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. Also may associate attributes with logical or target names. Each LRC periodically sends information about its logical name mappings to one or more RLIs.

See also [RLI](#)⁶.

logical file name A unique identifier for the contents of a file.

P

physical file name The address or the location of a copy of a file on a storage system.

R

Replica Location Index (RLI) Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents.

RLS attribute Descriptive information that may be associated with a logical or target name mapping registered in a Local Replica Catalog (LRC). Clients can query the LRC to discover logical names or target names that have specified RLS attributes.

⁶ #rli

Index

A

- administrative operations
 - configuring LRC server to stop updating RLI, 11
 - configuring LRC-to-RLI updates
 - compressed (Bloom filters), 11
 - uncompressed, 11
 - configuring settings (runtime only), 11
 - pinging the server, 11
 - stopping RLS server, 11
- architecture
 - for admin, 8

C

- client operations
 - attributes
 - adding an attribute to an lfn or pfn, 13
 - bulk adding attribute values, 13
 - bulk deleting attribute values, 13
 - bulk querying attributes, 13
 - defining a new attribute, 13
 - deleting an attribute definition, 13
 - modifying the value of an attribute, 13
 - removing an attribute, 13
 - retrieving the value of the specified attribute for an lfn or pfn, 13
 - searching for lfns or pfns which have the specified matching operator and value, 13
 - showing an attribute definition, 13
 - basic
 - adding physical filenames to mappings of logical filenames in a LocalReplicaCatalog, 13
 - creating a new lfn, pfn mapping in an LRC catalog, 13
 - deleting a lfn, pfn mapping from an LRC catalog, 13
 - exiting the interactive session, 13
 - bulk
 - bulk adding lfn, pfn mappings, 13
 - bulk creating lfn, pfn mappings, 13
 - bulk deleting lfn, pfn mappings, 13
 - bulk querying the LRC for lfns, 13
 - bulk querying the LRC for pfns, 13
 - bulk querying the RLI for lfns, 13
 - querying
 - performing a wildcarded query of an LRC server for mappings of lfn-pattern, 13
 - querying an LRC server for mappings of lfn, 13
 - querying an LRC server for mappings of pfn, 13

- querying an LRC server for mappings to pfn-pattern, 13
- querying an RLI server for mappings of lfn, 13
- querying an RLI server for mappings to lfn-pattern, 13
- using interactive mode, 13

D

- debugging, 32

E

- errors, 33

S

- server operations, 17
 - configuring RLS server as Location Replica Catalog (LRC), 17
 - configuring RLS server as Replica Location Index (RLI), 17

GT 4.2.1 Migrating Guide for RLS

Table of Contents

1. Migrating RLS from GT4.0	1
2. Migrating RLS from GT3	1
3. Migrating RLS from GT2	1
Glossary	1

<titleabbrev>Migrating Guide</titleabbrev>

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

1. Migrating RLS from GT4.0

Users of the GT4 RLS may reuse their existing databases with the GT4.2 RLS. No migration is required.

2. Migrating RLS from GT3

Users of the GT3 RLS may reuse their existing databases with the GT4 RLS. No migration is required.

3. Migrating RLS from GT2

Users of GT2 RLS must create the *RLI* database, which was not required in some GT2 versions of RLS. To create the RLI database run the RLI database script for your particular DBMS. See [database configuration instructions](#) for more details. This is only required if the RLS is deployed as an RLI (e.g., `rls_server true`).

Glossary

R

Replica Location Index (RLI)	Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents.
------------------------------	--

GT 4.2.1 RLS: Quality Profile

Table of Contents

1. Test coverage reports	1
2. Code analysis reports	1
3. Outstanding bugs	1
4. Bug Fixes	2
5. Performance reports	2

<titleabbrev>Quality Profile</titleabbrev>

1. Test coverage reports

- No reports are available at this time.

2. Code analysis reports

- No reports are available at this time.

3. Outstanding bugs

- *Threading/Libc Problems*: set `LD_ASSUME_KERNEL=2.2.5` in your environment and see [Platform Notes](#) for more information.
- [Bug 3656](#)¹: ACLs cannot be modified dynamically
- [Bug 4141](#)²: regex call in `auth.c`'s `auth_getperms`
- [Bug 4142](#)³: `globus-rls-admin -s` always indicates RLI does not exist
- [Bug 4512](#)⁴: RLS query returns incomplete result on 64bit system (*patch available*)
- [Bug 6085](#)⁵: RLS server crash with `GLOBUSTHREAD: pthread_mutex_lock()` failed
- [Bug 6239](#)⁶: Wildcard queries with underscores don't work with SQLite
- [Bug 6322](#)⁷: RLS crash on a Sparc/Solaris 10 box (possible duplicate of bug 6356)
- [Bug 6356](#)⁸: RLS crash on x86/Solaris 10 box (*patch available*)

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3656

² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4141

³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4142

⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4512

⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6085

⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6239

⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6322

⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6356

4. Bug Fixes

- [Bug 5999](#)⁹: [WS-RLS] UndefineAttributeClient fails to set values for non-nullable parameters
- [Bug 6009](#)¹⁰: Failure to build on FreeBSD
- [Bug 6100](#)¹¹: some bulk operations on attributes are no-ops
- [Bug 6103](#)¹²: RLS server crashes on 64-bit FC6
- [Bug 6104](#)¹³: getmethodargs does not protect against corrupt method or args

5. Performance reports

- [Performance and Scalability of a Replica Location Service](#)¹⁴

⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5999

¹⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6009

¹¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6100

¹² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6103

¹³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6104

¹⁴ <http://www.globus.org/alliance/publications/papers/chervenakhpdc13.pdf>

GT 4.2.1 Release Notes: RLS

Table of Contents

1. Component Overview	1
2. Feature summary	1
3. Summary of Changes in RLS	1
4. Bug Fixes	2
5. Known Problems	2
6. Technology dependencies	3
7. Tested platforms	3
8. Backward compatibility summary	3
9. Associated Standards	3
10. For More Information	4
Glossary	4

<titleabbrev>Release Note</titleabbrev>

1. Component Overview

The Replica Location Service (RLS) is a standalone server (i.e., it is not deployed in a Web services container) that provides for the registration and lookup of replica information. Within the RLS, there are two types of services, a catalog service and an index service.

2. Feature summary

Features New in GT 4.2.1

- None since GT 4.2.0.

Other Supported Features

- Comprehensive C and Java library for replica registration, replica lookup, replica attributes, index queries, and administrative tasks.
- Command line (`globus-rls-cli`) tool for client operations on catalogs and indexes.
- Command line (`globus-rls-admin`) tool for administrative tasks.

Deprecated Features

- None

3. Summary of Changes in RLS

- No changes since GT 4.2.0.

4. Bug Fixes

- [Bug 5999](#)¹: [WS-RLS] UndefineAttributeClient fails to set values for non-nullable parameters
- [Bug 6009](#)²: Failure to build on FreeBSD
- [Bug 6100](#)³: some bulk operations on attributes are no-ops
- [Bug 6103](#)⁴: RLS server crashes on 64-bit FC6
- [Bug 6104](#)⁵: getmethodargs does not protect against corrupt method or args

5. Known Problems

The following problems and limitations are known to exist for RLS at the time of the 4.2.1 release:

5.1. Limitations

- *Threading/Libc Problems*: set `LD_ASSUME_KERNEL=2.2.5` in your environment and see [Section 2, “Debian”](#) for more information.

5.2. Outstanding bugs

- *Threading/Libc Problems*: set `LD_ASSUME_KERNEL=2.2.5` in your environment and see [Platform Notes](#) for more information.
- [Bug 3656](#)⁶: ACLs cannot be modified dynamically
- [Bug 4141](#)⁷: regex call in auth.c's auth_getperms
- [Bug 4142](#)⁸: globus-rls-admin -s always indicates RLI does not exist
- [Bug 4512](#)⁹: RLS query returns incomplete result on 64bit system (*patch available*)
- [Bug 6085](#)¹⁰: RLS server crash with GLOBUSTHREAD: pthread_mutex_lock() failed
- [Bug 6239](#)¹¹: Wildcard queries with underscores don't work with SQLite
- [Bug 6322](#)¹²: RLS crash on a Sparc/Solaris 10 box (possible duplicate of bug 6356)
- [Bug 6356](#)¹³: RLS crash on x86/Solaris 10 box (*patch available*)

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=5999

² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6009

³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6100

⁴ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6103

⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6104

⁶ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3656

⁷ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4141

⁸ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4142

⁹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4512

¹⁰ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6085

¹¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6239

¹² http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6322

¹³ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6356

6. Technology dependencies

RLS depends on the following GT components:

- globus_core
- globus_common
- globus_io
- globus_gssapi_gsi
- globus_usage

RLS depends on the following 3rd party software:

- RDBMS: SQLite*, MySQL, PostgreSQL, or Oracle
- ODBC manager: iODBC, unixODBC
- ODBC driver: SQLite-ODBC*, MyODBC, psqLODBC, or Oracle

**The RLS comes installed with and configured to use these components.*

7. Tested platforms

Tested platforms for RLS include most 32-bit flavors of Linux and UNIX, including RedHat, Debian, CentOS, SUSE, Solaris/Sparc, Solaris/x86, and others.

8. Backward compatibility summary

Protocol changes since GT 4.0.x

- None

API changes since GT 4.0.x

- None

Exception changes since GT 4.0.x

- None

Schema changes since GT 4.0.x

- None

9. Associated Standards

Associated standards for RLS:

- The RLS is implemented as a conventional service and, as such, does not conform to the WSRF or other WS set of specifications.

10. For More Information

See [Replica Location Service \(RLS\)](#) for more information about this component.

Glossary