GT4 Admin Guide

GT4 Admin Guide

Published April 2005

Table of Contents

List of Tables

Chapter 1. Introduction

This guide is the starting point for everyone who wants to install Globus Toolkit 4.0. It will take you through a basic installation that installs Java WS Core and the following Base Services: a security infrastructure (GSI), GridFTP, Execution Services (GRAM), and Information Services (MDS4).

This guide is also available as a PDF [admin.pdf]. However, each component includes online reference material, which this guide sometimes links to. The master list of documentation is here [http://www-unix.globus.org/toolkit/docs/4.0/toc_all.html].

Chapter 2. Before you begin

Before you start installing the Globus Toolkit 4.0, there are a few things you should consider. The toolkit contains many subcomponents, and you may only be interested in some of them.

There are non-webservices implementations of Security, GridFTP, Resource Management (GRAM), Replica Location Service, and Information Services (MDS2). These all run on Unix platforms only.

Additionally, there are WSRF implementations of Security, Resource Management (GRAM), Reliable File Transfer (RFT), and Information Services (Index). All the Java clients to these services run on both Windows and Unix. The WSRF GRAM service requires infrastructure that only runs on Unix systems.

Therefore, if you are new to the toolkit and want to experiment with all of the components, you may want to use a Unix system. If you are interested in the Windows development, you may restrict yourself to the Java-based software.

Chapter 3. Software Prerequisites

Required software

- Globus Toolkit installer, from Globus Toolkit development download page [http://www-unix.globus.org/toolkit/downloads/development/]
- J2SE 1.4.2+ SDK from Sun [http://java.sun.com/j2se], IBM [http://www.ibm.com/developerworks/java/jdk] or BEA [http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/jrockit] (do not use GCJ [http://gcc.gnu.org/java/]).
- Ant 1.5.1+ [http://jakarta.apache.org/ant]. Do not use the ant distributed with Fedora Core 2.
- C compiler. If gcc [http://gcc.gnu.org], avoid version 3.2. 3.2.1 and 2.95.x are okay.
- GNU tar [http://www.gnu.org/software/tar/tar.html]
- GNU sed [http://www.gnu.org/software/sed/sed.html]
- zlib 1.1.4+ [http://www.gzip.org/zlib/]
- GNU Make [http://www.gnu.org/software/make/]
- sudo [http://www.courtesan.com/sudo/]
- JDBC compliant database. For instance, postgres [http://www.postgresql.org] 7.1+
- gpt-3.2autotools2004 (shipped with the installers, but required if building standalone GPT bundles/packages)

Optional software

- IODBC [http://www.iodbc.org/] (compile requirement for RLS)
- Tomcat [http://jakarta.apache.org/tomcat/] (runtime requirement for WebMDS)

Platform Notes

In this section, the word "flavor" refers to a combination of compiler type (gcc or other), 32 or 64 bit libraries, and debugging enabled or not.

Apple MacOS X

Binaries not available due to GPT bug 258 [http://bugzilla.ncsa.uiuc.edu/show_bug.cgi?id=258].

Debian

Some kernel/libc combinations trigger a threading problem. See bug #2194 [http://bugzilla.globus.org/globus/show_bug.cgi?id=2194]. The workaround is to set LD_ASSUME_KERNEL=2.2.5 in your environment.

Fedora Core

Fedora Core 2 and later ship with a broken ant. Install your own ant from http://ant.apache.org [http://ant.apache.org/] and either remove the ant RPM or edit /etc/ant.conf, setting ANT_HOME to your own ant installation.

FreeBSD

No known issues.

HP/UX

Specify --with-flavor=vendorcc32 on the configure line. GNU tar, GNU sed, and GNU make are required on the PATH.

Tested on a PA-RISC box with HPUX 11.11 with IPv6 patches.

- HP Ansi-C compiler, version B.11.11.12
- Java 1.4.2_06
- Apache Ant 1.6.2

For HP-UX on IA64, see the patches in bug 3234 [http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3234]

IBM AIX

Supported flavors are vendorcc32dbg/vendorcc32 and vendorcc64dbg/vendorcc64 using the Visual Age compilers (xlc). No gcc flavors are supported. Specify a flavor using --with-flavor=*`flavor`*.

GNU sed, tar, and make are required before the IBM ones in the PATH.

The toolkit has been tested on AIX 5.2 with:

- Visual Age C/C++ 6.0
- 32 bit version of IBM Java 1.4
- Apache Ant 1.5.4

Red Hat

No known issues.

Sun Solaris

Supported flavors are gcc32, gcc64, vendorcc32 and vendorcc64. The dbg flavors should work as well. For gcc64, a gcc built to target 64 bit object files is required. The gcc32dbg flavor will be used by default. Specify other flavors using --with-flavor=*`flavor`*.

GPT has problems with the Sun provided perl and tar: http://www.gridpackagingtools.org/book/latest-stable/ch01s07.html

The toolkit has been tested on Solaris 9 with:

- Sun Workshop 6 update 2 C 5.3
- gcc 3.4.3
- Sun Java 1.4.2_02
- Apache Ant 1.5.4

SuSE Linux

No known issues.

Windows

Only Java-only components will build. Please choose the Java WS Core-only download and follow the instructions in the Java WS Core System Administrator's Guide [http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html].

Chapter 4. Installing GT 4.0

1. Create a user named "globus". This non-privileged user will be used to perform administrative tasks such as starting and stopping the container, deploying services, etc. Pick an installation directory, and make sure this account has read and write permissions in the installation directory.

   ## Tip

   You might need to create the target directory as root, then chown it to the globus user:

   ```
   # mkdir /usr/local/globus-4.0.0
   # chown globus:globus /usr/local/globus-4.0.0
   ```

   ## Important

   If for some reason you do *not* create a user named "globus", be sure to run the installation as a *non-root* user. In that case, make sure to pick an install directory that your user account has write access to.

2. Download the required software noted in Chapter 3.

   ## Tip

   Be aware that Apache Ant will use the Java referred to by JAVA_HOME, *not* necessarily the first Java executable on your PATH. Be sure to set JAVA_HOME to the top-level directory of your Java installation before installing.

   Also, check the the section called “Platform Notes” if your OS includes ant already. Your `/etc/ant.conf` is probably configured to use gcj, which will fail to compile the Toolkit.

3. In this guide we will assume that you are installing to `/usr/local/globus-4.0.0`, but you may replace `/usr/local/globus-4.0.0` with whatever directory you wish to install to.

   As the globus user, run:

   ```
   globus$ export GLOBUS_LOCATION=/usr/local/globus-4.0.0
   globus$ ./configure --prefix=$GLOBUS_LOCATION
   ```

   You can use command line arguments to ./configure for a more custom install. Here are the lines to enable features which are disabled by default:

   ```
   Optional Features:
     --enable-prewsmds       Build pre-webservices mds. Default is disabled.
     --enable-wsgram-condor  Build GRAM Condor scheduler interface. Default is disabled.
     --enable-wsgram-lsf     Build GRAM LSF scheduler interface. Default is disabled.
     --enable-wsgram-pbs     Build GRAM PBS scheduler interface. Default is disabled.
     --enable-i18n           Enable internationalization. Default is disabled.
     --enable-drs            Enable Data Replication Service. Default is disabled.
     [...]
   Optional Packages:
     [...]
     --with-iodbc=dir        Use the iodbc library in dir/lib/libiodbc.so.
                             Required for RLS builds.
     --with-gsiopensshargs="args"
                             Arguments to pass to the build of GSI-OpenSSH, like
                             --with-tcp-wrappers
   ```

   For a full list of options, see **./configure --help**. For a list of GSI-OpenSSH options, see Table 12.1

4. Run:

```
globus$ make
```

Note that this command can take several hours to complete. If you wish to have a log file of the build, use **tee**:

```
globus$ make 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

5. Finally, run:

```
globus$ make install
```

This completes your installation. Now you may move on to the configuration sections of the following chapters.

Your next step is to setup security, which includes picking a CA to trust, getting host certificates, user certificates, and creating a grid-mapfile. The next three chapters cover these topics.

With security setup, you may start a GridFTP server, configure a database for RFT, and configure WS-GRAM. You may also start a GSI-OpenSSH daemon, setup a MyProxy server, run RLS, and use CAS. The following chapters will explain how to configure these technologies. If you follow the chapters in order, you will make sure of performing tasks in dependency order.

Chapter 5. GT4: Security: Pre-WS Authentication & Authorization Admin Guide

Introduction

This guide contains advanced configuration information for system administrators working with Pre-WS Authentication & Authorization. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

Authentication in the Globus Toolkit is based on X.509 certificates. This document describes the configuration steps required to:

- Determine whether or not to trust certificates issued by a particular Certificate Authority (CA),
- Provide appropriate default values for use by the **grid-cert-request** command, which is used to generate certificates,
- Request *service certificates*, used by services to authenticate themselves to users, and
- Specify identity mapping information.

Building and Installing

The security tools are installed as part of the Globus Toolkit installation process. For instructions on basic installation of the Globus Toolkit, see the Installation Guide [../../admin/docbook/].

Configuring

This section describes the configuration steps required to:

- Determine whether or not to trust certificates issued by a particular Certificate Authority (CA),
- Provide appropriate default values for use by the **grid-cert-request** command, which is used to generate certificates,
- Request service certificates, used by services to authenticate themselves to users, and
- Specify identity mapping information.

In general, Globus tools will look for a configuration file in a user-specific location first, and in a systemwide location if no user-specific file was found. The configuration commands described here may be run by administrators to create systemwide defaults, and by individuals to override those defaults.

Configuring Globus to Trust a Particular Certificate Authority

The Globus tools will trust certificates issued by a CA if (and only if) it can find information about that CA in the trusted certificates directory. The trusted certificates directory is located as described in the section called "Environment variable interface" and exists either on a per machine or on a per installation basis. The following two files must exist in that directory for each trusted CA:

Table 5.1. CA files

cert_hash.0	The trusted CA certificate.
cert_hash.signing_policy	A configuration file defining the distinguished names of certificates signed by the CA.

Pre-WS Globus components will honor a certificate only if:

- its CA certificate exists (with the appropriate name) in the *TRUSTED_CA* directory, and
- the certificate's distinguished name matches the pattern described in the signing policy file.

Java-based components ignore the signing policy file and will honor all valid certificates issued by trusted CAs.

The *cert_hash* that appears in the file names above is the hash of the CA certificate, which can be found by running the command:

```
$GLOBUS_LOCATION/bin/openssl x509 -hash -noout < ca_certificate
```

Some CAs provide tools to install their CA certificates and signing policy files into the trusted certificates directory. You can, however, create a signing policy file by hand; the signing policy file has the following format:

```
access_id_CA X509 'CA Distinguished Name'
pos_rights globus CA:sign
cond_subjects globus '"Distinguished Name Pattern"'
```

In the above, the *CA Distinguished Name* is the subject name of the CA certificate, and the *Distinguished Name Pattern* is a string used to match the distinguished names of certificates granted by the CA. Some very simple wildcard matching is done -- if the *Distinguished Name Pattern* ends with a '*', then any distinguished name that matches the part of the CA subject name before the '*' is considered a match. Note: the cond_subjects line may contain a space-separated list of distinguished name patterns.

A repository of CA certificates that are widely used in academic and research settings can be found here [http://www.terena.nl/tech/task-forces/tf-aace/tacar/certs.html].

Configuring Globus to Create Appropriate Certificate Requests

The **grid-cert-request** command, which is used to create certificates, uses the following configuration files:

Table 5.2. Certificate request configuration files

globus-user-ssl.conf	defines the distinguished name to use for a user's certificate request. The format is described here [http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT].
globus-host-ssl.conf	defines the distinguished name for a host (or service) certificate request. The format is described here

	[http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT].
`grid-security.conf`	a base configuration file that contains the name and email address for the CA.
`directions`	an optional file that may contain directions on using the CA.

Many CAs provide tools to install configuration files called `globus-user-ssl.conf.`*`cert_hash`*, `globus-host-ssl.conf.`*`cert_hash`*, `grid_security.conf.`*`cert_hash`*, and `directions.`*`cert_hash`* in the trusted certificates directory. The command:

```
grid-cert-request -ca cert_hash
```

will create a certificate request based on the specified CA's configuration files. The command:

```
grid-cert-request -ca
```

will list the available CAs and let the user choose which one to create a request for.

You can specify a default CA for certificate requests (i.e., a CA that will be used if **grid-cert-request** is invoked without the `-ca` flag) by making the following symbolic links (where *GRID_SECURITY* is the grid security directory and *TRUSTED_CA* is the trusted CA directory):

```
ln -s TRUSTED_CA/globus-user-ssl.conf.cert_hash \
      GRID_SECURITY/globus-user-ssl.conf
ln -s TRUSTED_CA/globus-host-ssl.conf.cert_hash \
      GRID_SECURITY/globus-host-ssl.conf
ln -s TRUSTED_CA/grid_security.conf.cert_hash \
      GRID_SECURITY/grid_security.conf
```

And optionally, if the CA specific `directions` file exists:

```
ln -s TRUSTED_CA/directions.cert_hash \
      GRID_SECURITY/directions
```

This can also be accomplished by invoking the **grid-default-ca** command.

The `directions` file may contain specific directons on how to use the CA. There are three types of printed messages:

- *REQUEST HEADER*, printed to a certificate request file,
- *USER INSTRUCTIONS*, printed on the screen when one requests a user certificate,
- *NONUSER INSTRUCTIONS*, printed on the screen when one requests a certificate for a service,

Each message is delimited from others with lines ----- *BEGIN message type TEXT* ----- and ----- *END message type TEXT* -----. For example, `directions` file would contain the following lines:

```
----- BEGIN REQUEST HEADER TEXT -----
This is a Certificate Request file

It should be mailed to ${GSI_CA_EMAIL_ADDR}
----- END REQUEST HEADER TEXT -----
```

If this file does not exist the default messages are printed.

Requesting Service Certificates

Different CAs use different mechanisms for issuing end-user certificates; some use mechanisms that are entirely web-based, while others require you to generate a certificate request and send it to the CA. If you need to create a certificate request for a service certificate, you can do so by running:

```
grid-cert-request -host hostname -service service_name
```

where *hostname* is the fully-qualified name of the host on which the service will be running, and *service_name* is the name of the service. This will create the following three files:

Table 5.3. Certificate request files

`cert.pe` *`GRID_SECURITY/service_name/service_name`*`em`	An empty file. When you receive your actual service certificate from your CA, you should place it in this file.

`cert_re`
`quest.p`
`GRID_SECURITY/service_name/service_name``em`
	The certificate request, which you should send to your CA.

`GRID_SECURITY/service_name/service_name``key.pem`	The private key associated with your certificate request, encrypted with the pass phrase that you entered when prompted by **grid-cert-request**.

The **grid-cert-request** command recognizes several other useful options; you can list these with:

```
grid-cert-request -help
```

Specifying Identity Mapping Information

Several Globus services map distinguished names (found in certificates) to local identities (e.g., unix logins). These mappings are maintained in the `gridmap` file. The `gridmap` file is discovered according to the rules described in the section called “Environment variable interface”. A gridmap line of the form:

```
"Distinguished Name" local_name
```

maps the distinguished name *`Distinguished Name`* to the local name *`local_name`*. A gridmap line of the form:

```
"Distinguished Name" local_name1,local_name2
```

maps *`Distinguished Name`* to both *`local_name1`* and *`local_name2`*; any number of local user names may occur in the comma-separated local name list.

Several tools exist to manage gridmap files. To add an entry to the gridmap file, run:

```
$GLOBUS_LOCATION/sbin/grid-mapfile-add-entry \
        -dn "Distinguished Name" \
        -ln local_name
```

To delete an entry from the `gridmap` file, run:

```
$GLOBUS_LOCATION/sbin/grid-mapfile-delete-entry \
        -dn "Distinguished Name" \
        -ln local_name
```

To check the consistency of the `gridmap` file, run

```
$GLOBUS_LOCATION/sbin/grid-mapfile-check-consistency
```

These commands recognize several useful options, including a `-help` option, which lists detailed usage information.

The location of the `gridmap` file is determined as follows:

1. If the `GRIDMAP` environment variable is set, the `gridmap` file location is the value of that environment variable.
2. Otherwise:
    - If the user is root (uid 0), then the `gridmap` file is `/etc/grid-security/grid-mapfile`.
    - Otherwise, the `gridmap` file is `$HOME/.gridmap`.

GSI File Permissions Requirements

- End Entity [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#End_Entity_Certificate_EEC] (User [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#User_Certificate], Host [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#Host_Certificate] and Service [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#Service_Certificate]) Certificates and the GSI Authorization Callout Configuration File

:

  - May not be executable
  - May not be writable by group and other
  - Must be either regular files or soft links

- Private Keys [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#Private_Key] and Proxy Credentials [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#Proxy_Credentials]:
  - Must be owned by the current (effective) user
  - May not be executable
  - May not be readable by group and other
  - May not be writable by group and other
  - Must be either regular files or soft links
- CA Certificates [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#CA_Certificate], CA Signing Policy Files [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#CA_Signing_Policy], the Grid Map File [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#Grid_Map_File] and the GAA Configuration File [http://www-unix.globus.org/toolkit/docs/4.0/security/glossary.html#GAA_Configuration_File]:
  - Have to be either regular files or soft links
- GSI Authorization callout configuration files
  - Must exist
  - Should be world readable
  - Should not be writable by group and other
  - Should be either a regular file or a soft link
- GSI GAA configuration files
  - Must exist
  - Should be world readable
  - Should not be writable by group and other
  - Should be either a regular file or a soft link

Deploying

This section is not applicable.

Testing

There is no content available at this time.

Security Considerations

There is no content available at this time.

Troubleshooting

Credential Errors

The following are some common problems that may cause clients or servers to report that credentials are invalid:

Your proxy credential may have expired

Use **grid-proxy-info** to check whether the proxy has actually expired. If it has, generate a new proxy with **grid-proxy-init**.

The system clock on either the local or remote system is wrong

This may cause the server or client to conclude that a credential has expired.

Your end-user certificate may have expired

Use **grid-cert-info** to check your certificate's expiration date. If it has expired, follow your CA's procedures to get a new one.

The permissions may be wrong on your proxy file

If the permissions on your proxy file are too lax (for example, if others can read your proxy file), Globus Toolkit clients will not use that file to authenticate. You can "fix" this problem by changing the permissions on the file or by destroying it (with **grid-proxy-destroy**and creating a new one (with **grid-proxy-init**). However, it is still possible that someone else has made a copy of that file during the time that the permissions were wrong. In that case, they will be able to impersonate you until the proxy file expires or your permissions or end-user certificate are revoked, whichever happens first.

The permissions may be wrong on your private key file

If the permissions on your end user certificate private key file are too lax (for example, if others can read the file), **grid-proxy-init** will refuse to create a proxy certificate. You can "fix" this by changing the permissions on the private key file; however, you will still have a much more serious problem: it's possible that someone has made a copy of your private key file. Although this file is encrypted, it is possible that someone will be able to decrypt the private key, at which point they will be able to impersonate you as long as your end user certificate is valid. You should contact your CA to have your end-user certificate revoked and get a new one.

The remote system may not trust your CA

Verify that the remote system is configured to trust the CA that issued your end-entity certificate. See the [TODO: add admin guide link] for details.

You may not trust the remote system's CA

Verify that your system is configured to trust the remote CA (or that your environment is set up to trust the remote CA). See the [TODO: add admin guide link] for details.

There may be something wrong with the remote service's credentials

It is sometimes difficult to distinguish between errors reported by the remote service regarding your credentials and errors reported by the client interface regarding the remote service's credentials. If you can't find anything wrong with your credentials, check for the same conditions (or ask a remote administrator to do so) on the remote system.

Gridmap errors

The following are some common problems that may cause clients or servers to report that user are not authorized:

The content of the gridmap file does not conform to the expected format

Use **grid-mapfile-check-consistency** to make sure that your gridmap conforms to the expected format.

The gridmap file does not contain a entry for your DN

Use **grid-mapfile-add-entry** to add the relevant entry.

Environment variable interface

- `X509_USER_PROXY` specifies the path to the proxy credential. If `X509_USER_PROXY` is not set, the proxy credential is created (by **grid-proxy-init**) and searched for (by client programs) in an operating-system-dependent local temporary file.
- `X509_USER_CERT` and `X509_USER_KEY` specify the path to the end entity (user, service, or host) certificate and corresponding private key. The paths to the certificate and key files are determined as follows:
  - For user credentials:
    1. If `X509_USER_CERT` and `X509_USER_KEY` exist and contain a valid certificate and key, those files are used.
    2. Otherwise, if the files `usercert.pem` and `userkey.pem` exist in the user's `.globus` directory, those files are used.
    3. Otherwise, if a PKCS-12 file called `usercred.p12` exists in the user's `.globus` directory, the certificate and key are read from that file.
  - For service credentials:
    1. If `X509_USER_CERT` and `X509_USER_KEY` exist and contain a valid certificate and key, those files are used.
    2. Otherwise, if the files `/etc/grid-security/`*`service/service`*`cert` and `/etc/grid-security/`*`service/service`*`key` exist and contain a valid certificate and key, those files are used.
    3. Otherwise, if the files `$GLOBUS_LOCATION/etc/grid-security/`*`service/service`*`cert` and `$GLOBUS_LOCATION/etc/grid-security/`*`service/service`*`key` exist and contain a valid certificate and key, those files are used.

4. Otherwise, if the files *service/service*`cert` and *service/service*`key` in the user's `.globus` directory, exist and contain a valid certificate and key, those files are used.

- For host credentials:

1. If `X509_USER_CERT` and `X509_USER_CERT` exist and contain a valid certificate and key, those files are used.
2. Otherwise, if the files `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem` exist and contain a valid certificate and key, those files are used.
3. Otherwise, if the files `$GLOBUS_LOCATION/etc/grid-security/hostcert.pem` and `$GLOBUS_LOCATION/etc/grid-security/hostkey.pem` exist and contain a valid certificate and key, those files are used.
4. Otherwise, if the files `hostcert.pem` and `hostkey.pem` in the user's `.globus` directory, exist and contain a valid certificate and key, those files are used.

- `GRIDMAP` specifies the path to the gridmap file, which is used to map distinguished names (found in certificates) to local names (such as login accounts). The location of the gridmap file is determined as follows:

1. If the `GRIDMAP` environment variable is set, the gridmap file location is the value of that environment variable.
2. Otherwise:
   - If the user is root (uid 0), then the gridmap file is `/etc/grid-security/grid-mapfile`.
   - Otherwise, the gridmap file is `$HOME/.gridmap`.

- `X509_CERT_DIR` is used to specify the path to the trusted certificates directory; this directory contains information about which CAs are trusted (including the CA certificates themselves) and, in some cases, configuration information used by **grid-cert-request** to formulate certificate requests. The location of the trusted certificates directory is determined as follows:

1. If the `X509_CERT_DIR` environment variable is set, the trusted certificates directory is the value of that environment variable.
2. Otherwise, if `$HOME/.globus/certificates` exists, that directory is the trusted certificates directory.
3. Otherwise, if `/etc/grid-security/certificates` exists, that directory is the trusted certificates directory.
4. Finally, if `$GLOBUS_LOCATION/share/certificates` exists, then it is the trusted certificates directory.

- `GSI_AUTHZ_CONF` is used to specify he path to the GSI authorization callout configuration file. This file is used to configure authorization callouts used by both the gridmap and the authorization API. The location of the GSI authorization callout configuration file is determined as follows:

1. If the `GSI_AUTHZ_CONF` environment variable is set, the authorization callout configuration file location is the value of this environment variable.
2. Otherwise, if `/etc/grid-security/gsi-authz.conf` exists, then this file is used.
3. Otherwise, if `$GLOBUS_LOCATION/etc/gsi-authz.conf` exists, then this file is used.
4. Finally, if `$HOME/.gsi-authz.conf` exists, the this file is used.

- `GSI_GAA_CONF` is used to specify he path to the GSI GAA (Generic Authorization and Access control) configuration file. This file is used to configure policy language specific plugins to the GAA-API. The location of the GSI GAA configuration file is determined as follows:

  1. If the `GSI_GAA_CONF` environment variable is set, the GAA configuration file location is the value of this environment variable.
  2. Otherwise, if `/etc/grid-security/gsi-gaa.conf` exists, then this file is used.
  3. Otherwise, if `$GLOBUS_LOCATION/etc/gsi-gaa.conf` exists, then this file is used.
  4. Finally, if `$HOME/.gsi-gaa.conf` exists, the this file is used.

- `GRID_SECURITY_DIR` specifies a path to a directory containing configuration files that specify default values to be placed in certificate requests; this environment variable is used only by the **grid-cert-request** and **grid-default-ca** commands.
- The location of the grid security directory is determined as follows:

  1. If the `GRID_SECURITY_DIR` environment variable is set, the grid security directory is the value of that environment variable.
  2. If the configuration files exist in `/etc/grid-security`, the grid security directory is that directory.
  3. if the configuration files exist in `$GLOBUS_LOCATION/etc`, the grid security directory is that directory.

Chapter 6. Basic Security Configuration

Set environment variables

In order for the system to know the location of the Globus Toolkit commands you just installed, you must set an environment variable and source the `globus-user-env.sh` script.

1. As globus, set **GLOBUS_LOCATION** to where you installed the Globus Toolkit. This will be one of the following:

   - Using Bourne shells:

     ```
     globus$ export GLOBUS_LOCATION=/path/to/install
     ```

   - Using csh:

     ```
     globus$ setenv GLOBUS_LOCATION /path/to/install
     ```

2. Source `$GLOBUS_LOCATION/etc/globus-user-env.`*`{sh,csh}`* depending on your shell.

   - Use *`.sh`* for Bourne shell:

     ```
     globus$ . $GLOBUS_LOCATION/etc/globus-user-env.sh
     ```

   - Use *`.csh`* for C shell.

     ```
     globus$ source $GLOBUS_LOCATION/etc/globus-user-env.csh
     ```

Obtain host certificates

You must have X509 certificates to use the GT 4.0 software securely (referred to in this documentation as *host certificates*). For an overview of certificates for GSI (security) see GSI Configuration Information [http://www.globus.org/security/config.html].

Host certificates must be:

- consist of the following two files: `hostcert.pem` and `hostkey.pem`
- must be in the appropriate directory for secure services: `/etc/grid-security/`
- must be for a machine which has a consistent name in DNS; you should *not* run it on a computer using DHCP where a different name could be assigned to your computer.

You have the following options:

Request a certificate from an existing CA

Your best option is to use an already existing CA. You may have access to one from the company you work for, or an organization you are affiliated with. Some universities provide certificates for their members and affiliates. Contact your support organization for details about how to acquire a certificate. You may find your CA listed in the TERENA Repository [http://www.terena.nl/tech/task-forces/tf-aace/tacar/certs.html].

If you already have a CA, you will need to follow their configuration directions. If they include a CA setup package, follow the CAs instruction on how to install the setup package. If they do not, you will need to create an `/etc/grid-security/certificates` directory and include the CA cert and signing policy in that directory. See Configuring a Trusted CA [http://www.globus.org/security/config.html#Configuring%20a%20Trusted%20CA] for more details.

This type of certificate is best for service deployment and Grid inter-operation.

SimpleCA

SimpleCA provides a wrapper around the OpenSSL CA functionality and is sufficient for simple Grid services. Alternatively, you can use OpenSSL's **CA.sh** command on its own. Instructions on how to use the SimpleCA can be found in Chapter 7.

SimpleCA is suitable for testing or when a certificate authority is not available.

Low-trust certificate

Globus offers a low-trust certificate available at http://gcs.globus.org:8080/gcs. This option should only be used as a last resort because it does not fulfill some of the duties of a real Certificate Authority.

This type of certificate is best suited for short term testing.

Make the host credentials accessible by the container

The host key (`/etc/grid-security/hostkey.pem`) is only readable to root. The container (hosting environment) will be running as a non-root user (probably the *`globus`* user) and in order to have a set of host credentials which are readable by the container, we need to copy the host certificate and key and change the ownership to the container user.

Note

This step assumes you have obtained a signed host certficate from your CA.

As root, run:

```
root# cd /etc/grid-security
root# cp hostkey.pem containerkey.pem
root# cp hostcert.pem containercert.pem
root# chown globus.globus containerkey.pem containercert.pem
```

At this point the certificates in `/etc/grid-security` should look something like:

```
root# ls -l *.pem
-rw-r--r-- 1 globus globus 1785 Oct 14 14:47 containercert.pem
-r-------- 1 globus globus  887 Oct 14 14:47 containerkey.pem
-rw-r--r-- 1 root   root   1785 Oct 14 14:42 hostcert.pem
-r-------- 1 root   root    887 Sep 29 09:59 hostkey.pem
```

Add authorization

Add authorizations for users:

Create `/etc/grid-security/grid-mapfile` as root.

You need two pieces of information:

- the subject name of a user

- the account name it should map to.

The syntax is one line per user, with the certificate subject followed by the user account name.

Run **grid-cert-info** to get your subject name, and **whoami** to get the account name:

```
bacon$ grid-cert-info -subject
/O=Grid/OU=GlobusTest/OU=simpleCA-mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon
bacon$ whoami
bacon
```

You may add the line by running the following as root:

```
root# $GLOBUS_LOCATION/sbin/grid-mapfile-add-entry -dn \
"/O=Grid/OU=GlobusTest/OU=simpleCA-mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon" \
-ln bacon
```

The corresponding line in the `grid-mapfile` should look like:

```
"/O=Grid/OU=GlobusTest/OU=simpleCA-mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon" bacon
```

Important

The quotes around the subject name are *important*, because it contains spaces.

Verify Basic Security

Now that you have installed a trusted CA, acquired a hostcert and a usercert, you may verify that your security setup is complete. As your user account, run the following command:

```
bacon$ grid-proxy-init -verify -debug

User Cert File: /home/bacon/.globus/usercert.pem
User Key File: /home/bacon/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u506
Your identity: /DC=org/DC=doegrids/OU=People/CN=Charles Bacon 332900
Enter GRID pass phrase for this identity:
Creating proxy ...++++++++++++
..................++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Fri Jan 28 23:13:22 2005
```

There are a few things you can notice from this command. Your usercert and key are located in `$HOME/.globus/`. The proxy certificate is created in `/tmp/`. The "up" stands for "user proxy", and the *`_u506`* will be your UNIX userid. It also prints out your distinguished name (DN), and the proxy is valid for 12 hours.

If this command succeeds, your single node is correctly configured. If it does not succeed, or you want to continue to configure multiple nodes, you may want to continue to the full security overview in the next chapter. Otherwise, you may proceed to the services chapter.

Chapter 7. GT 4.0 SimpleCA: Admin Guide

Introduction

This guide contains advanced configuration information for system administrators working with SimpleCA. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

The following are instructions for how to use SimpleCA to set up certificates for a GT 4.0 installation.

SimpleCA provides a wrapper around the OpenSSL CA functionality and is sufficient for simple Grid services. Alternatively, you can use OpenSSL's **CA.sh** command on its own. SimpleCA is suitable for testing or when a certificate authority (CA) is not available. You can find other CA options in the section called “Obtain host certificates”.

Building and Installing

Create users

Make sure you have the following users on your machine:

- Your *user* account, which will be used to run the client programs.
- A generic *globus* account, which will be used to perform administrative tasks such as starting and stopping the container, deploying services, etc. This user will also be in charge of managing the SimpleCA. To do this, make sure this account has read and write permissions in the `$GLOBUS_LOCATION` directory.

Run the setup script

A script was installed to set up a new SimpleCA. You only need to run this script *once* per Grid.

Run the setup script:

```
$GLOBUS_LOCATION/setup/globus/setup-simple-ca
```

2.1 Configure the subject name

This script prompts you for information about the CA you wish to create:

```
The unique subject name for this CA is:
cn=Globus Simple CA, ou=simpleCA-mayed.mcs.anl.gov, ou=GlobusTest, o=Grid

Do you want to keep this as the CA subject (y/n) [y]:
```

where:

Table 7.1. CA Name components

cn	Represents "common name". Identifies this particular certificate as the CA certificate within the "GlobusTest/ simpleCA-hostname" domain, which in this case is Globus Simple CA.
ou	Represents "organizational unit". Identifies this CA from other CAs created by SimpleCA by other people. The second "ou" is specific to your hostname (in this cases GlobusTest).
o	Represents "organization". Identifies the Grid.

Press **y** to keep the default subject name (recommended).

Configure the CA's email

The next prompt looks like:

```
Enter the email of the CA (this is the email where certificate
requests will be sent to be signed by the CA):
```

Enter the email address where you intend to receive certificate requests. It should be your real email address that you check, not the address of the globus user.

Configure the expiration date

Then you'll see:

```
The CA certificate has an expiration date. Keep in mind that
once the CA certificate has expired, all the certificates
signed by that CA become invalid.  A CA should regenerate
the CA certificate and start re-issuing ca-setup packages
before the actual CA certificate expires.  This can be done
by re-running this setup script.  Enter the number of DAYS
the CA certificate should last before it expires.
[default: 5 years (1825 days)]:
```

This is the number of days for which the CA certificate is valid. Once this time expires, the CA certificate will have to be recreated, and all of its certificates regranted.

Accept the default (recommended).

Enter a passphrase

Next you'll see:

```
Generating a 1024 bit RSA private key
........++++++
..............++++++
writing new private key to '/home/globus/.globus/simpleCA//private/cakey.pem'
Enter PEM pass phrase:
```

The passphrase of the CA certificate will be used only when signing certificates (with **grid-cert-sign**). It should be hard to guess, as its compromise may compromise all the certificates signed by the CA.

Enter your passphrase.

Important:

Your passphrase must *not* contain any spaces.

Confirm generated certificate

Finally you'll see the following:

```
A self-signed certificate has been generated
for the Certificate Authority with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-mayed.mcs.anl.gov/CN=Globus Simple CA

If this is invalid, rerun this script

setup/globus/setup-simple-ca

and enter the appropriate fields.

-----------------------------------------------------------------

The private key of the CA is stored in /home/globus/.globus/simpleCA//private/cak ey
The public CA certificate is stored in /home/globus/.globus/simpleCA//cacert.pem

The distribution package built for this CA is stored in

/home/globus/.globus/simpleCA//globus_simple_ca_68ea3306_setup-0.17.tar.gz
```

This information will be important for setting up other machines in your grid. The number *68ea3306* in the last line is known as your *CA hash*. It will be an 8 hexadecimal digit string.

Press any key to acknowledge this screen.

Your CA setup package finishes installing and ends the procedure with the following reminder:

```
**************************************************************************

Note: To complete setup of the GSI software you need to run the
following script as root to configure your security configuration
directory:

/opt/gt4/setup/globus_simple_ca_68ea3306_setup/setup-gsi

For further information on using the setup-gsi script, use the -help
option.  The -default option sets this security configuration to be
the default, and -nonroot can be used on systems where root access is
not available.

**************************************************************************

setup-ssl-utils: Complete
```

We'll run the setup-gsi script in the next section. For now, just notice that it refers to your `$GLOBUS_LOCATION` and the *`CA Hash`* from the last message.

Complete setup of GSI

To finish the setup of GSI, we'll run the script noted in the previous step.

Run the following as root (or, if no root privileges are available, add the **-nonroot** option to the command line):

`$GLOBUS_LOCATION/setup/globus_simple_ca_`*`CA_Hash`*`_setup/setup-gsi -default`

The output should look like:

```
setup-gsi: Configuring GSI security
Installing /etc/grid-security/certificates//grid-security.conf.CA_Hash...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

Host certificates

You must request and sign a host certificate and then copy it into the appropriate directory for secure services. The certificate must be for a machine which has a consistent name in DNS; you should not run it on a computer using DHCP where a different name could be assigned to your computer.

3.1 Request a host certificate

As root, run:

```
grid-cert-request -host 'hostname'
```

This creates the following files:

- `/etc/grid-security/hostkey.pem`
- `/etc/grid-security/hostcert_request.pem`
- (an empty) `/etc/grid-security/hostcert.pem`

Note: If you are using your own CA, follow their instructions about creating a hostcert (one which has a commonName (CN) of your hostname), then place the cert and key in the /etc/grid-security/ location. You may then proceed to the section called “User certificates”.

Sign the host certificate

1. As globus, run:

   ```
   grid-ca-sign -in hostcert_request.pem -out hostsigned.pem
   ```

2. A signed host certificate, named `hostsigned.pem` is written to the current directory.
3. When prompted for a passphrase, enter the one you specified in the section called “Enter a passphrase” (for the private key of the CA certificate.)
4. As root, move the signed host certificate to `/etc/grid-security/hostcert.pem`.

The certificate should be owned by root, and read-only for other users.

The key should be read-only by root.

User certificates

Users also must request user certificates, which you will sign using the *globus* user.

Request a user certificate

As your normal user account (*not globus*), run:

```
grid-cert-request
```

After you enter a passphrase, this creates

- `~$USER/.globus/usercert.pem` (empty)
- `~$USER/.globus/userkey.pem`
- `~$USER/.globus/usercert_request.pem`

Email the `usercert_request.pem` file to the SimpleCA maintainer.

Sign the user certificate

1. As the SimpleCA owner *globus*, run:

   ```
   grid-ca-sign -in usercert_request.pem -out signed.pem
   ```

2. When prompted for a password, enter the one you specified in the section called “Enter a passphrase” (for the private key of the CA certificate).
3. Now send the signed copy (`signed.pem`) back to the user who requested the certificate.
4. As your normal user account (*not globus*), copy the signed user certificate into `>~/.globus/` and rename it as `usercert.pem`, thus replacing the empty file.

The certificate should be owned by the user, and read-only for other users.

The key should be read-only by the owner.

Configuring

[high-level characterization of the configuration options for the component here]

Configure SimpleCA for multiple machines

So far , you have a single machine configured with SimpleCA certificates. Recall that in the section called “Complete setup of GSI” a CA setup package was created in `.globus/simpleCA/globus_simple_ca_HASH_setup-0.17.tar.gz`. If you want to use your certificates on another machine, you must install that CA setup package on that machine.

To install it, copy that package to the second machine and run:

```
$GLOBUS_LOCATION/sbin/gpt-build globus_simple_ca_HASH_setup-0.17.tar.gz gcc32dbg
```

Then you will have to perform **setup-gsi -default** from the section called “Sign the host certificate”.

If you are going to run services on the second host, it will need its own the section called “Host certificates” and grid-mapfile (as described in the basic configuration instructions in the section called “Add authorization”).

You may re-use your user certificates on the new host. You will need to copy the requests to the host where the SimpleCA was first installed in order to sign them.

Deploying

[information about deploying the component into various containers/environments]

Testing

To verify that the SimpleCA certificate is installed in `/etc/grid-security/certificates` and that your certificate is in place with the correct permissions, run:

```
user$ grid-proxy-init -debug -verify
```

After entering your passphrase, successful output looks like:

```
[bacon@mayed schedulers]$ grid-proxy-init -debug -verify

User Cert File: /home/user/.globus/usercert.pem
User Key File: /home/user/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1817
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=User Na
Enter GRID pass phrase for this identity:
Creating proxy ..............................++++++++++++
..............++++++++++++
Done
Proxy Verify OK
Your proxy is valid until: Sat Mar 20 03:01:46 2004
```

Security Considerations

[describe security considerations relevant for this component]

Troubleshooting

[help for common problems sysadmins may experience]

Chapter 8. GT 4.0 GridFTP : System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with GridFTP. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation. This guide should help you configure and run the GridFTP server in some standard configurations.

Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

Building and Installing

GridFTP is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide [../../admin/docbook/]. No extra installation steps are required for this component.

Building and Installing only GridFTP

If you wish to install GridFTP without installing the rest of the Globus Toolkit, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide [../../../admin/docbook/]. Perform steps 1-3, as written (Note that you do not need Ant, a JDK, or a JDBC database to build only GridFTP). However, instead of running "make" as directed in step 4,

Run:

```
globus$ make gridftp
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make gridftp 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

Building and Installing only the GridFTP server

If you wish to install only the GridFTP server, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide [../../../admin/docbook/] for prerequisites. Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
globus$ make gpt globus_gridftp_server
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make gpt globus_gridftp_server 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

Building and Installing a static GridFTP server

If you wish to build and install a statically linked set of GridFTP binaries, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide [../../../admin/docbook/] for prerequisites. Follow steps 1-2 as written. In step 3, however, you should

Run:

```
globus$ export GLOBUS_LOCATION=/usr/local/globus-4.0.0
globus$ ./configure --prefix=$GLOBUS_LOCATION --with-buildopts="--static"

globus$ make gpt globus_gridftp_server
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make gpt globus_gridftp_server 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

Configuring

Please see the configuration section of the Public Interfaces Guide

Deploying the GridFTP Server: `globus-gridftp-server`

It is assumed that the toolkit installation was successful and that Globus security is properly configured. For more information, see the Installation Guide [../../admin/docbook/]. Also be sure to reference the the section called “Configuration interface”.

Running in daemon mode

The server should generally be run as root in daemon mode, though it is possible to run it as a user (see below). When run as root you will need to have a host certificate.

Run the server:

```
globus-gridftp-server < -s | -S > <args>
```

where:

-s	Runs in the foreground. (this is the default mode)
-S	Detaches from the terminal and runs in the background.

The following additional steps may be required when running as a user other than root.

- Create a ~/.gridmap file, containing the DNs of any clients you wish to allow, mapped to the current username.
- Create proxy: `grid-proxy-init`

Running under inetd or xinetd

The -i command line option enables the server to be run under inetd or xinetd.

See the section called “Configuration interface” for example xinetd and inetd configuration entries.

Remote data-nodes and striped operation

The GridFTP server now supports separate front end (client control connection) and back end (data node) processes. In addition, a single front end process may connect to multiple back end data nodes.

When multiple back end data nodes are available, the server is said to be in a striped configuration, or simply, is a striped server. In this mode, transfers are divided over all available data nodes, thus allowing the combined bandwidth of all data nodes to be used.

Note: The connection between the front end and data nodes is referred to as the *ipc channel*.

The ability to use inetd or daemon execution modes applies to both front end servers and data nodes, and the same certificate and user requirements apply.

To start the front end:

```
globus-gridftp-server <args> -r <host:port>[,<host:port>,...]
```
To start the data-node:

```
globus-gridftp-server -p <port> -dn
```
The `-p <port>` option used on the data-node is the port that will be used for ipc connections. This is the port that you will register with the front end server.

For example:

```
machineB> globus-gridftp-server -p 6000 -dn
machineC> globus-gridftp-server -p 7000 -dn
machineA> globus-gridftp-server -p 5000 -r machineB:6000,machineC:7000
```
The client would only connect to the front end at `machineA:5000`, for example, using globus-url-copy with the -stripe option:

```
globus-url-copy -stripe gsiftp://machineA:5000/file file:///destination
   or
globus-url-copy -stripe gsiftp://machineA:5000/file gsiftp://machineX/destination
```
Where machineX may be another striped server or a standard GridFTP server.

Seperation of Processes

As is illustrated above, the GridFTP server can be seperated into frontend and data node processes. This is the arcitecture used to achieve a striped server, but it can also be exploited to achieve a higher level of security.

Running the server as root is often desirable because it allows the server to fork and setuid on a child processes related to an authenticated user. This allows the server to leverage the operating systems file system permissions and other security devices. However, it is not at all desireable to have a root running process listening on a port open to the world. If an attacker were to comprimise the process they could obtain root level access to the machine.

To overcome this security risk the gridftp server can be run in a frontend/backend manner. The frontend can be run as anyuser, say user globus, that has very limited access to the machine. The frontend is the processes open to the outside world. If it is comprimised an attacker has only gained access to that limited account. The backend is run as root, but configured to only allow connections from the frontend.

To start the front end:

```
globus-gridftp-server -p 7000 -r localhost:7001
```

and the backend:

```
globus-gridftp-server -p 7001 -dn -allow-from 127.0.0.1
```

Testing

If the globus-ftp-client-test package has been installed, our standard test suite may be run to verify functionality on your platform. Simply set up the globus environment, chdir to `$GLOBUS_LOCATION/test/globus_ftp_client_test/` and run `./TESTS.pl`

Please also see the Call for Community Testing [GridFTP_Call_for_Testing.html].

Security Considerations

The following are points to consider relative to security:

Two ways to configure your server

We now provide two ways to configuring your server:

- The classic installation. This is equivalent to any FTP server you would normally install. It is run as a root setuid process. Once the user is authenticated, the process does a setuid to the appropriate non-privileged user account.
- A new split process installation. In this configuration, the server consists of two processes:
  - The control channel (the process the external user connects to) runs as a non-privileged user (typically the globus user).
  - The data channel (the process that access the file system and moves the data) runs as a root setuid program as before, but is only contacted by the control channel process from a local machine. This means an external user is never connected to a root running process and thus minimizes the impact of an exploit. This does, however, require that a copy of the host cert and host key be owned by the non-privileged user. If you use this configuration, the non-privileged user should not have write permission to executables, configuration files, etc..

New authentication options

There are new authentication options available for the server in GT4.0.0:

- Anonymous: The server now supports anonymous access. In order for this to work, a configuration switch must explicitly enable it, a list of acceptable usernames must be defined, and an account under which the anonymous user should run must be defined. If the necessary configurations are in place, and the client presents a username that is in the list of acceptable anonymous users, then the session will be accepted and the process will setuid to the anonymous user account. We do not support chroot in this version of the server.
- Username / Password: This is standard FTP authentication. It uses a separate password file, used only by the GridFTP server, *NOT* the system password file.

Warning

WE HIGHLY RECOMMEND YOU NOT USE THIS. YOU WILL BE SENDING YOUR PASSWORD IN CLEAR TEXT OVER THE NETWORK.

We do, however, have some user communities who run only on internal networks for testing purposes and who do not wish to deal with obtaining GSI credentials. If you are considering this, we would recommend that you look at Simple CA and set up your own testbed CA. This can be done in less than an hour and then provides you full GSI security.

Troubleshooting

If you are having problems using the GridFTP server, try the steps listed below. If you have an error, try checking the server logs if you access to them. By default, the server logs to stderr, unless it is running from inetd, or its execution mode is detached, in which case logging is disabled by default.

The command line options -d , -log-level, -L and -logdir can affect where logs will be written, as can the configuration file options log_single and log_unique. See the Configuration information in the Public Interfaces Guide for more information on these and other configuration options.

Establish control channel connection

Verify that you can establish a control channel connection and that the server has started successfully by doing a telnet to the port on which the server is running:

```
% telnet localhost 2811
                Trying 127.0.0.1...
                Connected to localhost.
                Escape character is '^]'.
                220 GridFTP Server mldev.mcs.anl.gov 2.0 (gcc32dbg, 1113865414-1) ready.
```

If you see anything other than a 220 banner such as that, then the server has not started correctly.

Verify that there are no configuration files being unexpectedly loaded from /etc/grid-security/gridftp.conf or $GLOBUS_LOCATION/etc/gridftp.conf. If those files exist, and you did not intend for them to be used, rename them to .save, or specify -c none on the command line and try again.

If you can log into the machine where the server is, try running the server from the command line with only the -s :

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -s
```

The server will print the port it is listening on:

```
Server listening at gridftp.mcs.anl.gov:57764
```

Now try and telnet to that port. If you still do not get the banner listed above, something is preventing the socket connection. Check firewalls, tcp-wrapper, etc..

If you now get a correct banner, add -p 2811 (you will have to disable (x)inetd on port 2811 if you are using them or you will get port already in use):

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -s -p 2811
```

Now telnet to port 2811. If this does not work, something is blocking port 2811, check firewalls, tcp-wrapper, etc..

If this works correctly then re-enable your normal server, but remove all options but -i, -s, or -S.

Now telnet to port 2811. If this does not work, something is wrong with your service configuration check / etc/services, (x)inetd config, (x)inetd restarted, etc..

If this works, begin adding options back one at a time, verifying that you can telnet to the server after each option is added. Continue this till you find the problem or get all the options you want.

At this point, you can establish a control connection. Now try running globus-url-copy:

Try running globus-url-copy

Once you've verified that you can establish a control connection, try to make a transfer using globus-url-copy.

If you are doing a client/server transfer (one of your URLs has `file:` in it) then try:

```
globus-url-copy -vb -dbg gsiftp://host.server,running.on/dev/zero file:///dev/null
```

This will run until you control-c the transfer. If that works, reverse the direction:

```
globus-url-copy -vb -dbg file:///dev/zero gsiftp://host.server.running.on/dev/null
```

Again, this will run until you control-c the transfer.

If you are doing a third party transfer run this command:

```
globus-url-copy -vb -dbg gsiftp://host.server1.on/dev/zero gsiftp://host.server2.on/dev/null
```

Again, this will run until you control-c the transfer.

If the above transfers work, try your transfer again. If it fails, you likely have some sort of file permissions, typo in a file name, etc..

If your server starts...

If the server has started correctly, and your problem is with a security failure or gridmap lookup failure, verify that you have security configured properly here.

If the server is running and your client successfully authenticates, but has a problem at some other time during the session, please ask for help on discuss@globus.org. When you send mail or submit bugs, please always include as much of the following information as possible:

- Specs on all hosts involved (OS, processor, RAM, etc)
- globus-url-copy -version
- globus-url-copy -versions
- Output from telnet test above
- The actual command line you ran with -dbg added. Don't worry if the output gets long.
- Check that you are getting a FQDN and /etc/hosts that is sane.
- The server configuration and setup (/etc/services entries, (x)inetd configs, etc.)
- Any relevant lines from the server logs (not the entire log please)

Usage statistics collection by the Globus Alliance

The following GridFTP-specific usage statistics are sent in a UDP packet at the end of each transfer, in addition to the standard header information described in the Usage Stats [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] section.

- Start time of the transfer
- End time of the transfer
- Version string of the server
- TCP buffer size used for the transfer
- Block size used for the transfer

- Total number of bytes transferred
- Number of parallel streams used for the transfer
- Number of stripes used for the transfer
- Type of transfer (STOR, RETR, LIST)
- FTP response code -- Success or failure of the transfer

Note

The client (globus-url-copy) does NOT send any data. It is the *servers* that send the usage statistics.

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve and gauge the usage of the GridFTP server. Nevertheless, if you wish to disable this feature for GridFTP only, see the Logging section of the GridFTP configuration and command line options [http://www-unix.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Public_Interfaces.html#config]. Note that you can disable transmission of usage statistics globally for all C components by setting "GLOBUS_USAGE_OPTOUT=1" in your environment.

Also, please see our policy statement [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] on the collection of usage statistics.

Configuration interface

GridFTP server configuration overview

Note: Command line options and configuration file options may both be used but the command line *overrides* the config file.

The configuration file is read from the following locations, in the given order. Only the first found will be loaded.

- Path specified with the `-c <configfile>` command line option.
- $GLOBUS_LOCATION/etc/gridftp.conf
- /etc/grid-security/gridftp.conf

Options are allowed one per line, with the format:

`<option> <value>`
If the value contains spaces, they should be enclosed in double-quotes (")

Flags or boolean options should only have a value of 0 or 1

Blank lines and lines begining with # are ignored.

For example:

```
port 5000
allow_anonymous 1
anonymous_user bob
banner "Welcome!"
```

GridFTP server configuration options

Table 8.1. Informational Options

`help <0|1>`
`-h`
`-help`
	Show usage information and exit.
	
	Default value: FALSE

`longhelp <0|1>`
`-hh`
`-longhelp`
	Show more usage information and exit.
	
	Default value: FALSE

`version <0|1>`
`-v`
`-version`
	Show version information for the server and exit.
	
	Default value: FALSE

`versions <0|1>`
`-V`
`-versions`
	Show version information for all loaded globus libraries and exit.
	
	Default value: FALSE

Table 8.2. Modes of Operation

`inetd <0|1>`
`-i`
`-inetd`
	Run under an inetd service.
	
	Default value: FALSE

`daemon <0|1>`
`-s`
`-daemon`
	Run as a daemon. All connections will fork off a new process and setuid if allowed.
	
	Default value: TRUE

`detach <0|1>`
`-S`
`-detach`
	Run as a background daemon detached from any controlling terminals.
	
	Default value: FALSE

`exec <string>` `-exec <string>`	For staticly compiled or non-GLOBUS_LOCATION standard binary locations, specify the full path of the server binary here. Only needed when run in daemon mode. Default value: not set
`chdir <0|1>` `-chdir`	Change directory when the server starts. This will change directory to the dir specified by the chdir_to option. Default value: TRUE

chdir_to <string> -chdir-to <string>	Directory to chdir to after starting. Will use / if not set. Default value: not set
fork <0|1> -f -fork	Server will fork for each new connection. Disabling this option is only recommended when debugging. Default value: TRUE
single <0|1> -1 -single	Exit after a single connection Default value: FALSE

Table 8.3. Authentication, Authorization, and Security Options

auth_level <number> -auth-level <number>	0 = Disables all authorization checks. 1 = Authorize identity only. 2 = Authorize all file/resource accesses. If not set uses level 2 for front ends and level 1 for data nodes. Default value: not set
allow_from <string> -allow-from <string>	Only allow connections from these source ip addresses. Specify a comma seperated list of ip address fragments. A match is any ip address that starts with the specified fragment. Example: '192.168.1.' will match and allow a connection from 192.168.1.45. Note that if this option is used any address not specifically allowed will be denied. Default value: not set
deny_from <string> -deny-from <string>	Deny connections from these source ip addresses. Specify a comma seperated list of ip address fragments. A match is any ip address that starts with the specified fragment. Example: '192.168.2.' will match and deny a connection from 192.168.2.45. Default value: not set
cas <0|1> -cas	Enable CAS authorization. Default value: TRUE
secure_ipc <0|1> -si -secure-ipc	Use GSI security on ipc channel. Default value: TRUE
ipc_auth_mode <string> -ia <string> -ipc-auth-mode <string>	Set GSI authorization mode for the ipc connection. Options are: none, host, self or subject: <subject> Default value: host

`allow_anonymous <0|1>`
`-aa`
`-allow-anonymous`
	Allow cleartext anonymous access. If server is running as root anonymous_user must also be set. Disables ipc security.
	
	Default value: FALSE

`anonymous_names_allowed <string>` `-anonymous-names-allowed <string>`	Comma seperated list of names to treat as anonymous users when allowing anonymous access. If not set, the default names of 'anonymous' and 'ftp' will be allowed. Use '*' to allow any username. Default value: not set
`anonymous_user <string>` `-anonymous-user <string>`	User to setuid to for an anonymous connection. Only applies when running as root. Default value: not set
`anonymous_group <string>` `<string>`	Group to setgid to for an anonymous connection. If unset, the default group of anonymous_user will be used. Default value: not set
`pw_file <string>` `-password-file <string>`	Enable cleartext access and authenticate users against this /etc/passwd formatted file. Default value: not set
`connections_max <number>` `-connections-max <number>`	Maximum concurrent connections allowed. Only applies when running in daemon mode. Unlimited if not set. Default value: not set
`connections_disabled <0|1>` `-connections-disabled`	Disable all new connections. Does not affect ongoing connections. This would have be set in the configuration file and then the server issued a SIGHUP in order to reload that config. Default value: FALSE

Table 8.4. Logging Options

`log_level <string>`
`-d <string>`
`-log-level <string>`
	Log level. A comma seperated list of levels from: 'ERROR, WARN, INFO, DUMP, ALL'. Example: error,warn,info. You may also specify a numeric level of 1-255.
	
	Default value: ERROR

`log_module <string>` `-log-module <string>`	globus_logging module that will be loaded. If not set, logfile options apply. Default value: not set

`log_single <string>`
`-l <string>`
`-logfile <string>`
	Path of a single file to log all activity to. If neither this option or log_unique is set, logs will be written to stderr unless the execution mode is detached or inetd, in which case logging will be disabled.
	
	Default value: not set

`log_unique <string>`
`-L <string>`
`-logdir <string>`
	Partial path to which 'gridftp.(pid).log' will be appended to construct the log filename. Example: -L /var/log/gridftp/ will create a seperate log (/var/log/gridftp/gridftp.xxxx.log) for each process (which is normally each new client session). If neither this option or log_single is set, logs will be written to stderr unless the execution mode is detached or inetd, in which case logging will be disabled.
	
	Default value: not set

`log_transfer <string>`
`-Z <string>`
`-log-transfer <string>`
	Log netlogger style info for each transfer into this file.
	
	Default value: not set

`log_filemode <number>` `-log-filemode <number>`	File access permissions of log files. Should be an octal number such as 0644 (the leading 0 is required). Default value: not set
`disable_usage_stats <0|1>` `-disable-usage-stats`	Disable transmission of per-transfer usage statistics. See the Usage Statistics section in the online documentation for more information. Default value: FALSE
`usage_stats_target <string>` `-usage-stats-target <string>`	Comma seperated list of contact strings for usage statistics listeners. Default value: not set

Table 8.5. Single and Striped Remote Data Node Options

`remote_nodes <string>`
`-r <string>`
`-remote-nodes <string>`
	Comma seperated list of remote node contact strings.
	
	Default value: not set

`data_node <0|1>`
`-dn`
`-data-node`
	This server is a backend data node.
	
	Default value: FALSE

`stripe_blocksize <number>`
`-sbs <number>`
`-stripe-blocksize <number>`
	Size in bytes of sequential data that each stripe will transfer.
	
	Default value: 1048576

`stripe_layout <number>`
`-sl <number>`
`-stripe-layout <number>`
	Stripe layout. 1 = Partitioned, 2 = Blocked.
	
	Default value: 2

`stripe_blocksize_locked <0|1>` `-stripe-blocksize-locked`	Do not allow client to override stripe blocksize with the OPTS RETR command Default value: FALSE
`stripe_layout_locked <0|1>` `-stripe-layout-locked`	Do not allow client to override stripe layout with the OPTS RETR command Default value: FALSE

Table 8.6. Network Options

`blocksize <number>`
`-bs <number>`
`-blocksize <number>`
	Size in bytes of data blocks to read from disk before posting to the network.
	
	Default value: 262144

`sync_writes <0|1>` `-sync-writes`	Flush disk writes before sending a restart marker. This attempts to ensure that the range specified in the restart marker has actually been committed to disk. This option will probably impact performance, and may result in different behavior on different storage systems. See the manpage for sync() for more information. Default value: FALSE

Table 8.7. Network Options

`port <number>`
`-p <number>`
`-port <number>`
	Port on which a frontend will listend for client control channel connections, or on which a data node will listen for connections from a frontend. If not set a random port will be chosen and printed via the logging mechanism.
	
	Default value: not set

`control_interface <string>` `-control-interface <string>`	Hostname or IP address of the interface to listen for control connections on. If not set will listen on all interfaces. Default value: not set
`data_interface <string>` `-data-interface <string>`	Hostname or IP address of the interface to use for data connections. If not set will use the current control interface. Default value: not set

`ipc_interface <string>` `-ipc-interface <string>`	Hostname or IP address of the interface to use for ipc connections. If not set will listen on all interfaces. Default value: not set
`hostname <string>` `-hostname <string>`	Effectively sets the above control_interface, data_interface and ipc_interface options. Default value: not set
`ipc_port <number>` `-ipc-port <number>`	Port on which the frontend will listen for data node connections. Default value: not set

Table 8.8. Timeouts

`control_preauth_timeout <number>` `-control-preauth-timeout <number>`	Time in seconds to allow a client to remain connected to the control channel without activity before authenticating. Default value: 120
`control_idle_timeout <number>` `-control-idle-timeout <number>`	Time in seconds to allow a client to remain connected to the control channel without activity. Default value: 600
`ipc_idle_timeout <number>` `-ipc-idle-timeout <number>`	Idle time in seconds before an unused ipc connection will close. Default value: 600
`ipc_connect_timeout <number>` `-ipc-connect-timeout <number>`	Time in seconds before cancelling an attempted ipc connection. Default value: 60

Table 8.9. User Messages

`banner <string>` `-banner <string>`	Message to display to the client before authentication. Default value: not set
`banner_file <string>` `-banner-file <string>`	File to read banner message from. Default value: not set
`banner_terse <0|1>` `-banner-terse`	When this is set, the minimum allowed banner message will be displayed to unauthenticated clients.

	Default value: FALSE
`login_msg <string>` `-login-msg <string>`	Message to display to the client after authentication. Default value: not set
`login_msg_file <string>` `-login-msg-file <string>`	File to read login message from. Default value: not set

Table 8.10. Module Options

`load_dsi_module <string>` `-dsi <string>`	Data Storage Interface module to load. file and remote modules are defined by the server. Defaults to file unless the 'remote' option is specified, in which case the remote DSI is loaded. Default value: file
`allowed_modules <string>` `-allowed-modules <string>`	Comma seperated list of ERET/ESTO modules to allow, and optionally specify an alias for. Example: module1,alias2:module2,module3 (module2 will be loaded when a client asks for alias2). Default value: not set

Table 8.11. Other

`configfile <string>` `-c <string>`	Path to configuration file that should be loaded. Otherwise will attempt to load $GLOBUS_LOCATION/etc/gridftp.conf and /etc/grid-security/gridftp.conf. Default value: not set
`use_home_dirs <0|1>` `-use-home-dirs`	Set the startup directory to the authenticated users home dir. Default value: TRUE
`debug <0|1>` `-debug`	Sets options that make server easier to debug. Not recommended for production servers. Default value: FALSE

Configuring the GridFTP server to run under xinetd/inetd

Note: The service name used (gsiftp in this case) should be defined in `/etc/services` with the desired port.

Here is a sample gridftp server xinetd config entry:

```
service gsiftp
{
instances               = 100
socket_type             = stream
wait                    = no
user                    = root
env                     += GLOBUS_LOCATION=(globus_location)
env                     += LD_LIBRARY_PATH=(globus_location)/lib
server                  = (globus_location)/sbin/globus-gridftp-server
server_args             = -i
log_on_success          += DURATION
nice                    = 10
disable                 = no
}
```

Here is a sample gridftp server inetd config entry: (read as a single line)

```
gsiftp   stream   tcp   nowait   root   /usr/bin/env env   \
    GLOBUS_LOCATION=(globus_location)                     \
    LD_LIBRARY_PATH=(globus_location)/lib                 \
    (globus_location)/sbin/globus-gridftp-server -i
```

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

Introduction

This guide contains advanced configuration information for system administrators working with Java WS Core. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

Building and Installing

Java WS Core is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide [../../admin/docbook/]. No extra installation steps are required for this component.

The following are optional instructions for more advanced types of installations. These are for those advanced users who want to build the latest code from CVS or are just interested in the Java WS Core.

Building from source

1. Obtain the source code for Java WS Core:

   From CVS.

   a. To get the latest source from cvs execute:

   ```
   cvs -d :pserver:anonymous@cvs.globus.org:/home/globdev/CVS/globus-packages \
       checkout wsrf
   ```

   b. Change into the `wsrf` directory.

   ```
   cd wsrf
   ```

   From Core-only source distribution.

   a. Untar or unzip the distribution archive.

   ```
   tar xvfz ws-core-XXX-src.tar.gz
   ```

   b. Change into the unpacked distribution directory.

   ```
   cd ws-core-XXX
   ```

2. Set the `GLOBUS_LOCATION` environment variable to the absolute path of the target directory of your installation. On Windows:

```
set GLOBUS_LOCATION=c:\gt4
```

On Unix/Linux:

```
setenv GLOBUS_LOCATION /soft/gt4/
```

or

```
export GLOBUS_LOCATION=/soft/gt4/
```

If `GLOBUS_LOCATION` is not set, an `install` directory will be created under the current directory.

3. Run:

```
ant all
```

Additional arguments can be specified on the `ant` command line to customize the build:

- `-DwindowsOnly=false` - generate launch scripts for standard Globus tools such as `grid-proxy-init`, etc. (Unix/Linux only)
- `-Dall.scripts=true` - generate Windows and Unix launch scripts
- `-Denable.container.desc` - create and configure the container with a global security descriptor

Installing Core-only binary distribution

1. Untar or unzip the distribution archive.

```
tar xvfz ws-core-XXX-bin.tar.gz
```

2. Change into the unpacked distribution directory.

```
cd ws-core-XXX
```

3. Set the `GLOBUS_LOCATION` environment variable to the unpacked distribution directory. On Windows:

```
set GLOBUS_LOCATION=c:\gt4
```

On Unix/Linux:

```
setenv GLOBUS_LOCATION /soft/gt4/
```

or

```
export GLOBUS_LOCATION=/soft/gt4/
```

Note: Please make sure to have the JAAS [http://java.sun.com/products/jaas/index-10.html] library installed if running with J2SE 1.3.1.

Configuring

Configuration overview

Java WS Core provides per- `gar` configuration and supports configuration profiles. The configuration information of a service is mainly encapsulated in two separate configuration files:

- *server-config.wsdd* (Web Service Deployment Descriptor) - contains information about the web service.
- *jndi-config.xml* (JNDI configuration file) - contains information about the resource management.

A service that support security might also have the *security-config.xml* (security deployment descriptor) file. Please see the Security Descriptor [http://www-unix.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html] page in the GT4 WS Authorization Framework documentation for details.

All these configuration files are dropped into the `$GLOBUS_LOCATION/etc/<gar.id>/` directory during the deployment process.

Syntax of the interface:

Global Configuration

The global properties are specified in the `<globalConfiguration>` section of `*server-config.wsdd` files in the `$GLOBUS_LOCATION/etc/globus_wsrf_core/` directory. The configuration item *name* corresponds to the "*name*" attribute in a `<parameter>` sub element, and the *value* is put as a "*value*" attribute within the same parameter element.

Table 9.1. General configuration parameters

Name	Value	Description	Comments
logicalHost	<hostname>	This parameter specifies the hostname to use instead of the default local host. It is equivalent to setting the `GLOBUS_HOSTNAME` environment property. Can be FQDN or just hostname.	Optional
disableDNS	<boolean>	This parameter specifies whether to perform DNS lookup on the `logicalHost` parameter. By default `"false"` is assumed (DNS lookup is performed).	Optional
domainName	<domanin name>	This parameter specifies the domain name to append to the host name if the host name is not qualified by a domain.	Optional
publishHostName	<boolean>	This parameter specifies whether to publish the hostname or the ip address. It is only used when DNS lookups are enabled (`disableDNS` is false).	Optional

Table 9.2. Standalone/embedded container-specific configuration parameters

Name	Value	Description	Comments
containerThreads	<int>	This parameter controls the initial thread pool size for the container. By default it is set to 5.	Optional
containerThreadsMax	<int>	This parameter sets the maximum number of threads for the container. By default it is set to 4 * the `containerThread` setting.	Optional
containerThreadsHighWaterMark	<int>	This parameter controls when the thread pool of the container should start shrinking (if the number of idle threads exceeds this number). By default it is set to 2 * the `containerThread` setting.	Optional

Service Configuration

WSDD

An example of a deployment descriptor for a CounterService:

```
<service name="CounterService" provider="Handler"
         use="literal" style="document">
 <parameter name="className"
            value="org.globus.wsrf.samples.counter.CounterService"/>
 <parameter name="handlerClass"
            value="org.globus.axis.providers.RPCProvider"/>
 <parameter name="scope"
            value="Application"/>
 <wsdlFile>share/schema/core/samples/counter/counter_service.wsdl</wsdlFile>
 <parameter name="allowedMethodsClass"
            value="com.counter.CounterPortType"/>
 <parameter name="providers" value="
            DestroyProvider SetTerminationTimeProvider GetRPProvider
            SubscribeProvider GetCurrentMessageProvider"/>
</service>
```

Services are defined in a `<service>` element. The "*name*" attribute of the `<service>` element defines the remotely accessible name of the service. The service handle will have the form of <hosting environment URL>/foo, where:

- the hosting environment URL typically is `http://<host>:<port>/wsrf/services.`
- *foo* is the name of the service (`<service name="foo" ...>`).

The *use* attribute should be set to *literal* and the *style* attribute to *document* for all WSRF/WSN based services. The configuration information for a service is defined by various `<parameter>` sub-elements within a `<service>` element. The configuration item *name* corresponds to the "*name*" attribute in a `<parameter>` sub element, and the *value* is put as a "*value*" attribute within the same parameter element.

Table 9.3. Axis Standard Parameters

Name	*Value*	*Description*	*Comments*

className	<class>	This parameter specifies a class that implements the web service methods.	Required
handlerClass	<class>	This parameter specifies what dispatcher to use, to dispatch a request to a service method. This parameter is required if the *provider* attribute of the *service* is set to `Provider`. The default dispatcher we provide is called *org.globus.axis.providers.RPCProvider*. It enables special features such as operation providers or security support.	Recommended in our environment
scope	<value>	Scope value can be one of: *Request* (the default), *Application*, or *Session*. If *Request* scope is used, a new service object is created for each SOAP request that comes in for the service. If *Application* scope is used, only a single instance of the service object is created and used for all SOAP requests that come in for the service. If *Session* scope is used, a new service object is created for each session-enabled client who accesses the service. *Note:* Only *Request* and *Application* scope is supported when used with *org.globus.axis.providers.RPCProvider* handlerClass.	*Application* scope is recommended
wsdlFile	<path>	This parameter points to a wsdl file for the service. The wsdl file must contain the *wsdl:service* entry. The file location can be relative or absolute. A relative file location is recommended.	Required in our environment
allowedMethods	<list of methods>	This parameter specifies a space or comma separated list of method names that can be called via SOAP. `"*"` indicates that all methods of the service class can be invoked via SOAP.	Optional. By default all methods are allowed.

Table 9.4. Java WS Core Parameters

Name	*Value*	*Description*	*Comments*
loadOnStartup	<boolean>	If set to *true* this parameter will cause the web service and the corresponding ResourceHome (if any) to be initialized (with proper security settings if configured) at container startup. This is useful for restarting some tasks, etc. at container startup without having to call the service. Please check the Lifecycle and activation [../../common/javawscore/developer-index.html#Activation] section for details.	Optional
allowedMethodsClass	<class>	This parameter is similar to the *allowedMethods* standard Axis property but it specifies a Java class or an interface that is introspected to come up with a list of allowed methods that can be called remotely on the service. To is useful to easily restrict the SOAP-accessible methods of the service. Usually the class specified in this parameter would be the remote interface class generated for the service. This parameter only has effect if used with *org.globus.axis.providers.RPCProvider* handlerClass.	Optional
providers	<list of providers>	This parameter specifies a space separated list of provider names or class names. Please see operation provider support [../../common/javawscore/developer-index.html#s-javawscore-developer-OperationProvider] section for details. This parameter only has effect if used with *org.globus.axis.providers.RPCProvider* handlerClass.	Optional

Please see Custom Deployment [http://ws.apache.org/axis/java/user-guide.html#PublishingServicesWithAxis] for details on Axis Web Services Deployment Descriptor.

JNDI

An example of a JNDI configuration bit for a CounterService:

```
<service name="CounterService">
  <resource
           name="home"
           type="org.globus.wsrf.samples.counter.CounterHome">
    <resourceParams>
       <parameter>
          <name>factory</name>
          <value>org.globus.wsrf.jndi.BeanFactory</value>
       </parameter>
       <parameter>
          <name>resourceClass</name>
          <value>org.globus.wsrf.samples.counter.PersistentCounter</value>
       </parameter>
       <parameter>
          <name>resourceKeyName</name>
          <value>{http://counter.com}CounterKey</value>
       </parameter>
       <parameter>
          <name>resourceKeyType</name>
          <value>java.lang.Integer</value>
       </parameter>
    </resourceParams>
  </resource>
</service>
```

Each service in WSDD should have a matching entry in the JNDI configuration file with the same name. Under each service entry in JNDI different resource objects or entries might be defined. Please see the JNDI section [../../common/javawscore/developer-index.html#s-javawscore-developer-JNDIDetails] for details. Each service entry in JNDI should have a resource defined called `"home"`. That resource is the `ResourceHome` implementation for the service (as specified by the `type` attribute). Depending on the `ResourceHome` implementation different options can be configured for the `ResourceHome`. Currently we have two main base `ResourceHome` implementations: `org.globus.wsrf.impl.ResourceHomeImpl` and `org.globus.wsrf.impl.ServiceResourceHome`.

Note: All `"home"` resources must specify a `factory` parameter with `org.globus.wsrf.jndi.BeanFactory` value.

ResourceHomeImpl

This implementation is a generic `ResourceHome` implementation. It supports persistent resources, resource caching, resource sweeper, etc.

Table 9.5. ResourceHomeImpl parameters

Name	*Value*	*Description*	*Comments*
resourceKeyName	<qname>	This parameter specifies a QName of the resource key. The namespace is specified in the `{}`. For example, this QName will be used to discover the SOAP header that contains the key of the resource in the request.	Required
resourceKeyType	<class>	This parameter specifies the type of the resource key as	Optional. Defaults to `java.lang.String`

		a Java class. The key XML element is deserialized into this Java type. The Java type can be for any simple Java type, Axis generated bean, or a class with a type mapping.	
resourceClass	<class>	This parameter specifies the classname of the resource object. This is used to ensure that right type of resource object is added to resource home and to instantiate the right object if resource supports persistence.	Required
sweeperDelay	<long>	This parameter specifies how often the resource sweeper runs in milliseconds.	Optional. Defaults to 1 minute
cacheLocation	<jndi path>	This parameter specifies the JNDI location of the resource cache for this resource home. Please see Configuring Resource Cache below for details.	Optional

Configuring Resource Cache

If *ResourceHomeImpl* is configured with resource class that implements the *PersistenceCallback* interface it will store the resource objects wrapped in Java SoftReference [http://java.sun.com/j2se/1.4.2/docs/api/java/lang/ref/SoftReference.html]. That allows the JVM to automatically reclaim these resource objects thus reducing the memory usage. Since the JVM can decide to reclaim these objects at any point sometimes a resource object can be reclaimed between two subsequent invocations on the same resource. This for example can cause the state of the resource to be reloaded from disk on each call. To prevent the JVM from reclaiming the resource objects so quickly a cache can be setup up to hold direct references to these objects. A basic LRU (least recently used) cache implementation is provided. Other cache implementations can be used as long as they implement the *org.globus.wsrf.utils.cache.Cache* interface.

To configure a cache for *ResourceHomeImpl* first define a cache resource entry in JNDI:

```
<resource name="cache"
             type="org.globus.wsrf.utils.cache.LRUCache">
  <resourceParams>
     <parameter>
        <name>factory</name>
        <value>org.globus.wsrf.jndi.BeanFactory</value>
     </parameter>
     <parameter>
        <name>timeout</name>
        <value>120000</value>
     </parameter>
  </resourceParams>
</resource>
```

In this case a LRU cache is configured. The "*timeout*" parameter (in ms) is used to specify the idle time of the resource object before it is removed from the cache. The same cache resource can be reused in different services but usually once cache per service will be configured.

Once the cache resource entry is defined add the "*cacheLocation*" parameter to the service *home* resource. The "*cacheLocation*" parameter value is the JNDI name of the cache resource:

```
<service name="CounterService">
   <resource name="home" type="...">
      <resourceParams>
         ...
         <parameter>
            <name>cacheLocation</name>
            <value>java:comp/env/services/CounterService/cache</value>
         </parameter>
         ...
      </resourceParams>
   </resource>
   ...
   <resource name="cache"
             type="org.globus.wsrf.utils.cache.LRUCache">
   ...
   </resource>
</service>
```

Please note that once the object is removed from the cache it is still up to the JVM to actually reclaim the object.

ServiceResourceHome

This implementation does not accept any special parameters.

Usage Statistics Configuration

Java WS Core container and other GT services are configured to send out usage statistics. Please see the usage statistics section for more information.

The targets to which the usage statistics are sent to are configured via the *`usageStatisticsTargets`* parameter defined in the `<globalConfiguration>` section of the `$GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd` file. The *`usageStatisticsTargets`* parameter specifies a space separated list of targets to which the usage statistics of various components will be sent to. Each target is of form: `host[:port]` (port is optional, if not specified a default port will be assumed). By default usage statistics are sent to `usage-stats.globus.org:4810`.

To disable sending of the usage statistics remove this parameter, comment it out, or remove all of its values.

Configuration Profiles

Configuration profiles allow for the same Java WS Core installation to have multiple configurations. That is, the same installation can be used to run different containers each with different configuration.

When a gar file is deployed [../../common/javawscore/developer-index.html#deployGar], a `-Dprofile` option can be specified to deploy the configuration files under a specific profile name. If the profile name is specified, the deploy operation will drop the configuration file as `$GLOBUS_LOCATION/etc/<gar.id>/<profile.name>-server-config.wsdd` and/or `$GLOBUS_LOCATION/etc/<gar.id>/<profile.name>-jndi-config.xml`. The configuration profiles can also be created by hand simply by copying and/or renaming the configuration files appropriately. Each configuration profile should duplicate the contents of `$GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd` and `$GLOBUS_LOCATION/etc/globus_wsrf_core/jndir-config.xml` in order to make the basic functionality to work properly.

Once a configuration profile is created, the standalone container can be started [../../common/javawscore/rn01re01.html] with a `-profile` option to load configuration files in a specific profile.

Deploying

Recommended JVM settings for the container

It is recommended to increase the maximum heap size of the JVM when running the container. By default on Sun JVMs a 64MB maximum heap size is used. The maximum heap size can be set using the `-Xmx` JVM option. Example:

```
 $ setenv GLOBUS_OPTIONS -Xmx512M
 $ $GLOBUS_LOCATION/bin/globus-start-container
```

The above example will make the container start with maximum heap size set to 512MB.

It is also recommended to experiment with other JVM settings to improve performance. For example, the `-server` option on Sun JVMs enables a server VM which can deliver better performance for server applications.

Deploying into Tomcat

Please note that Tomcat 4.1.x and 5.0.x versions are supported. We recommend running Tomcat with Java 1.4.2+.

Note

Some GT services may not work properly in Tomcat.

To deploy Java WS Core installation into Tomcat run:

```
$ cd $GLOBUS_LOCATION
$ ant -f share/globus_wsrf_common/tomcat/tomcat.xml deploySecureTomcat -Dtomcat.dir=<tomcat.dir>
```

Where *<tomcat.dir>* is an *absolute* path to the Tomcat installation directory.

In addition to the above steps you may have to edit `<tomcat.dir>/webapps/wsrf/WEB-INF/web.xml` if you are running Tomcat on a non-default port, i.e. not using port 8443 (HTTPS). For example, if you run Tomcat on port 443 using HTTPS then the WSRF servlet entry should be modified as follows:

```
<web-app>
...
    <servlet>
        <servlet-name>WSRFServlet</servlet-name>
        <display-name>WSRF Container Servlet</display-name>
        <servlet-class>
            org.globus.wsrf.container.AxisServlet
        </servlet-class>
        <init-param>
            <param-name>defaultProtocol</param-name>
            <param-value>https</param-value>
        </init-param>
        <init-param>
            <param-name>defaultPort</param-name>
            <param-value>443</param-value>
        </init-param>
        <load-on-startup>true</load-on-startup>
    </servlet>
...
</web-app>
```

Please see the Tomcat & Transport Security [../../security/message/admin-index.html#s-message-admin-deploying] documentation for security related Tomcat configuration steps.

Enabling local invocations

To enable local innovations (developer reference [../../common/javawscore/developer-index.html#s-javawscore-developer-LocalInvocations]) in Tomcat you must add `axis-url.jar` to the CLASSPATH before starting Tomcat.

For example on Windows:

```
> cd <tomcat.dir>
> set CLASSPATH=<tomcat.dir>\common\lib\axis-url.jar
> bin\startup
```

On Unix/Linux (csh/tcsh):

```
$ cd <tomcat.dir>
$ setenv CLASSPATH <tomcat.dir>/common/lib/axis-url.jar
$ bin/startup
```

Creating WAR file

To create a `.war` of Java WS Core installation do:

```
$ cd $GLOBUS_LOCATION
$ ant -f share/globus_wsrf_common/tomcat/tomcat.xml war -Dwar.file=<war.file>
```

Where <*war.file*> specifies an *absolute* path of the war file.

Please note that deploying a `war` file might not be enough to have a working Java WS Core deployment. For example, in some cases the `xalan.jar` must be placed in the `endorsed` directory of the container.

Testing

To execute Java WS Core tests first ensure Ant is configured with JUnit (To install JUnit with Ant copy the junit.jar found in JUnit distribution to the $ANT_HOME/lib directory).

To execute the test do the following:

1. Start the standalone container with `-nosec` argument:

   ```
   $ cd $GLOBUS_LOCATION
   $ bin/globus-start-container -nosec
   ```

2. Run the interoperability tests:

   ```
   $ ant -f share/globus_wsrf_test/runtests.xml runServer \
         -Dtests.jar=$GLOBUS_LOCATION/lib/wsrf_test_interop.jar
   ```

3. Run the unit tests:

   ```
   $ ant -f share/globus_wsrf_test/runtests.xml runServer \
         -Dtests.jar=$GLOBUS_LOCATION/lib/wsrf_test_unit.jar -DbasicTestsOnly=true
   ```

Please see the developer guide [../../common/javawscore/developer-index.html#s-javawscore-developer-runningtests] for more information on running the tests and the testing infrastructure.

Security Considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `$GLOBUS_LOCATION/etc/<gar>/` directory) may contain private information such as database passwords, etc. Ensure that these configuration files are only readable by the user that is running the container. The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable

only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Permissions of persistent data

The services using subscription persistence API or other basic persistence helper API will store all or part of its persistent data under the `~/.globus/persisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Invocation of non-public service functions

A client can potentially invoke a service function that is not formally defined in the WSDL but it is defined in the service implementation class. There are two ways to prevent this from happening:

1. Define all service methods in your service class as either `private` or `protected`.
2. Configure appropriate *`allowedMethods`* or *`allowedMethodsClass`* parameter in the service deployment descriptor (please see the section called “Configuring ” for details).

Troubleshooting

`globus-stop-container` fails with an authorization error

By default `globus-stop-container` must be executed with the same credentials as the container is running with. If the *ShutdownService* or the container is configured with separate private key and certificate files (usually `/etc/grid-security/containercert.pem` and `/etc/grid-security/containerkey.pem`) do the following to stop the container:

```
$ grid-proxy-init -cert /etc/grid-security/containercert.pem \
                  -key /etc/grid-security/containerkey.pem \
                  -out containerproxy.pem
$ setenv X509_USER_PROXY containerproxy.pem
$ globus-stop-container
$ unsetenv X509_USER_PROXY
$ rm containerproxy.pem
```

Alternatively, the *ShutdownService* can be configured with a separate gridmap file to allow a set of users to stop the container. Please see the WS Authentication & Authorization [../../security/wsaa.html] section for details.

`globus-start-container` hangs during startup

By default Sun 1.4.x+ JVMs are configured to use `/dev/random` device as an entropy source. Sometimes the machine can run out of entropy and applications (such as the container) using the `/dev/random` device will block until more entropy is available. One workaround for this issue is to configure the JVM to use `/dev/urandom` (non-blocking) device instead. For Sun JVMs a `java.security.egd` system property can be set to configure a different entropy source. To set the system property and pass it to `globus-start-container` script do the following:

```
export GLOBUS_OPTIONS=-Djava.security.egd=file:/dev/urandom
```

or

```
setenv GLOBUS_OPTIONS -Djava.security.egd=file:/dev/urandom
```

Note: This does not apply to Windows machines.

Programs fail with `java.lang.NoClassDefFoundError: javax/security/...`

errors

These errors might occur when running with J2SE 1.3.1 and the JAAS [http://java.sun.com/products/jaas/index-10.html] library is not installed. Either install the JAAS [http://java.sun.com/products/jaas/install_notes.html] library or upgrade to J2SE 1.4.x or higher.

General troubleshooting information

In general, if you want to investigate a problem on your own please see the Debugging and Logging [../../common/javawscore/developer-index.html#s-javawscore-developer-debugging] section for details on how to turn on debugging. Also, please note that most of the command line clients have `-debug` option that will display more detailed error messages including the error stack traces. Also, searching the mailing lists [http://www-fp.globus.org/about/email-archive-search.html] such as discuss@globus.org [mailto:discuss@globus.org] or developer-discuss@globus.org [mailto:developer-discuss@globus.org] (before posting a message) can also be very fruitful. Finally, if you think you have found a bug please report it in our Bugzilla [http://bugzilla.globus.org/bugzilla/] system. Please include as much as detail about the problem as possible.

Usage statistics collection by the Globus Alliance

The following usage statistics are sent by Java WS Core by default in a UDP packet (in addition to the Java WS Core component code, packet version, timestamp, and the source IP address):

- On container startup:
  - container id - random number
  - container type - standalone, servlet, or unknown
  - event type - container startup
  - list of services - service names only
- On container shutdown:
  - container id - random number
  - container type - standalone, servlet, or unknown
  - event type - container shutdown

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

Also, please see our policy statement [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] on the collection of usage statistics.

Chapter 10. GT 4.0 Reliable File Transfer (RFT) Service: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with RFT. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

RFT is used to perform third-party transfers across GridFTP servers. It uses a database to store its state periodically so the transfers can be recovered from any failures. RFT uses standard grid security mechanisms for authorization and authentication of the users. In order to effectively use RFT you should have installed and configured a database with RFT database schemas and have the necessary security infrastructure in place to perform a 3rd party transfer.

Building and Installing

RFT is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide [../../admin/docbook/]. No extra installation steps are required for this component.

The following are specialized instructions for advanced developers who want to deploy latest code from CVS:

Build RFT from CVS:

1. Configure your CVSROOT to point to globus cvs location.
2. Run:

   ```
   cvs co ws-transfer
   ```

3. Run:

   ```
   cd ws-transfer/reliable
   ```

4. Set GLOBUS_LOCATION to point to your globus installation.
5. Run:

   ```
   ant deploy
   ```

Configuring

Required configuration: configuring the PostgreSQL database

PostgreSQL (Version 7.1 or greater) needs to be installed and configured for RFT to work. You can either use the

packages which came with your operating system (RPMs, DEBs, ...) or build from source. We used PostgreSQL version 7.3.2 for our testing and the following instructions are good for the same.

1. Install Postgresql. Instructions on how to install/configure postgresql can be found here [http://www.postgresql.org/docs/manuals/].

2. Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding -o "-i" switch to postmaster script (This can be init.d script found in /etc/init.d/postgresql or /var/lib/ depending on how you installed postgresql). Follow the intstructions here [http://www.postgresql.org/docs/7.4/static/postmaster-start.html] to start the postmaster with -i option.

3. Now you need to set security on the database you are about to create. You can do it by following the steps below:

   `sudo vi /var/lib/pgsql/data/pg_hba.conf` and append the following line to the file:

   ```
   host rftDatabase "username" "host-ip" 255.255.255.255 trust
   ```

   ```
   sudo /etc/init.d/postgresql restart
   ```

4. You will now need to create a postgresql user that would connect to the database. This is usually the account under which the container is running. You can create a postgresql user by running the following command: `su postgres createuser globus` . If you get the following error : `psql: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket "/tmp/.s.PGSQL.5432"?` This generally means that 1. either your postmaster is not started with -i option or 2. you did'nt restart the postmaster after above mentioned step

5. To create the database that is used for RFT, run (as user globus): `createdb rftDatabase`

6. To populate the RFT database with appropriate schemas, run: `psql -d rftDatabase -f $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql` Now that you have created a database to store RFT's state, the following steps configure RFT to find the database:

7. Open `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml`

8. Find the `dbConfiguration` section under `ReliableFileTransferService <service>` section.

9. Change the `connectionString` to point to the machine on which you installed Postgres and name of the database you used in step 2. If you installed Postgres on the same machine as your Globus install, the default should work fine for you.

10. Change the `userName` to the name of the user who owns/created the database and do the same for the password. (It also depends on how you configured your database.)

11. Don't worry about the other parameters in that section. The defaults should work fine for now.

12. Edit the configuration section under `ReliableFileTransferService`. There are two values that can be edited in this section.

13.

    - `backOff` : Time in seconds you want RFT to backoff before a failed transfer is retried by RFT. Default should work fine for now.

    - `maxActiveAllowed`: This is the number of transfers the container can do at given point. Default should be fine for now.

Deploying

RFT is deployed as part of a standard toolkit installation. Please refer to System Adminstrator's Guide [../../admin/docbook/] for details.

Testing

You need to checkout the tests from CVS because RFT tests are not included in the installer. Please follow these steps to run RFT unit tests:

RFT Testing

1. set `$GLOBUS_LOCATION` to point to your Globus install.
2. Start a gridftp server on the machine you are running the tests on default port. This can be done by running:

   ```
   $GLOBUS_LOCATION/sbin/globus-gridftp-server -p 2811 &
   ```
3. Start the container with RFT deployed in it.
4. Edit $GLOBUS_LOCATION/globus_wsrf_rft_test/test.properties. Put in appropriate values for properties like authzValue(self or host), HOST - host ip of container, PORT- port on which container is listening, sourceHost,destinationsHost - hostnames of gridftp servers. The default values will work fine if you are running the tests with a standard stand-alone container.
5. The *.xfr files in $GLOBUS_LOCATION/share/globus_wsrf_rft_test/ are the transfer files that will be used in the tests. Again the default values work fine if you followed the instructions so far.
6. Run the following command which will run all the rft unit tests:

   ```
   ant -Dtests.jar=$GLOBUS_LOCATION/lib/globus_wsrf_rft_test.jar -f share/globus_wsrf_rft_test/runte
   ```
7. Run the following command to generate the test-reports in html form:

   ```
   ant -f  share/globus_wsrf_rft_test/runtests.xml generateTestReport
   ```

Security Considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `etc/<gar>/` directory) contains private information such as database passwords and username. Ensure that these configuration files are only readable by the user that is running the container.

The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Access of information stored in the database

RFT stores the transfer request in a database. Proper security measures need to be taken to protect the access of the data by granting/revoking appropriate permissions on tables that are created for RFT use and other steps that are appropriate and consistent with site specific security measures.

Permissions of persistent data

RFT uses subscription persistence API from GT4 core to store all of its subscription data under the `~/.globus/persisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Troubleshooting

PostgreSQL not configured

Problem: If RFT is not configured properly to talk to a PostgreSQL database, you will see this message displayed on the console when you start the container :

```
"Error creating RFT Home: Failed to connect to database ...
Until this is corrected all RFT request will fail and all GRAM jobs that require staging will fail".
```

Solution: Usual mistake is Postmaster is not accepting TCP connections which means that you must restart Postmaster with -i option (see the section called “Required configuration: configuring the PostgreSQL database”).

More verbose error messages

Problem: Make RFT print more verbose error messages:

Solution: Edit $GLOBUS_LOCATION/container-log4j.properties and add following line to it: `log4j.category.org.globus.transfer=DEBUG` . For more verbosity add `log4j.category.org.globus.ftp=DEBUG` which will print out Gridftp messages too.

RFT fault-tolerance and recovery

RFT uses PostgreSQL to check-point transfer state in the form of restart markers and recover from transient transfer failures, using retry mechanism with exponential backoff, during a transfer. RFT has been tested to recover from source and/or destination server crashes during a transfer, network failures, container failures (when the machine running the container goes down), file system failures, etc. RFT Resource is implemented as a PersistentResource, so ReliableFileTransferHome gets initialized every time a container gets restarted. Please find more detailed description of fault-tolerance and recovery in RFT below:

- Source and/or destination GridFTP failures: In this case RFT retries the transfer for a configurable number of maximum attempts with exponential backoff for each retry (the backoff time period is configurable, also). If a failure happens in the midst of a transfer, RFT uses the last restart marker that is stored in the database for that transfer and uses it to resume the transfer from the point where it failed instead of restarting the whole file again. This failure is treated as a container-wide backoff for the server in question. What that means is that all other transfers going to/from that server, across all the requests in a container, will be backed off and retried. This is done in order to prevent further failures of the transfers by using knowledge available in the database.

- Network failures: Sometimes this happens due to heavy load on a network or for any other reason packets are lost or connections get timed out. This failure is considered a transient failure and RFT retries the transfer with exponential backoff for that particular transfer (and not the whole container as with the source and/or destination GridFTP failures).

- Container failures: These type of failures occur when the machine running the container goes down or if the container is restarted with active transfers. When the container is restarted, it restarts ReliableTransferHome which looks at the database for any active RFT resources and restarts them.

Failure modes that are not addressed:

- Running out of disk space for the database.

Usage statistics collection by the Globus Alliance

The following usage statistics are sent by default in a UDP packet at the end of life time of each RFT Resource (or when a RFT resource is destroyed).

- Total number of files transferred by RFT since RFT is installed
- Total number of bytes transferred by RFT since RFT is installed
- Total number of files transferred in this RFT Resource
- Total number of bytes transferred in this RFT Resource
- Creation time of this RFT Resource
- Factory Start Time

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the GRAM component. Nevertheless, if you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration [http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets] for instructions.

Also, please see our policy statement [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] on the collection of usage statistics.

Chapter 11. GT 4.0 WS GRAM : System Administrator's Guide

Introduction

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

Important

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

Building and Installing

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

Installation Requirements

Transport Level Security (TLS)

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

Functioning sudo

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

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

Platform Note: On AIX, sudo is not installed by default, but it is available as source and rpm here: AIX 5L Toolbox for Linux Applications [http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html]

Local scheduler

WS GRAM depends on a local mechanism for starting and controlling jobs. Included in the WS GRAM software is a Fork "scheduler", which requires no special software installed to execute jobs on the local host. However, to enable WS GRAM to execute and manage jobs to a batch scheduler, the scheduler software must be installed and configured prior to configuring WS GRAM. The WS GRAM scheduler adapters (interface to the batch schedulers) included in the GT 4.0 release are: PBS [http://www.openpbs.org/], Condor [http://www.cs.wisc.edu/condor/], LSF [http://www.platform.com/products/LSF/] Scheduler adapters that we have For configuration details, see the Configuring scheduler adapters section.

Other scheduler adapters that we know of are: Sun Grid Engine [http://www.lesc.ic.ac.uk/projects/SGE-GT4.html]

Reliable File Transfer Service (RFT)

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

Configuring

Typical Configuration

Configuring sudo

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

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

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

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

Configuring Scheduler Adapters

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

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

% make gt4-gram-pbs

% make install
```

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

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

```
% cd $GLOBUS_LOCATION/setup/globus

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

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

- files staging is performed correctly
-

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

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

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

Non-default Configuration

Non-default Credentials

To run the container using just a user proxy, instead of host creds, simply comment out the ContainerSecDesc parameter in this file `$GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd` as follows:

```
<!--
    <parameter
        name="containerSecDesc"
        value="etc/globus_wsrf_core/global_security_descriptor.xml"/>
 -->
```

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

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

You can get your subject DN by running this command:

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

Non-default GridFTP server

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

Non-default container port

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

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

When doing this, GRAM needs to be told the port to use to contact the RFT service, like so:

```
% cd $GLOBUS_LOCATION/setup/globus
% ./setup-gram-service-common --staging-port="4321"
```

Non-default gridmap

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

changed:

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

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

Example: *global_security_descriptor.xml*

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

sudoers

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

Non-default job resource limit

The current limit on the number of job resources (both exec and multi) allowed to exist at any one time is 1000. This limit was chosen from scalability tests as an appropriate precaution to avoid out-of-memory errors. To change this value to, say, 150, use the setup-gram-service-common script as follows:

```
% cd $GLOBUS_LOCATION/setup/globus
% ./setup-gram-service-common --max-job-limit="150"
```

Deploying

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

Testing

See the WS GRAM User's Guide [../../execution/wsgram/user-index.html#s-wsgram-user-usagescenarios] for information about submitting a test job.

Security Considerations

No special security considerations exist at this time.

Troubleshooting

The job manager detected an invalid script response

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

Usage statistics collection by the Globus Alliance

The following usage statistics are sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at the end of each job (i.e. when Done or Failed state is entered).

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

If you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration [http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets] for instructions.

Also, please see our policy statement [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] on the collection of usage statistics.

Configuration interface

Locating configuration files

All the GRAM service configuration files are located in subdirectories of the `$GLOBUS_LOCATION/etc` directory.

The names of the GRAM configuration directories all start with `gram-service`. For instance, with a default GRAM installation, the command line:

```
% ls etc | grep gram-service
```
gives the following output:

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

Web service deployment configuration

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

Three GRAM services are deployed:

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

Each service deployment information contains the name of the Java service implementation class, the path to the WSDL service file, the name of the operation providers that the service reuses for its implementation of WSDL-defined operations, etc. More information about the service deployment configuration information can be found here [http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/Java_WS_Core_Public_Interfaces.html#config].

JNDI application configuration

The configuration of WSRF resources and application-level service configuration not related to service deployment is contained in JNDI [http://java.sun.com/products/jndi/] files. The JNDI-based GRAM configuration is of two kinds:

Common job factory configuration

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

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

The data is categorized by service, because according to WSRF, in spite of the service/resource separation of concern, a given service will use only one XML Schema type of resource. In practice it is therefore clearer to categorize the configuration resource implementation by service, even if theoretically speaking a given resource implementation could be used by several services. For more information, refer to the Java WS Core documentation [http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/index.html].

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

- ManagedExecutableJobService

    - ManagedExecutableJobHome: configuration of the implementation of resources for the service.

- ManagedMultiJobService

    - ManagedMultiJobHome: configuration of the implementation of resources for the service

- ManagedJobFactoryService

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

Local resource manager configuration

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

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

For instance, let's assume the command line:

```
% ls etc | grep gram-service-
```
gives the following output:

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

In this example, the Multi, Fork and LSF job factory resources have been installed. `Multi` is a special kind of local resource manager which enables the GRAM services to support multijobs [http://www-unix.globus.org/toolkit/docs/4.0/execution/wsgram/user/managed-job-globusrun.html#multijobs].

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

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

```
    <service name="ManagedJobFactoryService">
        <!-- LRM configuration:  Fork -->
        <resource
            name="ForkResourceConfiguration"
            type="org.globus.exec.service.factory.FactoryResourceConfiguration">
            <resourceParams>
```

```
            [...]
            <parameter>
                <name>
                    localResourceManagerName
                </name>
                <value>
                    Fork
                </value>
            </parameter>
            <parameter>
                <name>
                    scratchDirectory
                </name>
                <value>
                    ${GLOBUS_USER_HOME}
                </value>
            </parameter>
        </resourceParams>
    </resource>
</service>
```

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

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

Security descriptor

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

- default security information for all remote invocations, such as:
  - the authorization method, based on a Gridmap file (in order to resolve user credentials to local user names)
  - limited proxy credentials will be rejected
- security information for the `createManagedJob` operation

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

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

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

GRAM and GridFTP file system mapping

The file `$GLOBUS_LOCATION/etc/gram-service/globus_gram_fs_map_config.xml` contains information to associate local resource managers with GridFTP servers. GRAM uses the GridFTP server (via RFT) to perform all file staging directives. Since the GridFTP server and the Globus service container can be run on separate hosts, a

mapping is needed between the common file system paths of these 2 hosts. This enables the GRAM services to resolve file:/// staging directives to the local GridFTP URLs.

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

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

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

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

The file system mapping schema doc is here [http://www-unix.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_fs_map.html].

Scheduler-Specific Configuration Files

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

Table 11.1. Scheduler-Specific Configuration Files

$GLOBUS_LOCATION/etc/globus-fork.conf		
	Configuration for the Fork SEG module implementation. The attributes names for this file are:	
	log_path	Path to the SEG Fork log (used by the globus-fork-starter and the SEG). The value of this should be the path to a world-writable file. The default value for this created by the Fork setup package is $GLOBUS_LOCATION/var/globus-

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

Chapter 12. GT 4.0 GSI-OpenSSH: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with GSI-OpenSSH. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

This guide is meant solely to cover the GSI aspects of GSI-OpenSSH, and is not meant to be a full manual for OpenSSH itself. Please refer to the OpenSSH Home Page [http://www.openssh.org/] for general documentation for OpenSSH.

Building and Installing

GSI-OpenSSH is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide [../../../admin/docbook/]. No extra installation steps are required for this component.

You can optionally pass build-time configure options to the GSI-OpenSSH package using the --with-gsiopensshargs option when running configure for your GT 4.0 installation. For example:

```
./configure --prefix=$HOME/globus
      --with-gsiopensshargs="--with-pam"
```

No options are typically needed for client-only installations, but options are often needed for full server functionality. The following table lists suggested options for different platforms.

Table 12.1. GSI-OpenSSH build arguments

Platform	Configuration
Linux	--with-pam --with-md5-passwords --with-tcp-wrappers
Solaris	--with-pam --with-md5-passwords --with-tcp-wrappers
Irix	--with-tcp-wrappers
AIX	--with-tcp-wrappers

Note: If you enable PAM support with the --with-pam configuration option, be sure to also set "UsePAM yes" in $GLOBUS_LOCATION/etc/ssh/sshd_config after installation.

If you have an already configured and installed system-wide SSHD and you would like your build of GSI-OpenSSH to behave similarly, investigate the configure options available in GSI-OpenSSH and select those options that would add the functionality that your current SSHD possesses. Be aware that since GSI-OpenSSH is based on OpenSSH, the standard set of functionality is turned on by default.

Please do not attempt to override the following options:

```
--prefix
--sysconfdir
--with-globus
--with-globus-flavor
--with-ssl-dir
```

Configuring

The GSI-enabled OpenSSH software is installed with a default set of configuration files, described below. You may want to modify the `ssh_config` file before using the clients and the `sshd_config` file before using the server.

If the GSI-enabled OpenSSH install script found existing SSH key pairs, it will create symbolic links to them rather than generating new key pairs. The SSH key pairs are not required for GSI authentication. However, if you wish to support other SSH authentication methods, make sure the sshd (running as root) can read the key pair files (i.e., beware of NFS mounts with root_squash). If running multiple sshds on a system, we recommend configuring them so they all use the same key pairs (i.e., use symbolic links) to avoid client-side confusion.

- `$GLOBUS_LOCATION/etc/ssh/moduli`

  moduli is some crypto parameter for generating keys.

- `$GLOBUS_LOCATION/etc/ssh/ssh_config`

  ssh_config contains options that are read by ssh, scp, and sftp at run-time. The installed version is the default provided by OpenSSH, with X11Forwarding enabled. You may need to customize this file for compatibility with your system SSH installation (i.e., compare with /etc/ssh/ssh_config).

- `$GLOBUS_LOCATION/etc/ssh/ssh_host_key[.pub]`

  Your system's RSA public-/private-key pair for SSH protocol 1 communications.

- `$GLOBUS_LOCATION/etc/ssh/ssh_host_dsa[.pub]`

  Your system's DSA public-/private-key pair for SSH protocol 2 communications.

- `$GLOBUS_LOCATION/etc/ssh/ssh_host_rsa[.pub]`

  Your system's RSA public-/private-key pair for SSH protocol 2 communications.

- `$GLOBUS_LOCATION/etc/ssh/ssh_prng_cmds`

  ssh_prng_cmds contains paths to a number of files that ssh-keygen may need to use if your system does not have a built-in entropy pool (like /dev/random).

- `$GLOBUS_LOCATION/etc/ssh/sshd_config`

  sshd_config contains options that are read by sshd when it starts up. The installed version is the default provided by OpenSSH, with X11Forwarding enabled. You may need to customize this file for compatibility with your system SSH installation (i.e., compare with /etc/ssh/sshd_config). For example, to enable PAM authentication, you will need to set "UsePAM yes" in this file.

Deploying

1. To install the GSI-Enabled OpenSSH Server on most systems, you must be a privileged user, such as root.

   ```
   sh$ /bin/su - root
   ```
   Note: If your system functions like this and you attempt to run these commands as a user other than root, these commands should fail.

2. (optional) Start a copy of your system's currently running SSH server on an alternate port by running, eg.

   ```
   sh# /usr/sbin/sshd -p 2000 &
   ```
   You may then choose to log in to this server and continue the rest of these steps from that shell. We recommend doing this since some sshd shutdown scripts do particularly nasty things like killing *all* of the running SSH servers on a system, not just the parent server that may be listening on port 22. Roughly translated, this step is about guaranteeing that an alternate method of access is available should the main SSH server be shutdown and your connection via that server be terminated.

3. Locate your server's startup/shutdown script directory. On some systems this directory may be located at /etc/rc.d/init.d, but since this location is not constant across operating systems, for the purposes of this document we will refer to this directory as INITDIR. Consult your operating system's documentation for your system's location.

4. Run the following command

   ```
   sh# mv $INITDIR/sshd $INITDIR/sshd.bak
   ```

5. Either copy or link the new sshd script to your system's startup/shutdown script directory.

   ```
   sh# cp $GLOBUS_LOCATION/sbin/SXXsshd $INITDIR/sshd
   ```

6. Shutdown the currently running main SSH server.

   ```
   sh# $INITDIR/sshd.bak stop
   ```

7. Provided you still have a connection to the machine, start the new SSH server.

   ```
   sh# $INITDIR/sshd start
   ```

8. Test the new server by connecting to the standard SSH port (22) and authenticating via multiple methods. Especially test that GSI authentication works correctly.

9. If you are performing a new install, or if the old server was not configured to be started at run-time and shutdown automatically at system halt or reboot, either use a system utility such as RedHat's chkconfig to configure the system for the correct run-levels, or manually link up the correct run-levels.

   ```
   sh# /sbin/chkconfig sshd reset
   ```
   The recommended run-levels are listed in a set of comments within the SXXsshd startup script. For example, on standard Unix systems we recommend running the GSI-Enabled OpenSSH server in run-levels two, three, four, and five.

10. Finally, if, as a precautionary measure, you started a SSH server on an alternate port in order to complete the install process, you can now safely stop all instances of that server.

Testing

1. Edit the file $GLOBUS_LOCATION/sbin/SXXsshd so that the GSI-Enabled OpenSSH server starts up on an alternate port.

2. Run the command

   ```
   sh# $GLOBUS_LOCATION/sbin/SXXsshd start
   ```
   and verify that the server is running by checking that it both shows up in a process listing and creates a file named $GLOBUS_LOCATION/var/sshd.pid.

3. From a remote machine attempt to connect to the local server on the modified test port using the standard SSH authentication methods plus authenticating via your GSI credentials. This may require you to authorize these users via an appropriate entry in the grid-mapfile.

4. Stop the SSH server by running the command

   ```
   sh# $GLOBUS_LOCATION/sbin/SXXsshd stop
   ```
   and reverse any changes you made that altered the port on which the server resided upon startup. After this step, running SXXsshd start should start the server on the default port (22).

Security Considerations

GSI-OpenSSH is a modified version of OpenSSH [http://www.openssh.org/] and includes full OpenSSH functionality. For more information on OpenSSH security, see the OpenSSH Security [http://www.openssh.org/security.html] page.

Troubleshooting

GSI authentication is very sensitive to clock skew. You must run a system clock synchronization service of some type on your system to prevent authentication problems caused by incorrect system clocks. We recommend NTP [http://www.ntp.org/]. Please refer to your operating system documentation or the NTP Home Page [http://www.ntp.org/] for installation instructions. Please also ensure your system timezone is set correctly.

Chapter 13. GT 4.0 MyProxy: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with MyProxy. 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

A typical MyProxy configuration has one dedicated myproxy-server for the site, with MyProxy clients installed on all systems where other Globus Toolkit client software is installed.

Building and Installing

MyProxy is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide [../../../admin/docbook/]. No extra installation steps are required for this component.

Configuring

No additional configuration is required to use MyProxy clients after they are installed, although you may want to set the MYPROXY_SERVER environment variable to the hostname of your myproxy-server in the default user environment on your systems.

To configure the myproxy-server, you must modify $GLOBUS_LOCATION/etc/myproxy-server.config. *If you skip this step, your myproxy-server will not accept any requests.* The default configuration does not enable any myproxy-server features to provide the greatest security until you have configured your server. To enable all myproxy-server features, uncomment the provided sample policy at the top of the myproxy-server.config config file, as follows:

```
#
# Complete Sample Policy
#
# The following lines define a sample policy that enables all
# myproxy-server features.  See below for more examples.
accepted_credentials  "*"
authorized_retrievers "*"
default_retrievers    "*"
authorized_renewers   "*"
default_renewers      "none"
```

Please see below for additional documentation on the myproxy-server.config options.

If you have root access, you can copy your `myproxy-server.config` file to `/etc/myproxy-server.config` so it is not overwritten by later installations.

The `myproxy-server.config` file sets the policy for the **myproxy-server(8)**, specifying what credentials may be stored in the server's repository and who is authorized to retrieve credentials. By default, the **myproxy-server(8)** looks for this file in `/etc/myproxy-server.config` and if it is not found there, it looks in `$GLOBUS_LOCATION/etc/myproxy-server.config`. The **myproxy-server -c** option can be used to specify an

alternative location. The file installed by default does not allow any requests.

The file also supports a **passphrase_policy_program** command for specifying an external program for evaluating the quality of users' passphrases. A sample program is installed in **$GLOBUS_LOCATION/share/myproxy/myproxy-passphrase-policy** but is not enabled by default.

Lines in the configuration file use limited regular expressions for matching the distinguished names (DNs) of classes of users. The lim- ited regular expressions support the shell-stype characters '*' and '?', where '*' matches any number of characters and '?' matches any single character.

The DN limited regexes should be delimited with double quotes ("DN regex").

The configuration file has the following types of lines:

Table 13.1. myproxy-server.config lines

accepted_credentials "DNregex"	Each of these lines allows any clients whose DNs match the given limited regex to connect to the myproxy-server and store creden- tials with it for future retrieval. Any number of these lines may appear. For backwards compatibility, these lines can also start with `allowed_clients` instead of `accepted_credentials`.
authorized_retrievers "DN regex"	Each of these lines allows the server administrator to set server-wide policies for authorized retrievers. If the client DN does not match the given limited regex the client is not allowed to retrieve the credentials previously stored by a client. In addition to the server-wide policy, MyProxy also provides support for per-credential policy. The user can specify the regex DN of the allowed retrievers of the credential when uploading the credential (using **myproxy-init(1)**). The retrieval client DN must also match the user specified regex. In order to retrieve credentials the client also needs to know the name and pass phrase provided by the client when the credentials were stored. Any number of these lines may appear. For backwards compatibility, these lines can also start with `allowed_services` instead of `authorized_retrievers`.
default_retrievers "DN regex"	Each of these lines allows the server administrator to set server-wide default policies. The regex specifies the clients who can access the credentials. The default retriever policy is enforced if a per-credential policy is not specified on upload (using **myproxy-init(1)**). In other words, the client can override this policy for a credential on upload. The per-credential policy is enforced in addition to the server-wide policy specified by the `authorized_retrievers` line (which clients can not override). Any number of these lines may be present. For backwards compatibility, if no `default_retrievers` line is specified, the default policy is "*", which allows any client to pass the per-credential policy check. (The client must still pass the `authorized_retrievers` check.)
authorized_renewers "DN regex"	Each of these lines allows the server administrator to set server-wide policies for authorized renewers. If the client DN does not match the given limited regex the client is not allowed to renew the credentials previously stored by

	a client. In addition to the server-wide policy, MyProxy also provides support for per-credential policy. The user can specify the regex DN of the allowed renewers of the credential on upload (using **myproxy-init(1)**). The renewal client DN must match both this regex and the user specified regex. In this case, the client must also already have a credential with a DN matching the DN of the credentials to be retrieved, to be used in a second authorization step (see the `-a` option for **myproxy-logon(1)**).
default_renewers "DN regex"	Each of these lines allows the server administrator to set server-wide default renewer policies. The regex specifies the clients who can renew the credentials. The default renewer policy is enforced if a per-credential policy is not specified on upload (using **myproxy-init(1)**). This is enforced in addition to the server-wide policy specified by the `authorized_renewers` line. Any number of these lines may appear. For backwards compatibility, if no `default_renewers` line is specified, the default policy is "*", which allows any client to pass the per-credential policy check. (The client must still pass the `authorized_renewers` check.)
passphrase_policy_program full-path-to-script	This line specifies a program to run whenever a passphrase is set or changed for implementing a local password policy. The program is passed the new passphrase via stdin and is passed the following arguments: username, distinguished name, credential name (if any), per-credential retriever policy (if any), and per-credential renewal policy (if any). If the passphrase is acceptable, the program should exit with status 0. Otherwise, it should exit with non-zero status, causing the operation in progress (credential load, passphrase change) to fail with the error message provided by the program's stdout. Note: You must specify the full path to the external program. `$GLOBUS_LOCATION` can't be used in the `myproxy-server.config` file.
max_proxy_lifetime hours	This line specifies a server-wide maximum lifetime for retrieved proxy credentials. By default, no server-wide maximum is enforced. However, if this option is specified, the server will limit the lifetime of any retrieved proxy credentials to the value given.

Deploying

A sample SysV-style boot script for MyProxy is installed at $GLOBUS_LOCATION/share/myproxy/etc.init.d.myproxy. To install on Linux, copy the file to /etc/rc.d/init.d/myproxy and run 'chkconfig --add myproxy'. You will need to edit the file to set the GLOBUS_LOCATION environment variable correctly.

Alternatively, to run the myproxy server out of inetd or xinetd, you need to do the following as root:

- Add the entries in $GLOBUS_LOCATION/share/myproxy/etc.services.modifications to the /etc/services or /etc/inet/services file.
- Add the entries in $GLOBUS_LOCATION/share/myproxy/etc.inetd.conf.modifications to /etc/inetd.conf or /etc/inet/inetd.conf, or copy $GLOBUS_LOCATION/share/myproxy/etc.xinetd.myproxy to /

etc/xinetd.d/myproxy. You'll need to modify the paths in the file according to your installation.

- Reactivate the inetd (or xinetd). This is typically accomplished by sending the SIGHUP signal to the daemon. Refer to the inetd or xinetd man page for your system.

Testing

To verify your myproxy-server installation and configuration, you can run the myproxy-server directly from your shell. If using a host certificate, you will need to run the myproxy-server as root. First, make sure your Globus environment is setup in your shell. Set the GLOBUS_LOCATION environment variable to the location of your MyProxy installation. Then, depending on your shell, run one of the following commands.

For csh shells:

source $GLOBUS_LOCATION/etc/globus-user-env.csh

For sh shells:

. $GLOBUS_LOCATION/etc/globus-user-env.sh

Then, run **$GLOBUS_LOCATION/sbin/myproxy-server -d**. The `-d` argument runs the myproxy-server in debug mode. It will write debugging messages to the terminal and exit after servicing a single request. You'll need to start it once for each test request. In another shell, you can run the MyProxy client programs to test the server.

If run without the `-d` argument, the myproxy-server program will start up and background itself. It accepts connections on TCP port 7512, forking off a separate child to handle each incoming connection. It logs information via the syslog service under the daemon facility.

Security Considerations

You should choose a well-protected host to run the myproxy-server on. Consult with security-aware personnel at your site. You want a host that is secured to the level of a Kerberos KDC, that has limited user access, runs limited services, and is well monitored and maintained in terms of security patches.

For a typical myproxy-server installation, the host on which the myproxy-server is running must have /etc/grid-security created and a host certificate installed. In this case, the myproxy-server will run as root so it can access the host certificate and key.

Troubleshooting

Please refer to the MyProxy user manual [../../security/myproxy/user-index.html#s-myproxy-user-troubleshooting].

Chapter 14. GT4 CAS Admin Guide

Introduction

This guide contains advanced configuration information for system administrators working with the Community Authorization Service (CAS). It provides references to information on procedures typically performed by system administrators, including installing, 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

Note: Typically a single CAS server is run per VO and multiple client installations are done. This document contains information about deploying a CAS server and is not needed for a CAS client installation. Please refer client install [#casClientInstall] for CAS client install

Building and Installing

The CAS server and client are built and installed as part of a default GT 4.0 installation. For basic installation instructions, refer to the GT 4.0 System Administrator's Guide [../../../admin/docbook/]. No extra installation steps are required for this component.

The CAS client can be installed by itself. Please refer to *FILL ME(link to top level install guide for installing a given package)*

Configuring

Configuration overview

The CAS service can be configured with a description of the VO the CAS service serves and the maximum lifetime of the assertions it can issue. Also, the service needs to be configured with information about the backend database it uses. Any database with a JDBC driver and reasonable SQL support can be used. That said, PostGres was used for development and testing and we strongly recommend that you use it. The CAS database schema to be used with PoseGres has been provided in*$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema/cas_pgsql_database_schema.sql*.

Other than that, the security settings of the service can be modified in the security descriptor associated with the CAS service. It allows for configuring the credentials that will be used by the service, the type of authentication and message protection required as well as the authorization mechanism. By default, the following security configuration is installed:

- Credentials are determined by the container level security descriptor. If there is no container level security descriptor or if it does not specify what credentials to use then default credentials are used.
- Authentication and message integrity protection is enforced for all methods except *queryResourceProperties* and *getResourceProperty*. This means that you may use any of GSI Transport, GSI Secure Message or GSI Secure Conversation when interacting with the CAS service.
- The standard authorization framework is not used for authorization. Instead the the service uses the backend database to determine if the call is permitted.

Note

Changing required authentication and authorization methods will require matching changes to the clients that contact this service.

Syntax of the interface

- By default, the CAS service is not loaded at start up. To change the behavior uncomment the *loadOnStartup* property set in *$GLOBUS_LOCATION/etc/globus_cas_service/server-config.wsdd* as shown below. If the property is uncommented, a) the CAS service is loaded at start up, b) database connection pool is initialized and c) if registration to MDS is enabled, the service registers itself to MDS/Index ?

```
<service name="CASService" provider="Handler" use="literal"
        style="document">
<!-- Uncomment if the service needs to be initialized at startup -->
        <parameter name="loadOnStartup" value="true"/>
        <parameter name="allowedMethodsClass"
            value="org.globus.cas.CASPortType"/>
         .
         .
         .
    </service>
```

- To change the maximum assertion lifetime and VO description, set the parameters `maxAssertionLifetime` and `voDescription` in `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` to the desired values.
- To alter the configuration of the database backend edit the `databseConfiguration` section of `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` as follows:

Table 14.1. Database parameters

driver	The JDBC driver to be used
connectionURL	The JDBC connection url to be used when connecting to the database
userName	The user name to connect to the database as
password	The corresponding database password
activeConnections	The maximum number of active connections at any given instance
onExhaustAction	The action to perform when the connection pool is exhausted. If value is 0 then fail, if 1 then block and if 2 then grow the pool (get more connections)
maxWait	The maximum time in milliseconds that the pool will wait for a connection to be returned
idleConnections	The maximum number of idle connections at any given time

- To alter the security descriptor configuration refer to Security Descriptors [http://www-unix.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html]. The file to be altered is `$GLOBUS_LOCATION/etc/globus_cas_service/security-config.xml`

Deploying

The CAS service is deployed as a part of a standard toolkit installtion. Please refer to the System Administrator's Guide [../../../admin] for details. Other than steps described in the above guide, the following are needed to deploy the CAS service.

Obtaining credentials for the CAS service

The CAS service can run with its own service specific credentials. Instructions for obtaining service credentials may be found here [http://www.globus.org/gt2.2/admin/guide-verify.html#ldapcert].

The standard administrator clients that come with the distribution by default use identity authorization to authorize the service they are running against (and expect that the CAS service has credentials that have the FQDN of the host the server is running on and the service name "cas" as part of DN). Command line options can be used to specify the identity of the CAS service, if default identity is not used. The command in the above mentioned web page [http://www.globus.org/gt2.2/admin/guide-verify.html#ldapcert] may be altered as follows to get credentials for CAS server:

```
casadmin$ grid-cert-request -service cas -host FQDN
```

The certificate and private key are typically placed in `/etc/grid-security/cas-cert.pem` and `/etc/grid-security/cas-key.pem`, respectively. In this document, the location of certificate and key files is referred to as CAS_CERT_FILE and CAS_KEY_FILE, respectively. (The subject name in these credentials is expected by CAS clients by default.)

Database installation and configuration

CAS uses a backend database to store all user data. This section briefly describes the installation of such a database and the creation of the database using the schema required by the CAS backend.

Installing the database

While any database with a JDBC driver and support for a reasonable set of SQL may be used, PostgreSQL has been used for development and testing. The driver for the same is included in the distribution. If a different database is used, the corresponding driver should be added to $GLOBUS_LOCATION/lib.

Brief instructions on how to install a database (specifically PostGres) can be found here [http://www-unix.globus.org/toolkit/3.0/ogsa/docs/admin/installation.html]. For more detailed instructions, please refer to documentation for the database you are installing.

Creating the CAS database

The schema for the database that needs to be created for CAS can be found at `$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema/cas_pgsql_database_schema.sql`

To create a database, for example *casDatabase*, on a PostgreSQL installation on local machine run the following commands:

```
casadmin$ createdb casDatabase

casadmin$ psql -U casadmin -d casDatabase -f  \
```

```
$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema/cas_pgsql_database_schema.sql
```

You will see a list of notices on the screen. Unless any of them say "ERROR", these are just informational output.

Bootstraping the CAS database

The CAS database needs to be initialized with data specific to CAS and information about a super user to allow bootstraping of CAS operations. The command line script *cas-server-bootstrap* can be used to achieve this.

```
cas-server-bootstrap [<options>] -d <dbPropFile> [ -implicit | -b <bootstrapFile> ]
```

Table 14.2. Command line options

`-help`	Prints the help message
`-debug`	Runs the script with debug trace
`-d` *`dbPropertiesFile`*	File name with database properties as follows: `dbDriver=`*`database driver name`* `dbConnectionURL=`*`database connection URL`* `dbUserName=`*`Username to access database`* `dbPassword=`*`Password for the above username`*
`-b` *`bootstrapFile`*	This option populates the database with super user data and points to a file with data to use for bootstraping the database. A template file for this can be found at `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapTemplate` and a sample file can be found at `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapSample`
`-implicit`	Populates database with a) CAS server implicit data: this adds the CAS server itself as a CAS object and implicit service/actions like enrolling users, objects and so on. b) service/action and namespace relevant to FTP like read, write and so on

Sample bootstrap command:

To bootstrap the CAS database with both implicit and user data the following command can be used. Prior to running the command, the following files need to be created with appropriate values filled in.

- `$GLOBUS_LOCATION/share/globus_cas_service/casDbProperties`

  ```
  dbDriver=org.postgresql.Driver
  dbConnectionURL=jdbc:postgresql://127.0.0.1/casDatabase
  dbUsername=tester
  dbPassword=foobar
  ```

- `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapProperties`

  ```
  ta-name=defaultTrustAnchor
  ta-authMethod=X509
  ```

```
ta-authData=/C=US/O=Globus/CN=Default CA
user-name=superUser
user-subject=/O=Grid/O=Globus/OU=something/CN=someone
userGroupname=superUserGroup
```

- Command to run

```
casadmin$ cd $GLOBUS_LOCATION
casadmin$ bin/cas-server-bootstrap \
 -d share/globus_cas_service/casDbProperties \
-implicit -b \ share/globus_cas_service/bootstrapProperties
```

Once the database has been created the CAS service needs to be configured to use it as described here [./../WS_AA_CAS_Public_Interfaces.html#s-cas-public-config].

Testing

CAS has two sets of tests, one for the backend database access module and another set to test the service itself. To install both tests, install the CAS test package (*gt4-cas-delegation-test-3.9-src_bundle.tar.gz*) using GPT *FILLME: instructions* into *GLOBUS_LOCATION*.

Assumptions:

- A backend database has been set up and configured
- The CAS service and tests are installed in *$GLOBUS_LOCATION*
- The sample commands assume:

  1. Container is started up on localhost and port 8443.
  2. database username as *tester*
  3. database name as *casDatabase*
  4. database is on host *foo.bar.gov* and default port.

Testing the backend database module

1. Run:

   ```
   cd $GLOBUS_LOCATION
   ```

2. Populate the file `etc/globus_cas_unit_test/casTestProperties` with the following database configuration information:

Table 14.3. Test database properties

dbDriver	The JDBC driver to be used
dbConnectionURL	The JDBC connection url to be used to connect to database
dbUsername	The user name to use when connecting to the database
dbPassword	The password corresponding to the username

3. The database needs to be empty for these tests to work and will be deleted by this target. Run:

   ```
   ant -f share/globus_cas_unit_test/cas-test-build.xml testDatabase
   ```

4. Test reports are placed in `$GLOBUS_LOCATION/share/globus_cas_unit_test/cas-test-reports`.

Important

The database bootstrap needs to be done again for the server to be ready to receive client requests.

Testing the CAS service module

These tests can be set up so as to be able to test multiple user scenario or they can be configured to run as just a single identity. The first set of tests are used to test admin functionality and set up the database for second user. As the second user the permissions and queries are tested to ensure that the set up worked.

All the configuraton information for the test needs to be configured in the `etc/globus_cas_unit_test/casTestProperties` file. The database section of the properties file needs to be configured as described here. In addition the following properties need to be configured to run the tests:

Table 14.4. Test properties

user1SubjectDN	The DN of the user running the first set of tests.
user2SubjectDN	The DN of the user running the second set of tests. This DN has to be different from value specified for user1SubjectDN. Note: Both tests can be run as same user as long as the DN of the certificate being used to run the tests matches the value specified in user1SubjectDN. In this case, the value of user2SubjectDN can be set to a arbitrary string.
maxAssertionLifetime	Should match the value set in the service configuration as shown in Configuration Information [./../WS_AA_CAS_Public_Interfaces.html#s-cas-public-config]
host	Host on which the CAS service is running
port	Port on which the CAS service is running
securityType	This is an optional parameter indicating the security type to use. Can be set to `message` for Secure Message or `conversation` for Secure Conversation and `transport` for Secure Transport (the default configuration).

protType	This is an optional parameter indicating protection type to use. Can be set to `signature` for integrity protection (the default configuration) or `encryption` for privacy protection.

Steps for testing:

1. Run:

   ```
   cd $GLOBUS_LOCATION
   ```

2. Source *`$GLOBUS_LOCATION/etc/globus-devel-env.sh`* or *`$GLOBUS_LOCATION/etc/globus-devel-env.csh`* or *`$GLOBUS_LOCATION/etc/globus-devel-env.bat`* as appropriate for your environment.

3. In the test properties file, set *`user2SubjectDN`* to the subject in your regular proxy. The following returns the appropriate string:

   ```
   casadmin$ java org.globus.tools.CertInfo -subject -globus
   ```

4. Generate an independent proxy using the following command:

   ```
   casadmin$ java org.globus.tools.ProxyInit -independent
   ```

5. Set the identity in the proxy generated from the above step as *user1SubjectDN* in the test properties file. The following command will return the relevant string:

   ```
   casadmin$ java org.globus.tools.ProxyInfo -subject -globus
   ```

6. Start the container on the port and host configured in Table 14.4

7. The following command runs the tests for self permissions and sets up the database for a user with subjectDN user2SubjectDN:

   ```
   casadmin$ ant -f share/globus_cas_unit_test/cas-test-build.xml user1TestService
   ```

8. To test as the second user, generate a proxy for the subject DN specified for the second user:

   ```
   casadmin$ java org.globus.tools.ProxyInit
   ```

9. The database needs to be empty for these tests to work and this target deletes all entries in database. Then run the following:

   ```
   casadmin$ ant -f share/globus_cas_unit_test/cas-test-build.xml user2TestService
   ```

10. Test reports are placed in `$GLOBUS_LOCATION/share/globus_cas_unit_test/cas-test-reports`.

11. After these tests, the CAS database needs to be reset. The following command will delete all entries from the database:

    ```
    casadmin$ psql -U casadmin -d casDatabase -f GLOBUS_LOCATION/share/cas/database_delete.sql
    ```

Important

The database bootstrap needs to be done again for the server to be ready to receive client requests.

Example of CAS Server Administration

The following contains an example of administering the CAS server policies using the CAS administrative clients

described *FILLME: add link to admin command line when its done.*.

Alice, Bob and Carol are three members of a community who have set up a Community Authorization service:

- Alice's role is primarily to administer the CAS server.
- Bob is an analyst who needs read access to much of the community data.
- Carol is a scientist who needs to be able to both read and write community data.

These examples show how:

1. Alice adds the users Bob and Carol to the CAS server
2. Alice adds a FTP server with some data available to the community
3. Alice adds permissions for the users using the CAS administration clients.

These examples assume the following:

- Alice has installed the CAS server and bootstrapped the database with herself as super user. Please refer to previous chapters in this guide for details on setting up server and bootstrapping with data.
- Alice's nickname on the CAS server is *alice* and at bootstrap she has created a user group, *suGroup*,which has super user permissions on the database.
- The CAS service URL is http://localhost:8080/wsrf/services/CASService.
- For all commands listed below the enviroment variable $GLOBUS_LOCATION has been set to point to the Globus Toolkit installation and the commands are run from $GLOBUS_LOCATION.
- The environment variable CAS_SERVER_URL has been set to point to the CAS server URL, http://localhost:8080/wsrf/services/CASService.

Adding a user group

Since at the time of booting up the CAS server only the user group that has super user permissions on the CAS server is created, Alice will want to create another user group to which new users can be added and to which permissions on newly enrolled CAS entities may be given. This also eases the process of giving same rights to many users. Given that there are two types of roles in the community she might want to create two groups, *analysts* and *scientists*.

Also, all permissions on the newly created group will be given to users of a particular user group. For example, Alice may want all users of the user group *analysts* to be able to manipulate the group.

To create a new user group Alice uses the *cas-group-admin* client. It requires a name for the new group being created, say *analysts*.

```
alice%  cas-group-admin user create analysts analysts
```

This will create a user group *analysts* and give all users in that group the permission to manage the group (i.e add users, remove users and so on). She can similarly create a group called *scientist*.

Adding a trust anchor

Prior to adding Bob and Carol to the CAS server, Alice needs to ensure that the trust anchors for both have been added. If they share the trust anchor with Alice then this step can be skipped, since at bootstrap Alice's trust anchor would have been added to the database.

In our example Alice and Carol share a trust anchor different from Bob's. Therefore, Alice needs to add Bob's trust anchor by using the*cas-enroll* client with the *trustAnchor* option. She needs to provide details about the trust anchor such as the authentication method and authentication data used.

```
alice%  cas-enroll trustAnchor analysts AbcTrust X509 "/C=US/O=some/CN=ABC CA"
```

The above will enroll a trust anchor with nickname *AbcTrust*, that uses *X509* as its authentication method and has the DN specified in the command. The members of the *analysts* user group are given all rights on this object. This implies that any user who has this trust anchor is assumed to present credentials signed by this trust anchor.

Adding users

Now Alice can add Bob and Carol as users using the *cas-enroll* command with the *user* option. She needs to provide the user's subject DN and a reference to the trust anchor used by the user. As with any entity added to the CAS server, the admin needs to choose a user group whose members will have all permissions on that entity. In this example, Alice would like the members of the user group *suUser* to be able to maninipulate the user entity *Bob*.

```
alice% cas-enroll user suUser bob "/O=Our Community/CN=Bob Foo" AbcTrust
```

Alice uses a similar command to add Carol to the CAS database.

Adding users to a user group

The CAS server allows rights to be assigned only to user groups and not to individual users. Hence before Alice can assign rights to Bob or Carol, she needs to add them to some user group. She does this by using the **cas-group-add-entry** client with the *user* option to indicate she is adding to a user group. This client requires the group name and the nick name of the user who needs to be added. To add Bob to the *analysts* group, the command would be:

```
alice% cas-group-add-entry user analysts bob
```

If a user group *scientists* was created, Carol could similarly be added as a member.

Adding a new FTP server

Alice now has the community users in the database. The next step is to add some resources. Because the community currently has a the FTP server foo.bar.edu available to it she will add it to the CAS database.

Each resource or object in the CAS server has a namespace associated with it that defines certain features. For example, it can define the comparison algorithm that is to be used when this object's name is compared. It may also define the base URL that should be prefixed to objects that belong to this namespace. In this case, Alice chooses to use the *FTPDirectoryTree* namespace that is added to the CAS server at startup. She uses the *cas-enroll* client with the *object* option to add the FTP server to the CAS database:

```
alice% cas-enroll object suGroup ftp://foo.bar.edu/* FTPDirectoryTree
```

This command adds the FTP server as an object and gives all members of the *suGroup* rights to manipulate the object.

To be able to grant/revoke access on an individual directory, add an object for the directory. For example, if Alice would like to be able to manipulate the *data* directory on the server as a separate entity, the following command will add an object for that.

```
alice% cas-enroll object suGroup ftp://foo.bar.edu/data/* FTPDirectoryTree
```

Creating an object group

Alice suspects that the community will end up with more directories containing data on other servers that will have polices identical with the ones on the /data directory on foo.bar.edu. To manage this she is going to create an object group called *data* and assign foo.bar.edu/data to this group. This will allow her to grant rights on this group and easily add other directories to this group later.

To create a group called *data*, she uses the *cas-group-admin* client with the *group* and *create* options:

```
alice% cas-group-admin object create suGroup data
```

This creates an object group called *data* and the members of *suGroup* get all rights on this group and hence should be able to add/remove members, grant rights to add/delete from this group to others and also delete this group.

Adding members to an object group

Alice now can add foo.bar.edu/data to the *data* group. She can do this by using the *cas-group-add-entry* with the *object* option. To add the above described object, *ftp://foo.bar.edu/data/** in the namespace *FooFTPNamespace* to the object group *data* Alice uses the following command:

```
alice% cas-group-add-entry object data object FooFTPNamespace ftp://foo.bar.edu/data/*
```

In the above command:

- the first *object* refers to the group type.
- *data* is the name of the object group.
- the second *object* refers to the type of CAS entity that is being added as a member.
- the last two parameters define the namespace and the object that needs to be added.

Adding service types

Alice now needs to add information about the kinds of rights that can be granted for these objects. These are stored as *service types* and relevant actions are mapped to these service types.

In this scenario, the kind of service types that Alice should add would be *file*, *directory* and so on. To do so the *cas-enroll* client with the *serviceType* option may be used. To add a service type called *file*and give members of *suGroup* all rights on this service type Alice uses the following command.

```
alice% cas-enroll serviceType suGroup file
```

Adding action mappings

The relevant action mappings to the above mentioned service types would be *read*, *write* and so on. Alice needs to add these mappings to the database so that she can grant rights that allow a user to have *file/read* or *file/write* permissions on some object.

To add action mappings to a service type, she uses the **cas-action** client with `add` option. The following command adds a mapping of action *read* to service type *file*.

```
alice% cas-action add file add
```

Similarly, she can add other mappings, like *write*, to this service type.

Granting permissions

Alice now has resources in the object group *data* and users in the user groups *analysts* and *scientists*. She now wants to grant permissions on the *data* group to the analysts and scientists, namely read permissions to the analysts and read and write permissions to the scientists.

To grant permissions Alice needs to use the **cas-rights-admin** with the `grant` option. To give read permissions to the analysts group Alice runs:

```
alice% cas-rights-admin grant analysts objectGroup data serviceAction file read
```

She similarly grants rights to *scientists* group.

Security Considerations

- The database username/password is stored in the service configuration file and the test properties file. Ensure correct permissions to protect the information.

Troubleshooting

Database connectivity errors

If the CAS service fails with following error:

```
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.userException
 faultSubcode:
 faultString: org.apache.commons.dbcp.DbcpException: Connection
refused. Check t hat the hostname and port are correct and that the
postmaster is accepting TCP/I P connections.
```

- Ensure the database properties (connectionURL, userName, password) in *$GLOBUS_LOCATION/globus_cas_service/jndi-config.xml* are correct.
- Ensure that the database is set up with permission to receive TCP/IP connections

Credential Errors

The following are some common problems that may cause clients or servers to report that credentials are invalid:

Your proxy credential may have expired

Use **grid-proxy-info** to check whether the proxy has actually expired. If it has, generate a new proxy with **grid-proxy-init**.

The system clock on either the local or remote system is wrong

This may cause the server or client to conclude that a credential has expired.

Your end-user certificate may have expired

Use **grid-cert-info** to check your certificate's expiration date. If it has expired, follow your CA's procedures to get a

new one.

The permissions may be wrong on your proxy file

If the permissions on your proxy file are too lax (for example, if others can read your proxy file), Globus Toolkit clients will not use that file to authenticate. You can "fix" this problem by changing the permissions on the file or by destroying it (with **grid-proxy-destroy**and creating a new one (with **grid-proxy-init**). However, it is still possible that someone else has made a copy of that file during the time that the permissions were wrong. In that case, they will be able to impersonate you until the proxy file expires or your permissions or end-user certificate are revoked, whichever happens first.

The permissions may be wrong on your private key file

If the permissions on your end user certificate private key file are too lax (for example, if others can read the file), **grid-proxy-init** will refuse to create a proxy certificate. You can "fix" this by changing the permissions on the private key file; however, you will still have a much more serious problem: it's possible that someone has made a copy of your private key file. Although this file is encrypted, it is possible that someone will be able to decrypt the private key, at which point they will be able to impersonate you as long as your end user certificate is valid. You should contact your CA to have your end-user certificate revoked and get a new one.

The remote system may not trust your CA

Verify that the remote system is configured to trust the CA that issued your end-entity certificate. See the [TODO: add admin guide link] for details.

You may not trust the remote system's CA

Verify that your system is configured to trust the remote CA (or that your environment is set up to trust the remote CA). See the [TODO: add admin guide link] for details.

There may be something wrong with the remote service's credentials

It is sometimes difficult to distinguish between errors reported by the remote service regarding your credentials and errors reported by the client interface regarding the remote service's credentials. If you can't find anything wrong with your credentials, check for the same conditions (or ask a remote administrator to do so) on the remote system.

Chapter 15. GT 4.0 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 GT 4.0 System Administrator's Guide [../../admin/docbook/]. Read through this guide before continuing!

Building and Installing

For detailed instructions on building and installing RLS, see Appendix A.

Configuring

Configuration overview

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` [http://www-unix.globus.org/toolkit/docs/4.0/data/rls/RLS_Public_Interfaces.html#s-rls-Public_Inerfaces-config] 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.

Syntax of the interface

Table 15.1. Settings

`acl user: permission [permission]`	`acl` entries may be a combination of DNs and local usernames. If a DN is not found in the gridmap file then it is used to search the `acl` list. 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.

	`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. There may be multiple `acl` entries, the first match found is used to determine a user's privileges. `[permission]` is one or more of the following values: • `lrc_read` Allows client to read an LRC. • `lrc_update` Allows client to update an LRC. • `rli_read` Allows client to read an RLI. • `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.
`authentication true|false`	Enable or disable GSI authentication. The default value is `true`. If authentication is enabled (`true`), clients should use the URL schema `rls:` to connect to the server. If authentication is *not* enabled (`false`), clients should use the URL schema `rlsn:`.
`db_pwd password`	Password to use to connect to the database server. The default value is `changethis`.
`db_user databaseuser`	Username to use to connect to database server. The default value is `dbperson`.
`idletimeout seconds`	Seconds after which idle connections close. The default value is `900`.
`loglevel N`	Sets loglevel to N (default is `0`). Higher levels mean more verbosity.
`lrc_bloomfilter_numhash N`	Number of hash functions to use in Bloom filters. The default value is `3`. Possible values are 1 through 8. This value, in conjunction with`lrc_bloomfilter_ratio`, will determine the

	number of false positives that may be expected when querying an RLI that is updated via Bloom filters. *Note:* The default values of `3` and `10` give a false positive rate of approximately 1%.
`lrc_bloomfilter_ratio N`	Sets ratio of bloom filter size (in bits) to number of LFNs in the LRC catalog (in other words, size of the Bloom filter as a multiple of the number of LFNs in the LRC database.) Only meaningful if Bloom filters are used to update an RLI. Too small a value will generate too many false positives, too large wastes memory and network bandwidth. The default value is `10`. *Note:* The default values of `3` and `10` give a false positive rate of approximately 1%.
`lrc_buffer_time N`	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`.
`lrc_dbname`	Name of LRC database. The default value is `lrcdb`.
`lrc_server true|false`	If LRC server, the value should be `true`. The default value is `false`.
`lrc_update_bf seconds`	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 softstate update. This can be much smaller than the interval between updates without using Bloom filters (`lrc_update_ll`). The default value is `300`.
`lrc_update_factor N`	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 softstate update), then the interval between RLI updates for this server (`lrc_update_ll`) is multiplied by the value of this option.
`lrc_update_immediate true|false`	Turns LRC to RLI immediate mode updates on (`true`) or off (`false`). The default value is `false`.
`lrc_update_ll seconds`	Number of seconds before an LRC server does a LFN list softstate update. The default value is `86400`.

`lrc_update_retry seconds`	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`.
`maxbackoff seconds`	Maximum seconds to wait before re-trying listen in the event of an I/O error. The default value is `300.`
`maxfreethreads N`	Maximum number of idle threads, excess threads are killed. The default value is `5.`
`maxconnections N`	Maximum number of simultaneous connections. The default value is `100`.
`maxthreads N`	Maximum number of threads running at one time. The default value is `30`.
`myurl URL`	URL of server. The default value is `rls://<hostname>:port`
`odbcini filename`	Sets environment variable `ODBCINI`. If not specified, and `ODBCINI` is not already set, then the default value is `$GLOBUS_LOCATION/var/odbc.ini.`
`pidfile filename`	Filename where pid file should be written. The default value is `$GLOBUS_LOCATION/var/<programname>.pid`.
`port N`	Port the server listens on. The default value is `39281.`
`result_limit limit`	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 `result_limit` records are returned by a query.
`rli_bloomfilter true|false`	RLI servers must have this set to accept Bloom filter updates.

	If `true`, then only Bloom filter updates are accepted from LRCs. If `false`, full LFN lists are accepted. *Note:* If Bloom filters are enabled, then the RLI does *not* support wildcarded queries.
`rli_bloomfilter_dir none|default|pathname`	If an RLI is configured to accept bloom filters (`rli_bloomfilter true`), then Bloom filters may be saved to this directory after updates. This directory is scanned when an RLI server starts up and is used to initialize Bloom filters for each LRC that updated the RLI. 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. 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. • `none` Bloom filters are not saved to disk. This is the default. • `default` Bloom filters are saved to the default directory: • `$GLOBUS_LOCATION/var/rls-bloomfilters` if `GLOBUS_LOCATION` is set • else, `/tmp/rls-bloomfilters` • `pathname` 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.
`rli_dbname database`	Name of RLI database. The default value is `rlidb`.

`rli_expire_int seconds`	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. The default value is `28800`.
`rli_expire_stale seconds`	Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC.) The default value is `86400`. This value should be no smaller than `lrc_update_ll`. Stale RLI entries are not returned in queries. *Note:* If the LRC server is responding, this value is not used. Instead the value of `lrc_update_ll` or `lrc_update_bf` is retrieved from the LRC server, multiplied by 1.2, and used as the value for this option.
`rli_server true|false`	If RLI server, the value should be `true`. The default value is `false`.
`rlscertfile filename`	Name of X.509 certificate file identifying server. This value is set by setting environment variable `X509_USER_CERT`.
`rlskeyfile filename`	Name of X.509 key file for server. This value is set by setting environment variable `X509_USER_KEY`.
`startthreads N`	Number of threads to start initially. The default value is `3`.
`timeout seconds`	Timeout (in seconds) for calls to other RLS servers (eg for LRC calls to send an update to an RLI).

Deploying

This section does not apply to the RLS.

Testing

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 -d [-N]
   ```

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
```

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 configureation 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.

Troubleshooting

Information on troubleshooting can be found in the FAQ [http://www-unix.globus.org/toolkit/rls/faq/].

Usage statistics collection by the Globus Alliance

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 [http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html] on the collection of usage statistics.

Appendix A. GT 4.0 RLS: Building and Installing

This procedure includes the steps required to set up an RLS server. Post setup configuration (tuning the server parameters etc) are not included in this document.

Requirements

You need to download and install the following software (follow the links to download):

- Installation of GT 4.0 [http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/]
- a Relation Database Server (RDBMS) that supports ODBC. We provide instructions for PostgreSQL and MySQL.
    - If you use PostgreSQL [http://www.postgresql.org], you'll also need psqlODBC [http://gborg.postgresql.org] (the ODBC driver for PostgreSQL)
    - If you use MySQL [http://www.mysql.com], you'll also need the MyODBC [http://www.mysql.com] (Connector/ODBC) packages. MySQL's top level installation directory must be specified. By default these are assumed to be in $GLOBUS_LOCATION.
- the iODBC [http://www.iodbc.org] package is used to interface to the ODBC layer of the RDBMS. The location of iODBC and the odbc.ini file must be specified before installing the RLS server.

Setting environment variables

The following environment variables can be used to override the default locations. These should be set prior to installing the RLS server.

The location of iODBC and the odbc.ini file must be specified before installing the RLS server. Also if you're using MySQL its top level installation directory must be specified. By default these are assumed to be in $GLOBUS_LOCATION.

In addition if you're building from source and wish to build the client Java API (included in the server bundles) you need to set the path to the Java Development Toolkit (JDK) version 1.4 or later.

Table A.1. RLS Build Environment Variables

Variable	Default
`GLOBUS_IODBC_PATH`	`$GLOBUS_LOCATION`
`ODBCINI`	`$GLOBUS_LOCATION/var/odbc.ini`
`JAVA_HOME`	none
`GLOBUS_MYSQL_PATH`	`$GLOBUS_LOCATION (if using MySQL)`

You can use the following commands to set these variables. You only need to set these variables for RLS installation, they are not used when running RLS. This document assumes you are using the *csh* shell or one of its variants, if you're using *sh* or something similar (eg *bash*) you should change the *setenv* commands to *export variable=value*

.

- `setenv GLOBUS_IODBC_PATH $GLOBUS_LOCATION`
- `setenv ODBCINI $GLOBUS_LOCATION/var/odbc.ini`
- `setenv JAVA_HOME /usr/jdk/1.4`
- `setenv GLOBUS_MYSQL_PATH $GLOBUS_LOCATION # if using MySQL`

Installing iODBC

Run the install commands

The following commands were used during RLS development to install iODBC version 3.0.5.

```
% cd $IODBCSRC % ./configure --prefix=$GLOBUS_IODBC_PATH --disable-gtktest --with-pthreads --disable
    --with-iodbc-inidir=$ODBCINIDIR % gmake % gmake install
```

where:

- `$IODBCSRC` is the directory where you untarred the iODBC sources
- `$ODBCINIDIR` is the directory where you plan to install the `odbc.ini` file (which you will create in the next step.)

Create the odbc.ini file

Create the `odbc.ini` file in `$ODBCINIDIR`:

The contents should include the path to where you intend to install the ODBC driver for your RDBMS (such as `psqlodbc.so` or `libmyodbc3.so`).

The following is an example that should work with psqlODBC. It assumes you will name your LRC and RLI databases `lrc1000` and `rli1000`:

```
[ODBC Data Sources]
lrc1000=lrc database
rli1000=rli database


[lrc1000]
Description=LRC database
DSN=lrc1000
Servertype=postgres
Servername=localhost
Database=lrc1000
ReadOnly=no


[rli1000]
Description=RLI database
DSN=rli1000
Servertype=postgres
Servername=localhost
Database=rli1000
ReadOnly=no


[Default]
```

```
Driver=/path/to/psqlodbc.so
Port=5432
```

Note: You do not need an RLI database if you plan to use Bloom filters for LRC to RLI updates (Bloom filters are kept in memory), in this case you can omit the RLI entries below.

Bug: psqlODBC will not find a Data Source Name (DSN) in the system *odbc.ini* file *$ODBCINIDIR/odbc.ini* . It will find DSNs in the user's *odbc.ini* file if it exists <$HOME/.odbc.ini).

One work around is to copy or symlink the system *odbc.ini* file to each user's home directory. psqlODBC does find system DSNs in a file called *odbcinst.ini* , which is looked for in the etc subdirectory where iODBC was installed *$GLOBUS_IODBC_PATH/etc/odbcinst.ini* . So another option besides creating user *.odbc.ini* files is to copy or symlink the system odbc.ini file to *$GLOBUS_IODBC_PATH/etc/odbcinst.ini* . Someone who understands this better may have a better answer.

Changing how clients connect to the server (for MySQL only)

If you're using MySQL and have changed how MySQL clients connect to the MySQL server in *my.cnf* (eg the port number or socket name) then you should set option to `65536` in `odbc.ini` for each database. This tells MyODBC to read the client section of `my.cnf` to find the changed connection parameters.

```
[lrc1000] option = 65536
[rli1000] option = 65536
```

Installing the relational database

We include instructions for both PostgreSQL (the section called “Using PostgreSQL”) and MySQL(the section called “Using MySQL ”).

Using PostgreSQL

If your relational database of choice is PostgreSQL, you need to install and configure both PostgreSQL and psqlODBC (the ODBC driver for PostgreSQL) as follows:

Installing PostgreSQL

Running the install commands

The commands used to install Postgres 7.2.3 on the RLS development system were as follows.

```
% cd $POSTGRESSRC % ./configure --prefix=$GLOBUS_LOCATION % gmake % gmake install
```

`$POSTGRESSRC` is the directory where the PostgreSQL source was untarred.

Initializing PostgreSQL

Initialize PostgreSQL and start the server by running:

```
initdb -D /path/to/postgres-datadir
postmaster -D /path/to/postgres-datadir -i -o -F
```

The `-o -F` flags to postmaster disable `fsync()` calls after transactions (which, although it improves performance, raises the risk of DB corruption.)

Creating the user and password

Create the database user (in our example, called `dbuser`) and password that RLS will use:

```
createuser -P dbuser
```

Important: Be sure to do periodic `vacuum` and `analyze` commands on all your Postgres databases. The Postgres 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 softstate updates) timeout and fail. Please see the Postgres documentation for further details.

Installing psqlODBC

Install psqlODBC by running the following commands (the following commands were used to install psqlODBC 7.2.5):

```
% cd $PSQLODBCSRC
% setenv CPPFLAGS -I$(IODBC_INSTALLDIR)/include
% ./configure --prefix=$GLOBUS_LOCATION --enable-pthreads
% gmake
% gmake install
```

where `$PSQLODBCSRC` is the directory where you untarred the psqlODBC source.

Note: The configure script that comes with psqlODBC supports a `--with-iodbc` optionl. However when the RLS developers used this it resulted in RLS servers with corrupt memory that would dump core while opening the database connection. It seems to work fine (with iODBC) without this option.

You can now continue to instructions for Installing the RLS Server. See the section called “Installing the RLS Server”.

Using MySQL

If your relational database of choice is MySQL, you'll need to install and configure both MySQL and the MyODBC (Connector/ODBC) packages as follows:

Installing MySQL

Once you've installed and configured MySQL you must start the database server and create the database user/password that RLS will use to connect to the database.

Starting database server

Start the database server by running:

```
mysqld_safe [--defaults-file path to your my.cnf file ]
```

Creating the user and password

To create the database user and password that RLS will use you must run the MySQL command line tool *mysql* , and specify the following commands.

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

These commands assume the username you will create for RLS is `dbuser` with password `dbpassword` , and the database(s) you will create for your LRC and/or RLI server are `lrc1000` and `rli1000`.

Creation of the LRC and/or RLI databases is covered below in the section called “Configuring the RLS Database”.

Installing MyODBC

Recommended Version: 3.51.06

These instructions assume that iODBC was installed in `$GLOBUS_LOCATION`, this may be changed by changing the

--with-iodbc-includes and *--with-iodbc-libs* options.

Running install commands

Install MyODBC in `$GLOBUS_LOCATION` (you may choose a different directory if you wish, by changing the *--prefix* option to *configure* below.):

```
% cd $MYODBCSRC
% ./configure --prefix=$GLOBUS_LOCATION
     --with-mysql-libs=$GLOBUS_MYSQL_PATH/lib/mysql
         --with-mysql-includes=$GLOBUS_MYSQL_PATH/include/mysql
         --with-iodbc-includes=$GLOBUS_LOCATION/include
         --with-iodbc-libs=$GLOBUS_LOCATION/lib
         --with-odbc-ini=$ODBCINIDIR
% gmake
% gmake install
```

where:

- `$MYODBCSRC` is the directory where you untarred the MyODBC sources.
- `$ODBCINIDIR` is the directory where you created the *odbc.ini* file.

Bug: There is a bug in MyODBC version 3.51.05 and earlier. The debug code is not thread safe, and the RLS server will get a segmentation violation and die if this code is enabled. In versions 3.51.05 and later the debug code can be disabled with the configure option `--without-debug`. In earlier versions it is disabled by defining `DBUG_OFF`, as in the following example:

```
setenv CFLAGS -DBUG_OFF
```

You can now continue to instructions for installing the RLS Server. See the section called “Installing the RLS Server”

Installing the RLS Server

Download the appropriate bundle. RLS is included as part of the Globus Toolkit bundle. See the Globus Toolkit Development Downloads [http://www-unix.globus.org/toolkit/downloads/development/] for a listing of available software.

Note: When using these bundles, RLS will *not* be built by the installer script *unless* the environment variable GLOBUS_IODBC_PATH is set.

RLS is installed as a part of the standard install. For basic installation instructions, see the Installation Guide [http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/]

Configuring the RLS Database

RLS server configuration is specified in `$GLOBUS_LOCATION/etc/globus-rls-server.conf`, please see the main page *globus-rls-server(8)* for complete details. Some of the configuration options (such as database user/password) are mentioned below.

Creating a user and password

Create a *database user* that the RLS server will use to connect to the DBMS.

The database user and password you pick must be specified in the RLS server configuration file, as well as the name of the database(s) you will create (see below).

`db_user` *`dbuser`*

```
db_pwd dbpassword
lrc_dbname lrc1000  # optional (if LRC server)
rli_dbname rli1000  # optional (if RLI server)
```

Choosing database for RLS server

Decide which database(s) the RLS server will use (and that you will create in step ?):

- 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 highly compressed Bloom filters. Since Bloom filters are so small, they are kept in memory and no database is required. An RLS server can be configured as both an LRC and RLI server

Configuring database schema

Configure the schema file(s) for the database(s) you will create.

GT 4.0 installed the schema files for the LRC and RLI databases 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

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`.

By default the database user is *dbuser* , the LRC database name is *lrc1000* and the RLI database name is *rli1000*.

Creating the database(s)

Create the database(s) with the following commands (note once again that you do *not* need to create an RLI database if you are configuring an RLI server updated by Bloom filters):

For PostgreSQL, run:

```
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.sql
psql -W -U dbuser  -d rli1000  -f $GLOBUS_LOCATION/setup/globus/globus-rls-rli-postgres.sql
```

For MySQL, run:

```
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
```

Configuring the RLS Server

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 the options.

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
```

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

Starting the RLS Server

Start the RLS Server by running:

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

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.

Stopping the RLS Server

Stop the RLS Server by running:

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

Configuring the RLS Server for the WS MDS Index Service

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

1. To enable Index Service reporting, add the contents of the file `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf` to the WS MDS Index Service configuration file: `$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-resoruce-ldif.conf`, you can use the following commands to do so:

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

Redhat 9 Incompatibility

This note applies to Redhat 9 but could also apply to other Linux distributions.

There have been occurences of RLS servers hanging on Redhat 9 systems.

The external symptoms are:

1. 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
   ```

2. 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()`.

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
  - 2.4.20-8
- *glibc:*
  - glibc-2.3.2-27.9.7

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:

```
<--- START --->
#!/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
<--- END --->
```

On i586 systems, set:

```
LD_ASSUME_KERNEL=2.2.5
```