GT 6.0 Installation Appendix

GT 6.0 Installation Appendix

Table of Contents

List of Tables

List of Examples

Chapter 1. Advanced Installation for GT 6.0

This section introduces building from Git, and references the advanced installation sections of component documentation.

1. Advanced Installation

1.1. Building from Git

The Globus Toolkit is available for download from github. See https://github.com/globus/globus-toolkit/tree/globus_6_branch to branch or checkout the toolkit source code.

After checking out the toolkit, run **autoreconf -i** in the checkout root to generate configure scripts for building the toolkit components.

1.2. Building a specific package from source

If you need to build a specific package from the source installer, you can use the per-package make targets that exist in the source installer's Makefile. Instead of simply running "make" in the steps above, you can, for example, run "make globus_common" which will build the globus_common package and its dependencies, or "make globus_common-only" which will build exactly and only the globus_common package. Similar targets exist for each package.

1.3. Detailed installation instructions for these components

The following is a list of links to more detailed installation information available for the following components:

- GRAM5 Installation
- Building and installing GridFTP
- Building and installing MyProxy
- Optional Build-Time Configuration for GSI-OpenSSH

1.4. Building an update package without an installer

If you need to build an updated package that has been released without a source installer (for example, a security update to a package, or a new version of MyProxy,) you can use the familiar **configure; make; make install** sequence to rebuild that package.

Appendix A. Packaging details

1. The makefile

You do not have to build every subcomponent of this release. The makefile specifies subtargets for different functional subpieces.

Makefile targets

- gram: GRAM5
- gridftp: GridFTP

Note that all of these targets require the "install" target also. So, for instance, to build GridFTP alone, you would run:

```
$ ./configure --prefix=/path/to/install
$ make gridftp install
```

2. Linking with Globus Toolkit Libraries

Since GT 2.0, the toolkit has included a script called globus-makefile-header that can be used to assemble the cflags and link line information when linking a program with libraries included in the toolkit. This script would walk the package metadata dependency tree to ensure that all needed flags were included, without duplicates. This method worked, and continues to work in GT 6.0, but we consider it to be obsolete, as we have added support for using pkg-config[1]

Pkg-config is very similar in concept to globus-makefile-header, but it has gained widespread adoption across a range of unix platforms.

To get the cflags and link line information to link to the globus-ftp-client library, for example, you could

```
$pkg-config --cflags --libs globus-ftp-client
```

Each Globus Toolkit library has a pkg-config metadata file that is installed as part of its devel package.

For more information about pkg-config, please see the pkg-config homepage.[2]

[1] http://www.freedesktop.org/wiki/Software/pkg-config
[2] http://www.freedesktop.org/wiki/Software/pkg-config

Appendix B. Environment Variables in GT 6.0

1. Common Runtime Environmental Variables

1.1. Environmental variables for XIO

The vast majority of the environment variables that affect the Globus XIO framework are defined by the driver in use. The following are links to descriptions of the more common driver environment variables:

- TCP Driver Environment Variables[1]
- File Driver Environment Variables[2]
- GSI Driver Environment Variables[3]
- UDP Driver Environment Variables[4]

1.2. Environment variables for C Common Libraries

GLOBUS_HOSTNAME	Set this variable to the fully qualified name of the local machine's hostname.
GLOBUS_DOMAIN_NAME	Set this variable to the domain name to be used to qualify the local machine's hostname.
GLOBUS_ERROR_OUTPUT	Set this variable to 1 to cause Globus libraries to display error information to stderr.
GLOBUS_ERROR_VERBOSE	Set this variable to 1 to enable verbose error messages.
GLOBUS_I18N	Set this variable to 1 to attempt to use localized messages. (Currently not working)
GLOBUS_LOCATION	Set this variable to the path where the Globus Toolkit is installed, so that Globus tools can find libraries and data files. This is only needed if the Globus Toolkit was built with the source installer.
GLOBUS_THREAD_MODEL	Set to the name of a thread model to control the operation of the Globus event driver. Valid values are (depending on the platform) `none` for non-threaded operation (the default), `pthread` for POSIX threads, or `windows` for Windows threads.

2. Security Environmental Variables

2.1. Environmental Variables for GSI C

2.1.1. Credentials

Credentials are looked for in the following order:

1. service credential
2. host credential

3. proxy credential
4. user credential

`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 *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.pem` and `/etc/grid-security/`*`service/service`*`key.pem` exist and contain a valid certificate and key, those files are used.
3. Otherwise, if the files `$GLOBUS_LOCATION/etc/grid-security/`*`service/service`*`cert.pem` and `$GLOBUS_LOCATION/etc/grid-security/`*`service/service`*`key.pem` exist and contain a valid certificate and key, those files are used.
4. Otherwise, if the files *`service/service`*`cert.pem` and *`service/service`*`key.pem` 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_KEY` 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 `$GLOBUS_LOCATION/etc/hostcert.pem` and `$GLOBUS_LOCATION/etc/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.

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.

2.1.2. Gridmap file

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

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

- Otherwise, the grid map file is `$HOME/.gridmap`.

2.1.3. Trusted CAs directory

`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.

2.1.4. GSI authorization callout configuration file

`GSI_AUTHZ_CONF` is used to specify the 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, then this file is used.

2.1.5. GAA (Generic Authorization and Access control) configuration file

`GSI_GAA_CONF` is used to specify the 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, then this file is used.

2.1.6. Grid security directory

`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.

2.1.7. Using TLS

`GLOBUS_GSSAPI_FORCE_TLS` specifies whether to use TLS by default when establishing a security context. The default behavior if this is not set is to use SSLv3.

2.1.8. Name Comparisons

`GLOBUS_GSSAPI_NAME_COMPATIBILITY` specifies what name matching algorithms are supported by GSSAPI for mutual authentication and `gss_compare_name`. This variable may be set to any of the following values:

`STRICT_GT2`	Strictly backward-compatible with GT 2.0 name matching. X.509 subjectAltName values are ignored. Names with hyphens are treated as wildcarded as described in the security considerations documentation. Name matching will rely on canonical host name associated with connection IP addresses.
`STRICT_RFC2818`	Support RFC 2818[5] server identity processing. Hyphen characters are treated as normal part of a host name. DNSName and IPAddress subjectAltName extensions are matched against the host and port passed to GSSAPI. If subjectAltName is present, X.509 SubjectName is ignored.
`HYBRID`	Support a hybrid of the two previous name matching algorithms, liberally matching both hyphen wildcards, canonical names associated with IP addresses, and subjectAltName extensions.

If this variable is not set, the `HYBRID` behavior is used.

2.2. Environmental variables for MyProxy

Please refer to the MyProxy Reference Manual[6] for documentation of MyProxy environment variable interfaces.

2.3. Environmental variables for GSI-OpenSSH

The GSI-enabled OpenSSHD needs to be able to find certain files and directories in order to properly function.

The items that OpenSSHD needs to be able to locate, their default location and the environment variable to override the default location are:

- *Host key*

[6] http://myproxy.ncsa.uiuc.edu/man/

Default location: /etc/grid-security/hostkey.pem

Override with X509_USER_KEY environment variable

- *Host certificate*

  Default location: /etc/grid-security/hostcert.pem

  Override with X509_USER_CERT environment variable

- *Grid map file*

  Default location: /etc/grid-security/grid-mapfile

  Override with GRIDMAP environment variable

- *Certificate directory*

  Default location: /etc/grid-security/certificates

  Override with X509_CERT_DIR environment variable

3. Data Management Environmental Variables

3.1. Environment variables for GridFTP

The GridFTP *server* or *client* libraries do not read any environment variable directly, but the security and networking related variables described below may be useful.

- Non-WS (General) Authentication & Authorization Environment Variables.
- XIO Network Driver Environment Variables.

Appendix C. Installing SimpleCA

1. 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 `simpleca` account, which will be used to administer the Simple CA. This is created automatically if you install SimpleCA from RPM or Debian packages.
- A generic `globus` account, if you will be building from the source installer.

2. Install SimpleCA

SimpleCA can be installed in three ways, from a debian package, from an RPM package, and from the source installer. These installation methods are described in Installing GT 6.0

To install SimpleCA from binary packages, install the packages globus-simple-ca and globus-gsi-cert-utils-progs and their dependencies. On Debian based systems, use the command

```
elephant# apt-get install globus-simple-ca globus-gsi-cert-utils-progs
```

On RPM-based systems, use the command

```
elephant# yum install globus-simple-ca globus-gsi-cert-utils-progs
```

To install SimpleCA from the source installer, build the `globus_simple_ca` and `globus_gsi_cert_utils` installer targets with the command

```
globus@elephant% make globus_simple_ca globus_gsi_cert_utils
```

Afterward, run the command

```
globus@elephant% make install
```

3. Create SimpleCA Administrator Account

Create a user to adminster the SimpleCA. You can use the the `globus` user you used to build Globus, or another user that you create. For the purposes of this document, we'll assume a user named `simpleca`. Log in to that user, and run the **grid-ca-create** command. This will prompt for information needed to name the certificate, how to contact the CA administrator, lifetime of the CA certificate, and passphrase, and will then generate the new CA certificate and private key. Command-line options described in grid-ca-create can be used to avoid some of these prompts.

4. Invoking grid-ca-create

If you are creating a SimpleCA for testing purposes, you can use the `-noint` command-line option to **grid-ca-create** to use the default values for all prompts like this:

```
simpleca@elephant% grid-ca-create -noint
```

This will create a SimpleCA in the `simpleca`'s home directory with the passphrase `globus`. You can then move on to the Using a SimpleCA chapter of this document. For step-by-step details to create a customized SimpleCA, continue reading this chapter.

As the `simpleca` user, run the command **grid-ca-create**, and you'll see output like this:

```
simpleca@elephant% grid-ca-create
    C e r t i f i c a t e    A u t h o r i t y    S e t u p

This script will setup a Certificate Authority for signing Globus
users certificates.  It will also generate a simple CA package
that can be distributed to the users of the CA.

The CA information about the certificates it distributes will
be kept in:

/home/simpleca/.globus/simpleCA
```

This intro screen shows the path that the CA will be created into (in this example, `/home/simpleca/.globus/simpleCA`). The other commands needed by SimpleCA will automatically look in that path by default when invoked by the `simpleca` user.

5. Configure the subject name

The **grid-ca-create** program next prompts you for information about the name of CA you wish to create:

```

The unique subject name for this CA is:
cn=Globus Simple CA, ou=simpleCA-elephant.globus.org, ou=GlobusTest, o=Grid

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

To accept the default name, enter **`y`**. To choose a different name, type **`n`**, after which you will be prompted by

```
Enter a unique subject name for this CA:
```

The subject name is an X.509 distinguished name. Typical name component type abbreviations used in Grids are:

Table C.1. CA Name components

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

6. Configure the CA's email

The next prompt looks like this:

```
Enter the email of the CA (this is the email where certificate
requests will be sent to be signed by the CA) [simpleca@elephant.globus.org]:
```

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. When users request certificates with **grid-cert-request**, they will be instructed to send the request to this address.

7. 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.

To accept the default, hit **enter**, or otherwise, enter a value in days.

8. Create a Passphrase to Encrypt the CA's Private Key

The next prompt will be for the passphrase for the CA's private key. It will be used to decrypt the CA's private key when signing certificates. It should be hard to guess, as its compromise might compromise all the certificates signed by the CA. You will be prompted twice for the passphrase, to verify that you typed it correctly. Enter the passphrase at these prompts.

```
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

9. SimpleCA Distribution Files

Finally **grid-ca-create** will create a tarball containing the public information about the CA, including its public certificate, signing policy, and supported X.509v3 extensions. This information is needed on machines that will be trusting the CA, and also on machines which will be used to request certificates from this CA.

Since we didn't run in this example as root, **grid-ca-create** will not be able to write the CA files to system paths, so it displays a warning message indicating that. We can use the tarball output here, or packages described below to install the CA support files on this and other machines.

The package output summary looks like this:

```
Insufficient permissions to install CA into the trusted certifiicate
directory (tried ${sysconfdir}/grid-security/certificates and
${datadir}/certificates)
```

```
Creating RPM source tarball... done
 globus_simple_ca_68ea3306
```

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 is an 8 digit hexadecimal string which is a hash of the subject name of the CA certificate.

The tarball contains Debian and RPM package metadata, so that it can be compiled to a binary package which can be easily installed on this and other systems on your Grid. It can also be packaged as a GPT setup package for compatibility with older versions of the Globus Toolkit.

10. Generating Binary CA Packages

The grid-ca-package command can be used to generate RPM, debian, or legacy GPT packages for a SimpleCA, or for any other CA which is installed on a host. These packages can make it easy to distribute the CA certificate and policy to other hosts with which you want to establish Grid trust relationships.

10.1. Generating RPM Packages

To generate an RPM package for the CA which we created, use the following command:

```
simpleca@elephant% grid-ca-package -r -cadir ~/.globus/simpleCA
Creating RPM source tarball... done
        globus_simple_ca_68ea3306.tar.gz
Creating RPM binary.../home/simpleca/globus-simple-ca-68ea3306
```

The resulting rpm package will be placed in the current directory. As root, you can install this via the **yum** or **rpm** tools. This package can then be installed on any RPM-based system.

10.2. Generating Debian Packages

To generate an Debian package for the CA which we created, use the following command:

```
simpleca@elephant% grid-ca-package -d -cadir ~/.globus/simpleCA
    Creating RPM source tarball... done
        globus_simple_ca_68ea3306.tar.gz
Creating debian binary...dpkg-buildpackage: export CFLAGS from dpkg-buildflags (origin: ve

...
Lots of dpkg-buildpackage output
```

The resulting debian package will be placed in the current directory. As root, you can install this via the **dpkg** tool.

10.3. Generating GPT Packages

The **grid-ca-package** command can also generate GPT packages in the form similar to previous versions of the Globus Toolkit. This is done with the `-g` and `-b` command-line options. See grid-ca-package for more details.

11. Examining a Certificate Request

To examine a certificate request, use the command **openssl req -text -in *`REQNAME`***, as shown in the following example.

Example C.1. Examine a Certificate Request

```
simpleca@elephant% openssl req -noout -text -in certreq.pem
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: o=Grid, OU=GlobusTest, OU=simpleCA-elephant.globus.org, OU=local, CN=Joe
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    79:bd:a7:29:16:77:4c:e9:82:d3:73:a0:25:34:c7:
                    25:07:67:b3:2d:11:c1:e2:c9:b1:ec:41:20:a7:9a:
                    b7:2f:ee:d4:88:78:14:ff:d4:f2:f9:1b:d3:56:bc:
                    37:6f:f0:06:ea:b0:6f:70:12:a8:34:ac:8e:be:98:
                    00:b9:b8:ec:39:b5:6b:23:ad:1b:00:62:4b:cc:79:
                    97:cc:56:fb:54:7b:03:6d:a7:76:27:4e:ce:bd:94:
                    d0:eb:59:6b:25:c5:30:b0:47:15:bc:11:d5:7e:ff:
                    04:13:70:de:3b:8f:80:65:ae:63:82:61:38:f9:c6:
                    03:4a:92:b0:de:6f:bb:0a:bd
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            Netscape Cert Type:
                SSL CA, S/MIME CA, Object Signing CA
    Signature Algorithm: sha1WithRSAEncryption
        85:70:a6:5d:de:be:61:45:83:48:43:8d:4b:4b:4a:79:79:98:
        0d:6c:d4:a9:96:26:41:a4:c2:94:10:92:ad:eb:ad:c5:3c:bf:
        d6:4e:84:0a:db:46:96:a9:52:5b:90:cc:6a:d1:57:73:27:98:
        9e:e2:8c:9a:7f:b4:ab:a8:28:2b:02:98:a2:d8:69:73:5e:12:
        ad:5b:de:0c:6e:60:e0:0f:2c:ad:8d:b9:59:3b:d3:49:19:52:
        e0:e1:8a:57:f2:c3:a6:4d:b9:2c:5c:58:ef:0e:59:84:55:8e:
        16:fc:f4:39:82:13:6f:28:a9:59:e3:c8:f1:4e:87:75:33:4f:
        ae:be
```

In this case, you see a certificate request with the subject distinguished name `o=Grid, OU=GlobusTest, OU=simpleCA-elephant.globus.org, OU=local, CN=Joe User`.

12. Signing a Certificate Request

If you are satisfied with the certificate request and are willing to sign it, use the **grid-ca-sign** command to do so. The command will store a copy of the newly signed certificate in the SimpleCA directory, so that you can keep track of what you've signed, and will also write it to the value of the `-out` parameter. Transmit this result file back to the user which requested the certificate.

Example C.2. Sign with grid-ca-sign

```
simpleca@elephant% grid-ca-sign -in certreq.pem -out cert.pem


To sign the request
please enter the password for the CA key:
```

```
The new signed certificate is at: /home/simpleca/.globus/simpleCA/newcerts/01.pem
```

Once you've signed the certificate, if it is a user certificate, you must communicate it back to the user, perhaps via email.

13. Revoking a Certificate

SimpleCA does not yet provide a convenient interface to revoke a signed certificate, but it can be done with the **openssl** command.

Example C.3. Revoke a certificate

```
simpleca@elephant% openssl ca -config ~/.globus/simpleCA/grid-ca-ssl.conf -revoke ~/.globu
Using configuration from /home/simpleca/.globus/simpleCA/grid-ca-ssl.conf
Enter pass phrase for /home/simpleca/.globus/simpleCA/private/cakey.pem:
Revoking Certificate 01.
Data Base Updated
```

Once a certificate is revoked, you can generate a Certificate Revocation List (CRL) for your CA, which will be a signed list of certificates which have been revoked. Sites which use your CA will need to keep the CRL up to date to be able to reject revoked certificates. This CRL can be generated with an **openssl** command. See ca(1) for details about how to control the CRL lifetime and other options.

Example C.4. Create CRL

```
simpleca@elephant% openssl ca -config ~/.globus/simpleCA/grid-ca-ssl.conf -gencrl > CAHASH
Using configuration from /home/simpleca/.globus/simpleCA/grid-ca-ssl.conf
Enter pass phrase for /home/simpleca/.globus/simpleCA/private/cakey.pem:
```

The output file `CAHASH.crl` (based on the hash of your CA subject name) should be distributed to sites which trust your CA, so that they can install it into the trusted certificate directory.

14. Renewing a CA

The **openssl** command can be used to renew a CA certificate. This will generate a new CA certificate with the same subject name and private key as before, but valid for a different time interval. This new certificate packaged and distributed as before using grid-ca-package.

Example C.5. Renew CA Certificate

```
simpleca@elephant% openssl req -key ~/.globus/simpleCA/private/cakey.pem -new -x509 -days
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:
Level 0 Organizational Unit [GlobusTest]:
Level 1 Organizational Unit [simpleCA-elephant.globus.org]:
Name (E.g., John M. Smith) []:Globus Simple CA
```

Important

The Subject Name of the new certificate must match *exactly* the previous certificate name, or clients will not recognize it as the correct certificate.

15. Security considerations for SimpleCA

The operator of a CA must protect the private key of the CA. It should not be stored unencrypted or on a network filesystem.

Simple CA enforces the subject name policies in the simple CA's configuration files. If modified, the signing_policy file distributed to clients of the CA must also be modified.

Appendix D. Troubleshooting your installation

The following is a list of links that take you to information about troubleshooting your installation by component

- Common Runtime components
  - XIO
  - C Common Libraries
- Security components
  - GSI C
  - MyProxy
  - GSI-OpenSSH
- Data Management components
  - GridFTP
- Execution Management components
  - GRAM5

Appendix E. Detailed Configuration by Component

The following is a list of links that take you to information about detailed configuration for each component.

- Common Runtime components
  - XIO
- Security components
  - GSI C
  - MyProxy
  - GSI-OpenSSH
- Data Management components
  - GridFTP
- Execution Management components
  - GRAM5

Appendix F. Security Considerations in GT 6.0

1. Common Runtime

1.1. Security considerations for XIO

Globus XIO is a framework for creating network protocols. Several existing protocols, such as TCP, come built into the framework. XIO itself introduces no known security risks. However, all network applications expose systems to the risks inherent when outsiders can connect to them. Also included in the XIO distribution is the GSI driver, which provides a driver that allows for secure connections.

2. Security

2.1. Security considerations for GSI C

- During host authorization, the toolkit treats host names of the form "hostname-`ANYTHING`.edu" as equivalent to "hostname.edu". This means that if a service was set up to do host authorization and hence accept the certificate "hostname.edu", it would also accept certificates with DNs "hostname-`ANYTHING`.edu".

  The feature is in place to allow a multi-homed host following a "hostname-interface" naming convention, to have a single host certificate. For example, host "grid.test.edu" would also accept the likes of "grid-1.test.edu" or "grid-foo.test.edu".

  Note

  The string `ANYTHING` matches only the name of the host and not domain components. This means that "hostname.edu" will not match "hostname-foo.sub.edu", but will match "host-foo.edu".

  Note

  If a host was set up to accept "hostname-1.edu", it will not accept "hostname-`ANYTHING`.edu" but will accept "hostname.edu". That is, only one of the names being compared may contain the hyphen character in the host name.

  A bug[1] has been opened to see if this feature needs to be modified.

  In GT 6.0, it is possible to disable this behavior, by setting the enviornment variable `GLOBUS_GSSAPI_NAME_COMPATIBILITY` to `STRICT_RFC2818`.

2.2. MyProxy 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.

2.3. GSI-OpenSSH Security Considerations

GSI-OpenSSH is a modified version of OpenSSH[2] and includes full OpenSSH functionality. For more information on OpenSSH security, see the OpenSSH Security[3] page.

3. Data Management

3.1. Security Considerations

3.1.1. Ways to configure your server

There are various ways to configure your GridFTP server that provide varying levels of security. For more information, see System Administrator's Guide.

3.1.2. Firewall requirements

If the GridFTP *server* is behind a firewall:

1. Contact your network administrator to open up port 2811 (for GridFTP control channel connection) and a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

2. Set the environment variable GLOBUS_TCP_PORT_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where `min`,`max` specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP server to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of (local) ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_SOURCE_RANGE=min,max
   ```

   where `min`,`max` specify the port range that you have opened for the outgoing connections on the firewall. This restricts the GridFTP server to bind to a local port in this range for outbound connections. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Note

If the server is behind NAT, the `--data-interface <real ip/hostname>` option needs to be used on the server.

If the GridFTP *client* is behind a firewall:

1. Contact your network administrator to open up a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

[2] http://www.openssh.org/
[3] http://www.openssh.org/security.html

2. Set the environment variable GLOBUS_TCP_PORT_RANGE

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP client to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the GridFTP client to bind to a local port in this range for outbound connections. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Additional information on Globus Toolkit Firewall Requirements is available here[4].

4. Execution Management

4.1. Security Considerations

4.1.1. Gatekeeper Security Considerations

GRAM5 runs different parts of itself under different privilege levels. The **globus-gatekeeper** runs as root, and uses its root privilege to access the host's private key. It uses the *grid map file* to map *Grid Certificates* to local user ids and then uses the `setuid()` function to change to that user and execute the **globus-job-manager** program

4.1.2. Job Manager Security Considerations

The **globus-job-manager** program runs as a local non-root account. It receives a delegated limited *proxy certificate* from the GRAM5 client which it uses to access Grid storage resources via GridFTP and to authenticate job signals (such as client cancel requests), and send job state callbacks to registered clients. This proxy is generally short-lived, and is automatically removed by the job manager when the job completes.

The **globus-job-manager** program uses a publicly-writable directory for job state files. This directory has the *sticky* bit set, so users may not remove other users files. Each file is named by a UUID, so it should be unique.

4.1.3. Fork SEG Module Security Considerations

The Fork Scheduler Event Generator module uses a globally writable file for job state change events. This is not recommended for production use.

[4] http://www.globus.org/toolkit/security/firewalls/

Appendix G. Usage Statistics

The following components collect usage statistics as outlined here (along with information about how to opt-out): Usage Statistics in GT[1]

1. Data Management Usage Statistics

1.1. GridFTP-specific usage statistics

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[1] 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, use the `-disable-usage-stats` option of globus-gridftp-server. 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[3] on the collection of usage statistics.

2. Execution Management Usage Statistics

2.1. GRAM5-specific usage statistics

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.

[1] ../../Usage_Stats.html
[1] /toolkit/docs/6/6.0/Usage_Stats.html
[3] /toolkit/docs/latest-stable/Usage_Stats.html

- Job Manager Session ID
- dryrun used
- RSL Host Count
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_UNSUBMITTED`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_FILE_STAGE_IN`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_FILE_STAGE_OUT`
- Timestamp when job hit `GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE`
- Job Failure Code
- Number of times status is called
- Number of times register is called
- Number of times signal is called
- Number of times refresh is called
- Number of files named in file_clean_up RSL
- Number of files being staged in (including executable, stdin) from http servers
- Number of files being staged in (including executable, stdin) from https servers
- Number of files being staged in (including executable, stdin) from ftp servers
- Number of files being staged in (including executable, stdin) from gsiftp servers
- Number of files being staged into the GASS cache from http servers
- Number of files being staged into the GASS cache from https servers
- Number of files being staged into the GASS cache from ftp servers
- Number of files being staged into the GASS cache from gsiftp servers
- Number of files being staged out (including stdout and stderr) to http servers
- Number of files being staged out (including stdout and stderr) to https servers
- Number of files being staged out (including stdout and stderr) to ftp servers
- Number of files being staged out (including stdout and stderr) to gsiftp servers
- Bitmask of used RSL attributes (values are 2^id from the gram5_rsl_attributes table)
- Number of times unregister is called

- Value of the `count` RSL attribute
- Comma-separated list of string names of other RSL attributes not in the set defined in `globus-gram-job-manager.rvf`
- Job type string
- Number of times the job was restarted
- Total number of state callbacks sent to all clients for this job

The following information can be sent as well in a job status packet but it is not sent unless explicitly enabled by the system administrator:

- Value of the executable RSL attribute
- Value of the arguments RSL attribute
- IP adddress and port of the client that submitted the job
- User DN of the client that submitted the job

In addition to job-related status, the job manager sends information periodically about its execution status. The following information is sent by default in a UDP packet (in addition to the GRAM component code, packet version, timestamp, and source IP address) at job manager start and every 1 hour during the job manager lifetime:

- Job Manager Start Time
- Job Manager Session ID
- Job Manager Status Time
- Job Manager Version
- LRM
- Poll used
- Audit used
- Number of restarted jobs
- Total number of jobs
- Total number of failed jobs
- Total number of canceled jobs
- Total number of completed jobs
- Total number of dry-run jobs
- Peak number of concurrently managed jobs
- Number of jobs currently being managed
- Number of jobs currently in the UNSUBMITTED state
- Number of jobs currently in the STAGE_IN state

- Number of jobs currently in the PENDING state
- Number of jobs currently in the ACTIVE state
- Number of jobs currently in the STAGE_OUT state
- Number of jobs currently in the FAILED state
- Number of jobs currently in the DONE state

Also, please see our policy statement[4] on the collection of usage statistics.

[4] /toolkit/docs/latest-stable/Usage_Stats.html

Glossary

C

CA Certificate	The CA's certificate. This certificate is used to verify signature on certificates issued by the CA. GSI typically stores a given CA certificate in `/etc/grid-security/certificates/<hash>.0`, where <hash> is the hash code of the CA identity.
certificate	A public key plus information about the certificate owner bound together by the digital signature of a CA. In the case of a CA certificate, the certificate is self signed, i.e. it was signed using its own private key.
client	A process that sends commands and receives responses. Note that in GridFTP, the client may or may not take part in the actual movement of data.

G

GAA configuration file	A file that configures the Generic Authorization and Access control GAA libraries. When using GSI, this file is typically found in `/etc/grid-security/gsi-gaa.conf`.
grid map file	A file containing entries mapping certificate subjects to local user names. This file can also serve as a access control list for GSI enabled services and is typically found in `/etc/grid-security/grid-mapfile`. For more information see the Gridmap section here.
grid security directory	The directory containing GSI configuration files such as the GSI authorization callout configuration and GAA configuration files. Typically this directory is `/etc/grid-security`. For more information see this.
GSI authorization callout configuration file	A file that configures authorization callouts to be used for mapping and authorization in GSI enabled services. When using GSI this file is typically found in `/etc/grid-security/gsi-authz.conf`.

H

host certificate	An EEC[2] belonging to a host. When using GSI this certificate is typically stored in `/etc/grid-security/hostcert.pem`. For more information on possible host certificate locations see the GSI C Developer's Guide.
host credentials	The combination of a host certificate and its corresponding private key.

P

private key	The private part of a key pair. Depending on the type of certificate the key corresponds to it may typically be found in `$HOME/.globus/userkey.pem` (for user certificates), `/etc/grid-security/hostkey.pem` (for host certificates) or `/etc/grid-security/<service>/<service>key.pem` (for service certificates).

	For more information on possible private key locations see this.
proxy certificate	A short lived certificate issued using a EEC. A proxy certificate typically has the same effective subject as the EEC that issued it and can thus be used in its place. GSI uses proxy certificates for single sign on and delegation of rights to other entities. For more information about types of proxy certificates and their compatibility in different versions of GT, see http://dev.globus.org/wiki/Security/ProxyCertTypes.
proxy credentials	The combination of a proxy certificate and its corresponding private key. GSI typically stores proxy credentials in `/tmp/x509up_u<uid>` , where <uid> is the user id of the proxy owner.

S

server	A process that receives commands and sends responses to those commands. Since it is a server or service, and it receives commands, it must be listening on a port somewhere to receive the commands. Both FTP and GridFTP have IANA registered ports. For FTP it is port 21, for GridFTP it is port 2811. This is normally handled via inetd or xinetd on Unix variants. However, it is also possible to implement a daemon that listens on the specified port. This is described more fully in in the Architecture section of the GridFTP Developer's Guide.
service credentials	The combination of a service certificate and its corresponding private key.

U

user credentials	The combination of a user certificate and its corresponding private key.