

GT 4.2.1 CAS Admin Guide

GT 4.2.1 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 [Installing GT 4.2.1](#). 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 to the documentation for [CAS client install \[1\]](#).

Table of Contents

1. Building and Installing	1
2. Configuring	2
1. Configuration overview	2
2. Loading the CAS service at start up	2
3. Configuring the VO Description	3
4. Configuring the maximum assertion lifetime	3
5. Configuring database backend	3
6. Configuring security descriptor	3
7. Configuring with a GridFTP Server	4
8. Configuring CAS to manage policy for web service.	4
9. CAS auto-registration with default WS MDS Index Service	4
10. Registering CAS manually with default WS MDS Index Service	6
3. Deploying	7
1. Obtaining credentials for the CAS service	7
2. Database installation and configuration	7
3. Deploying into Tomcat	9
4. Setting up CAS for GridFTP	10
1. Overview	10
2. Files and directories in CAS	10
3. Configuring the CAS server to work with the GridFTP server	11
4. Transferring files using CAS and GridFTP	14
5. Managing policy	14
6. Authorization Enforcement at the GridFTP Server	15
5. Testing	16
1. Testing the back end database module	16
2. Testing the CAS service module	17
6. Example of CAS Server Administration	20
1. Adding a user group	20
2. Adding a trust anchor	21
3. Adding users	21
4. Adding users to a user group	21
5. Adding a new FTP server	21
6. Creating an object group	22
7. Adding members to an object group	22
8. Adding service types	22
9. Adding action mappings	23
10. Granting permissions	23
7. Example of using CAS Service for Web Service Policy Management	24
1. Example of using CAS Service as Authorization Service	24
2. Example of using CAS Service as Local PDP	26
8. Security Considerations	30
1. Security Considerations for CAS	30
9. Debugging	31
1. Logging in Java WS Core	31
10. Troubleshooting	33
1. Error Messages	34
2. Credential Troubleshooting	35
Glossary	38

List of Tables

2.1. Database parameters	3
2.2. Mapping from web services object to CAS object	4
3.1. Command line options	8
5.1. Test database properties	16
5.2. Test properties	17
7.1. Entity Mapping	24
7.2. Operation Details	27
10.1. WS A&A Authorization Framework Error Messages	34
10.2. Credential Errors	36

Chapter 1. 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 [Installing GT 4.2.1](#). No extra installation steps are required for this component.

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

Chapter 2. Configuring

1. Configuration overview

The CAS service can be configured with the following :

- server start up configuration
- a description of the VO the CAS service serves
- the maximum lifetime of the assertions it can issue
- information about the back end database it uses. Any database with a JDBC driver and reasonable SQL support can be used. The schema that works with Derby database, MySQL and PostGres is distributed and can be found at `$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema`.
- the security settings of the service and 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.

2. Loading the CAS service at start up

By default, the CAS service is not loaded at start up. To change this behavior, uncomment the *loadOnStartup* property set in `$GLOBUS_LOCATION/etc/globus_cas_service/server-config.wsdd` as shown below.

Once the *loadOnStartup* property is uncommented, the following happens at start up:

1. The CAS service is loaded.
2. The database connection pool is initialized.
3. The service registers itself to the default MDS Index Service.

```
<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>
```

3. Configuring the VO Description

To change the VO description, set the parameter `voDescription` in `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` to the desired values.

4. Configuring the maximum assertion lifetime

To change the maximum assertion lifetime set the parameters `maxAssertionLifetime` in `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` to the desired values.

5. Configuring database backend

To alter the configuration of the database back end edit the `databaseConfiguration` section of `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` as described below. If you are using the default Derby installation, the only parameter to change is the `connectionURL` to replace `GLOBUS_LOCATION` with the actual location of your toolkit installation.

Table 2.1. Database parameters

<code>driver</code>	The JDBC driver to be used
<code>connectionURL</code>	The JDBC connection url to be used when connecting to the database
<code>userName</code>	The user name to connect to the database as
<code>password</code>	The corresponding database password
<code>activeConnections</code>	The maximum number of active connections at any given instance
<code>onExhaustAction</code>	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)
<code>maxWait</code>	The maximum time in milliseconds that the pool will wait for a connection to be returned
<code>idleConnections</code>	The maximum number of idle connections at any given time

6. Configuring security descriptor

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 which 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 back end database to determine if the call is permitted.

To alter the security descriptor configuration refer to [Security Descriptors](#). The file to be changed is `$GLOBUS_LOCATION/etc/globus_cas_service/security-config.xml`.

 **Note**

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

 **Important**

If the service is configured to use GSI Secure Transport, then container credentials are used for the handshake, irrespective of whether service level credentials are specified.

7. Configuring with a GridFTP Server

CAS is used to administer access rights to files and directories and the GridFTP server can be configured to enforce those rights.

For detailed information about configuring CAS for use with a GridFTP server, see [How to Set up CAS with GridFTP](#).

8. Configuring CAS to manage policy for web service.

The CAS server can be used to administer rights for access to web services. The mapping from CAS objects to the web service resource is shown on this table:

Table 2.2. Mapping from web services object to CAS object

Object	EPR of WS resource as string. The OGSA-AuthZ specification defines how a EPR can be represented as a string and a utility for such is provided at <code>org.globus.wsrf.impl.security.EPRUtil</code> .
Object namespace	The object namespace is used to get both a comparison algorithm and the basename. For web services policy we need exact comparison and also don't have any base name. An implicit namespace <code>casDefaultNs</code> with the required properties is added to the service.
Service type	The OGSA-AuthZ specification defines a service type to use for web services operation as "http://www.gridforum.org/namespaces/2003/06/ogsa-authorization/saml/action/operation" This is defined as a constant in <code>org.globus.wsrf.impl.security.authorization.SAMLAuthorizationConstants</code> and is added implicitly.
Action	This is the actual operation on the webservice. For example method "add" on Counter Service.

An example scenario is described [here](#).

9. CAS auto-registration with default WS MDS Index Service

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

 **Note**

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

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

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

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

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

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

9.1. Configuring resource properties

By default, the `VoDescription` resource property (which describes the virtual organization relevant to the CAS Service) is sent to the default Index Service.

You can configure which resource properties are sent in the registration.xml file, `$GLOBUS_LOCATION/etc/globus_cas_service/registration.xml`. The following is the relevant section of the file:

```
<Content xsi:type="agg:AggregatorContent "
  xmlns:agg="http://mds.globus.org/aggregator/types">
  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
  <agg:GetResourcePropertyPollType
  xmlns:cas="http://www.globus.org/07/2004/cas">
  <!-- Specifies that the index should refresh information
  every 8 hours (28800000ms) -->
  <agg:PollIntervalMillis>28800000</agg:PollIntervalMillis>
```

```
<!-- specifies that all Resource Properties should be
collected from the RFT factory -->

<agg:ResourcePropertyName>cas:VoDescription</agg:ResourcePropertyName>

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

10. Registering CAS manually with default WS MDS Index Service

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

Chapter 3. Deploying

The CAS service is deployed as a part of a standard toolkit installation. Please refer to the [Installing GT 4.2.1](#) for details. Other than the steps described in the above guide, the following are needed to deploy the CAS service.

1. 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](#)¹.

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 the default identity is not used. The command in the above mentioned [web page](#)² may be altered as follows to get credentials for the 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 locations of certificate and key files are referred to as `CAS_CERT_FILE` and `CAS_KEY_FILE`, respectively. The subject name in these credentials is expected by CAS clients by default.

2. Database installation and configuration

CAS uses a back end 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 back end.

2.1. Installing the database

While any database with a JDBC driver and support for a reasonable set of SQL may be used, Derby has been used for development and testing. The driver for the same is included in the distribution. A default Derby database is created for CAS during installation and the CAS server uses the database in the embedded mode. One set of CAS tests also use the database in the embedded mode and another set uses in the network mode.

Note

Any JDBC compliant database might be used as backend. If a different database is used, the corresponding driver should be added to `$GLOBUS_LOCATION/lib`.

2.2. Creating the CAS database

By default a Derby database is created during deploy time at `$GLOBUS_LOCATION/var`. But schema for using PostGres or MySQL as backend is provided at `$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema/cas_pgsql_database_schema.sql` or `$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema/cas_mysql_database_schema.sql`. Please refer to PostGres or MySQL documentation for steps involved in creating the database.

¹ <http://www.globus.org/toolkit/docs/2.4/admin/guide-verify.html#ldapcert>

² <http://www.globus.org/toolkit/docs/2.4/admin/guide-verify.html#ldapcert>

 **Note**

This documentation assumes the database name to be `casDatabase`

2.3. Bootstrapping the CAS database

The CAS database needs to be initialized with data specific to CAS and information about a super user to allow bootstrapping 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 3.1. Command line options

<code>-help</code>	Prints the help message.
<code>-debug</code>	Runs the script with debug trace.
<code>-d dbProperties-File</code>	File name with database properties as follows: <code>dbDriver=database driver name</code> <code>dbConnectionURL=database connection URL</code> <code>dbUserName=Username to access database</code> <code>dbPassword=Password for the above username</code>
<code>-b bootstrapFile</code>	This option populates the database with super user data and points to a file with data to use for bootstrapping the database. A template file for this can be found at <code>\$GLOBUS_LOCATION/share/globus_cas_service/bootstrapTemplate</code> and a sample file can be found at <code>\$GLOBUS_LOCATION/share/globus_cas_service/bootstrapSample</code> .
<code>-implicit</code>	Populates the 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; and 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. The values shown here are for default Derby database set up. If using any other database configuration, please fill in or uncomment corresponding values as appropriate.

- `$GLOBUS_LOCATION/share/globus_cas_service/casDbProperties`

```
# Database driver to use
dbDriver=org.apache.derby.jdbc.EmbeddedDriver
# Connection URL to database
dbConnectionURL=jdbc:derby:casDatabase
# Database user name
dbUsername=casAdmin
```

```
# Database password
dbPassword=foobar
```

- `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapProperties`. Copy the template file `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapTemplate` as `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapProperties` and fill in the values as described below. Refer `$GLOBUS_LOCATION/share/globus_cas_service/bootstrapSample` for a sample.

```
# This file is used to populate the backend CAS database with a super user.
# Nick name for trust anchor for super user
ta-name=defaultTrustAnchor
# Trust Anchor authentication method, typically X509 Certificates
ta-authMethod=X509
# Trust Anchor authentication data, typically X509 DN
ta-authData=/C=US/O=Globus/CN=Default CA
# Super user nickname
user-name=superUser
# Super user subject name, typically DN
user-subject=/O=Grid/O=Globus/OU=something/CN=suUser
# Super user group name
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](#).

3. Deploying into Tomcat

CAS has been tested to work without any additional setup when deployed into Tomcat. Please follow these [basic instructions](#) to deploy GT4 services into Tomcat. Note that the Java WS Core module needs to be built and configured as described in the previous sections.

Chapter 4. How to Set Up CAS to Use with GridFTP

This document is intended to be used with the [CAS](#) and [GridFTP](#) documents, to configure and use CAS with GridFTP.

1. Overview

CAS is used to administer access rights to files and directories and GridFTP server can be configured to enforce those rights. The following is an overview of the process:

1. A CAS administrator sets up rights on the CAS server.
2. A user retrieves those rights in the form of a signed assertion and presents it to the GridFTP server at the time of access.
3. If the assertion presented has the necessary permissions and the GridFTP server trusts the CAS server that issued it, access is allowed.

In a default Globus Toolkit installation, CAS is installed with a set of credentials associated with the CAS server. A CAS administrator has all rights on the CAS server and manages users. To be used with GridFTP, the administrator needs to add users, add the files and directories he wants protected and grant specific rights to the files.

The user then retrieves a CAS assertion from the server, for either specific files/directories or all his access rights stored in CAS. This is done using the client side command, [cas-proxy-init](#), which embeds the assertion in the user's proxy. Another client side command, [cas-wrap](#) can then be used to run the GridFTP command, [globus-url-copy](#), to ensure that the proxy is used.

Details about the CA who issues these credentials needs to be added to the CAS server as a trust anchor. Each user has a nick name, a subject DN and is associated with a trust anchor. A particular user is designated as superuser, so has permissions to add users and manager rights. This can be set up using the [bootstrap](#) step.

Users who are not administrators can be added using [admin command line tools](#). Permissions in a CAS database are granted to user groups, rather than users. So a user group needs to be created and users added to it prior to assigning permissions.

2. Files and directories in CAS

Files and directories are "objects" in CAS. Each object has a name and a namespace associated with it. The namespace determines the scope of the object and also the comparison method to use with object names. A predefined namespace, called FTPDirectoryTree is loaded in the CAS database upon bootstrap and this is the namespace expected for GridFTP objects.

For example, if permission on file "some_file_path" on host "somehost.edu" needs to be managed, it needs to be added as an object, with name "ftp://somehost.edu/some_file_path" and namespace "FTPDirectoryTree". This object "ftp://somehost.edu/some_file_path" will be recognized by the CAS-enabled GridFTP server at "somehost.edu" as referring to the file named "some_file_path". An object in the FTPDirectoryTree namespace with the name "ftp://somehost.edu/some_directory_path/*" will be recognized by the CAS-enabled GridFTP server at "somehost.edu" as referring to all files and directories under "some_directory_path".

In some cases, it may be desirable to have a GridFTP server recognize CAS assertions that use a hostname other than the server's fully qualified domain name. Starting the GridFTP server with the option "-H otherhost" will cause the GridFTP server to recognize objects with names that start with "ftp://otherhost/" instead.

The actions that are allowed on a file and a directory are listed in the following table:

Action	Result for a file	Result for a directory
read	Gives permission to read the file.	Gives permission to chdir to the directory.
lookup	Gives the right to get Unix stat() information.	Gives the right to chdir to and to list the contents of the directory.
write	Allows modification of an existing file.	Gives the right to chdir to the directory.
create	Allows creation of the file if it does not exist.	Allows creation of the directory if it does not exist and gives the right to chdir to the directory if it does exist.
delete	Allows deletion of the file.	Allows deletion of the directory, if empty; also gives the right to chdir to the directory.
chdir	Does not apply.	Allows making the directory the current default directory.

These actions are predefined and loaded onto CAS database when bootstrap is done. Apart from these actions on files/directories, there is another action "authz_assert" which is used to override the existing assertions (either from proxy or from a previous "authz_assert" command) with the assertions received over the GridFTP control channel. FTP clients can use the command "SITE AUTHZ_ASSERT" to send assertions to the GridFTP server.

The following is a summary of supported FTP commands and the permissions they require:

Typical Client Command	FTP Protocol Command	Rights Required
get	RETR	read
put	STOR	write, if file exists; create, if file does not exist
delete	DELE	delete
ls	LIST	lookup
chdir	CWD	any of: chdir, lookup, read, write, create, or delete
mkdir	MKD	create
rmdir	RMD	delete
rename	RNFR / RNTD	read and delete on old file; write on new file, if it exists, create on new file, if it does not exist

3. Configuring the CAS server to work with the GridFTP server

3.1. Step 1: Set up the CAS server.

This must be done by user "casAdmin".

3.2. Step 2: Run the CAS server with host credentials

Run the CAS server with host credentials `/O=Grid/OU=test/CN=foo.bar.edu`.



Note

The CA that issues these credentials should be trusted by the GridFTP server. Setting up of trusted CA is described [here](#)

3.3. Step 3: Enable CAS support in the GridFTP server

The GridFTP Server reads two files (`gsi-authz.conf` and `gsi-gaa.conf`) to determine how to perform certain authorization and mapping functions. If these files are not present (as is the case after a standard Globus Toolkit installation), the GridFTP server will not support CAS authorization (that is, the GridFTP server will ignore the CAS policy assertions in the user's credential and determine the user's permissions based solely on the user's identity).

Run **[setup-globus-gaa-authz-callout]** to create `gsi-authz.conf` and `gsi-gaa.conf` files that will cause the GridFTP server to honor CAS policy assertions. There are two ways to run this command:

1. To create the config files in the directory `/etc/grid-security` (where the GridFTP server looks for them by default), run the following as root:

```
$GLOBUS_LOCATION/setup/globus/setup-globus-gaa-authz-callout
```

2. To create the config files to another directory, run:

```
$GLOBUS_LOCATION/setup/globus/setup-globus-gaa-authz-callout -d mydir
```

Where 'mydir' is the path to the desired directory. You then must make sure the GridFTP server finds these files by setting the following environment variable *before* starting the GridFTP server:

- Set `GSI_AUTHZ_CONF` to `mydir/gsi-authz.conf`.
- Set `GSI_GAA_CONF` to `mydir/gsi-gaa.conf`.

By default, `setup-globus-gaa-authz-callout` will not overwrite an existing configuration file. Use the following options to overwrite existing GridFTP config files:

- Use the `-force` option to overwrite an existing `gsi-authz.conf` file
- Use the `-overwrite_gaa_config` option to overwrite an existing `gsi-gaa.conf` file.

3.4. Step 4: Configure the GridFTP server to trust the CAS server

The previous step configured the GridFTP server to understand CAS credentials. However, the GridFTP server will not allow a user authenticating with a CAS credential to perform any action that it would not allow the CAS server itself to perform. To configure the GridFTP server to trust a particular CAS server:

1. Create a local user account corresponding to the CAS server.
2. Use file permissions to allow that user account to have the desired level of file access.

3. Create a gridmap entry that maps the CAS server's distinguished name to that local account.

3.5. Step 5: Add a CAS administrator

Add a CAS admin, we'll use the CAS nickname, "casAdmin", and DN "/O=Grid/OU=test/CN=CAS Administrator". This credential is issued by a CA with certificate "/O=Grid/OU=test/CN=CA"

3.5.1. Using bootstrap to add a CAS administrator

This can be setup during [bootstrap](#) and below is the bootstrap file for the above scenario.

```
ta-name=defaultTrustAnchor
ta-authMethod=X509
ta-authData=/O=Grid/OU=test/CN=CA
user-name=superUser
user-subject=/O=Grid/OU=test/CN=CAS Administrator
userGroupname=superUserGroup
```

3.6. Step 6: Add a CAS user group

Since permissions are only on user groups, we need to add a user group, for example "readGroup". This can be done using [cas-group-admin](#).

```
cas-group-admin user create superUserGroup readGroup
```



Note

The superUserGroup here determined the user group that has permissions to manipulate the group that is being created i.e readGroup.

3.7. Step 7: Add a CAS user

Add the user who needs access to files. For example, add User1, with DN "/O=Grid/OU=test/CN=User 1" issued by the CA in [Step 4], using [cas-enroll]:

```
cas-enroll user superUserGroup User1 "/O=Grid/OU=test/CN=User 1" defaultTrustAnchor
```



Note

The superUserGroup here determined the user group that has permissions to manipulate the user that is being created i.e User1.

3.8. Step 8: Add user to the user group

To add the CAS user to the CAS user group, use [cas-group-add-entry](#):

```
cas-group-add-entry user readGroup User1
```

3.9. Step 9: Add CAS object

Add the file (on which you want to set permissions) as a CAS object with the name "ftp://foo.bar.edu/tmp/file1" and namespace "FTPDirectoryTree":

```
cas-enroll object superUserGroup "ftp://foo.bar.edu/tmp/file1" "FTPDirectoryTree"
```

 **Note**

The `superUserGroup` here determined the user group that has permissions to manipulate the object that is being created.

3.9.1. Giving permissions to all files in a directory

To give permissions to all files in a directory, create the CAS object with a wild card `"*"`. For example, object name `"ftp://foo.bar.edu/tmp/admin/*"`. If permission is given on this object, all files in that directory are affected.

For example, if you create a new user group called `"adminGroup"`, you give read permission to all users in that group as follows, any user in `"adminGroup"` can read any file in `"/tmp/admin"` on that server.

```
cas-rights-admin grant adminGroup object FTPDirectoryTree
    "ftp://foo.bar.edu/tmp/admin/*" serviceAction file read
```

3.10. Step 10: Add permission for users

Add permission for users in `"readGroup"` to be able to read object added in the previous step.

```
cas-rights-admin grant readGroup object FTPDirectoryTree
    "ftp://foo.bar.edu/tmp/file1" serviceAction file read
```

Now any user added to `readGroup` can read the file (that was added in Step 9).

4. Transferring files using CAS and GridFTP

Once the CAS server has been configured (as above), if `User1` wants to transfer the file `"/tmp/file1"` on `"ftp://foo.bar.edu"`, `User1` performs the following two steps:

1. Runs **cas-proxy-init** to get the CAS assertion:

```
cas-proxy-init -p casProxy
```

 **Note**

This gets the assertion from the CAS server, generates a proxy with the assertion and writes it out to `"casProxy"`.

2. Runs **cas-wrap** to transfer the file:

```
cas-wrap -p casProxy globus-url-copy gsiftp://somehost.edu/some_file_path
file:///some_file_path
```

5. Managing policy

The CAS administrator can create groups of users and groups of objects to ease policy management.

Also, if the administrator wants to provide a set of rights, such as read and lookup, the administrator can create a service action group with these action as members. User groups can then be provided with access to all actions in the group.

Similarly the administrator can create groups of objects, which would be a group of GridFTP server and files on it, and grant access rights to users.

6. Authorization Enforcement at the GridFTP Server

The following describes how the GridFTP server processes a file transfer using CAS:

1. During the authentication phase, GridFTP server extracts the CAS assertion in the proxy credentials and stores it.
2. If the GridFTP server receives a CAS assertion over the control channel, it overwrites the old assertion (if any) with the new one.
3. Upon receiving a file transfer request, the GridFTP server looks for a CAS assertion.
4. If the assertion is present, checks whether the file being accessed has permission. If assertion has permission for the file, the access is allowed. Otherwise an error is thrown.
5. If no CAS assertion is found, the presence of user DN is checked in the gridmap file. If DN exists, then access is allowed. Otherwise an error is thrown.

Chapter 5. Testing

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

Assumptions:

- A back end database has been set up and configured. By default Derby is set up and configured.
- The CAS service and tests are installed in *\$GLOBUS_LOCATION*.
- The sample commands assume:
 1. The container is started up on localhost and port 8443.
 2. The default Derby database configuration is used, i.e database name is *casDatabase* and user is *casAdmin*

1. Testing the back end database module

1. Run:

```
cd $GLOBUS_LOCATION
```

2. Populate the file *etc/globus_cas_unit_test/casTestProperties* with the database configuration information shown in the following table.

Table 5.1. Test database properties

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

If default Derby configuration is required,

- a. Uncomment the lines in *etc/globus_cas_unit_test/casTestProperties* following the comment *# Uncomment below for Derby Embedded Driver.*
 - b. Edit *dbConnectionURL* to replace *GLOBUS_LOCATION* with path to your installation. For example, if your *GLOBUS_LOCATION* is */temp/globus*, then the property should look like *dbConnectionURL=jd-bc:derby:/temp/globus/var/casDatabase*
 - c. Comment out all other occurrences of these properties
3. The database needs to be empty for these tests to work and will be deleted by this target. Run:

```
set GLOBUS_LOCATION=C:\globus
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.

2. Testing the CAS service module

These tests can be set up so as to be able to test multiple user scenarios 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 a second user. As the second user the permissions and queries are tested to ensure that the setup worked.

As with the previous any database backend can be used for testing. If the default Derby database is used, it needs to be configured in network mode since these tests require that both the tested CAS server and the tests use the database. Two files need to be modified for configuring these tests and they are described below.

1. The `etc/globus_cas_unit_test/casTestProperties` needs to be configured with appropriate database information and service information.

The database information is configured similar to previous test, but for the default Derby installation, the network driver needs to be used.

- a. Uncomment the lines in `etc/globus_cas_unit_test/casTestProperties` following the comment *# Uncomment below for Derby Network Driver.*
- b. Edit `dbConnectionURL` to replace `GLOBUS_LOCATION` with path to your installation. For example, if your `GLOBUS_LOCATION` is `/temp/globus`, then the property should look like `dbConnectionURL=jd-bc:derby:/temp/globus/var/casDatabase`
- c. Comment out all other occurrences of these properties.

Now edit the server specific information as shown below:

Table 5.2. Test properties

<code>user1SubjectDN</code>	The DN of the user running the first set of tests.
<code>user2SubjectDN</code>	The DN of the user running the second set of tests. This DN has to be different from the value specified for <code>user1SubjectDN</code> . Note: Both tests can be run as the same user as long as the DN of the certificate being used to run the tests matches the value specified in <code>user1SubjectDN</code> . In this case, the value of <code>user2SubjectDN</code> can be set to an arbitrary string.
<code>maxAssertionLifetime</code>	Should match the value set in the service configuration as shown in Configuration Information .
<code>host</code>	Host on which the CAS service is running.
<code>port</code>	Port on which the CAS service is running.
<code>securityType</code>	This is an optional parameter indicating the security type to use. Can be set to <code>message</code> for Secure Message or <code>conversation</code> for Secure Conversation or <code>transport</code> for Secure Transport (the default configuration).
<code>protType</code>	This is an optional parameter indicating the protection type to use. Can be set to <code>signature</code> for integrity protection (the default configuration) or <code>encryption</code> for privacy protection.

2. Edit The `etc/globus_cas_unit_test/jndi-config.xml` to replace `GLOBUS_LOCATION` with the actual Globus Location of your install.

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 [CAS Test Properties](#).

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. Now 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 and all entries need to be deleted.

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

 **Important**

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

Chapter 6. Example of CAS Server Administration

The following contains an example of administering the CAS server policies using the CAS administrative clients described. *FILLME: add olink 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 the 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 environment 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`.

1. 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 the 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 *scientists*.

2. 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.

3. 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 manipulate 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.

4. 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 nickname 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.

5. 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 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 the 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 *FTP-DirectoryTree* 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
```

6. 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.

7. 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 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.

8. 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
```

9. 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 the `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.

10. 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.

Chapter 7. Example of using CAS Service for Web Service Policy Management

The CAS server can be used to store and manage policy for web services. The web service authorization decisions will require policy on a user accessing a particular operation on a said resource. Entities map to the CAS model as follows:

Table 7.1. Entity Mapping

User	A user group in CAS service with user as member.
Operation	A Service/Action entry is used to represent a web service operation. <ul style="list-style-type: none">• Service type is the OGSA-AuthZ URI for operation "http://www.gridforum.org/namespaces/2003/06/ogsa-authorization/saml/action/operation". This is added to the CAS server as an implicit object.• Action is the local part of the web service operation QName
WS Service/Resource	The EPR of resource represented as string. The EPR is converted to string using the algorithm specified in the OGSA AuthZ specification.

The next sections discuss a few different ways the CAS server can be used to manage web services access policy.

1. Example of using CAS Service as Authorization Service

This section describes using the CAS server as an authorization service to manage policy for web services. The sample service used here is the Counter Service distributed as a part of Java WS Core component. The CAS server needs to be bootstrapped with information about a super user and the sample assumes that the super user is the client. The methods on Counter Service that need to be protected need to be added as actions and the EPR of the created counter resource needs to be added as an object.

1. Configure the CAS service as described in [here](#)
2. Bootstrap the CAS service as described in [here](#). Ensure that the *user-subject* in the bootstrap properties file to be the DN of the client. For example, following is a sample bootstrap properties file

```
ta-name=ta
ta-authMethod=X509
ta-authData=/C=US/O=Globus/CN=Default CA
user-name=su
user-subject=/C=US/O=Globus Alliance/OU=User/CN=101497d3dcd.3dcd5aef
userGroupname=superUserGroup
```

3. For this example, configure the following sample descriptor as the service security descriptor for Secure Counter Service. It configures the relevant PDP to be used to leverage CAS for authorization.

Example of using CAS Service for Web Service Policy Management

- a. Download [sample security descriptor](#)¹ and save to `$GLOBUS_LOCATION/etc/globus_wsrf_core_samples_counter`.
 - b. Edit the descriptor file to have the identity of the authorization service as value of parameter `authzServiceIdentity`.
 - c. To configure it as descriptor, edit `$GLOBUS_LOCATION/etc/globus_wsrf_core_samples_counter/server-config.wsdd` and edit description for `SecureCounterService` to be as follows: `<parameter name="securityDescriptor" value="etc/globus_wsrf_core_samples_counter/counter-cas-security-config.xml" />`
4. Start up container with `-nosec` option.
- ```
bin/globus-start-contianer -nosec
```
5. In another shell, set the `CAS_SERVER_URL` environment variable to point to the CAS service.
- ```
set CAS_SERVER_URL=http://localhost:8080/wsrf/services/CASService
```
6. Also set the `CAS_SERVER_IDENTITY` environment variable to point to the identity of the CAS service..
- ```
set CAS_SERVER_IDENTITY="/C=US/O=Globus Alliance/OU=User/CN=101497d3dcd.3dcd5aef"
```
7. Create a user group (`testUGp`) to give permission to access counter service.
- ```
bin\cas-group-admin.bat -m msg user create superUserGroup testUGp
```
8. Add super user to the `testUGp`. The nick name for super user would have been specified in the bootstrap properties file. In this sample it is `su`.
- ```
bin\cas-group-add-entry.bat -m msg user testUGp su
```
9. Add "createCounter" (an operation on counter service) as an action of the above service type.
- ```
bin\cas-action.bat -m msg add "http://www.gridforum.org/namespaces/2003/06/ogsa-authori
```
10. Add "add" (an operation on counter resource) as an action of the above service type.
- ```
bin\cas-action.bat -m msg add "http://www.gridforum.org/namespaces/2003/06/ogsa-authori
```
11. Add "destroy" (an operation on counter resource) as an action of the above service type.
- ```
bin\cas-action.bat -m msg add "http://www.gridforum.org/namespaces/2003/06/ogsa-authori
```
12. The create counter operation is on the secure counter service (rather than any resource). The security descriptor will use the CAS service to authorize requests on the service. Hence policy needs to be written for the createCounter operation. First step is the enroll the `SecureCounterService` endpoint as an object in CAS.

¹ `../counter-cas-security-config.xml`

```
bin\cas-enroll.bat -m msg object testUGp http://192.168.1.101:8080/wsrf/services/Secure
```

13. In this command, policy for users in testUGp to invoke createCounter operation on the SecureCounterService endpoint.

```
bin\cas-rights-admin.bat -m msg grant testUGp object casDefaultNS http://192.168.1.101:
```

14. Now we create a new counter resource using the counter client and use the -b option to get the string representation of EPR. The string representation of EPR is used to add rights on the counter resource that was created in this command.

```
bin\counter-create.bat -s http://localhost:8080/wsrf/services/SecureCounterService -m m
```

15. Enroll the counter resource as an object. The object name will be the string representation of the EPR from previous step. Since we don't want any basename for this resource, use *casDefaultNS* as the namespace.

```
bin\cas-enroll.bat -m msg object testUGp "http://192.168.1.101:8080/wsrf/services/Secur
```

16. Grant rights for *testUGp* to be able to add to the counter resource.

```
bin\cas-rights-admin.bat -m msg grant testUGp object casDefaultNS "http://192.168.1.101
```

17. Attempt adding to the counter resource.

```
bin\counter-add.bat -e epr -m msg 10 -z none
```

18. Attempt destroying the resource. You will see an authorization exception.

```
bin\wsrf-destroy.bat -m msg -z one -e epr
```

2. Example of using CAS Service as Local PDP

This section describes using the CAS server as a PDP local to the container to manage authorization policy for web services. Unlike the previous section the authorization service interface of CAS is not used. But a PDP, `org.globus.cas.impl.LocalCasPdp`, distributed with the CAS server code base is used to contact a co-hosted CAS server for authorization decisions. This PDP uses Java calls on the CAS resource to obtain policy information and returns a decision of PERMIT or DENY.

For a service to use this feature, the authorization configuration at service/resoruce level should be the `LocalCasPdp`. To grant access to users, the users need to be added to the CAS database. This can be done either during bootstrap or during runtime. The resource owner can then facilitate fine grained, per operation authorization by storing appropriate permissions on the local CAS server.

Following is a list of steps that highlights how the `CasLocalPdp` can be sued:

- The test service used to as a part of this distribution to test this feature is used as the example service. The service consists of three methods, with following characteristics.

Table 7.2. Operation Details

Name	Purpose	Authentication Re-quired	Authorization Required	Invocation on ser-vice/resource
create	Create a new resource and return EPR. Initialize with integer value and an array of CAS user nickname allowed to access new resource.	Yes, any scheme	None	Service
getValue	Return current value of resource	No	No	Resource
add	Integer value as parameter to add to current value	Yes, any scheme	LocalCasPdp	Resource

- To configure the test service to use enforce authentication for create method, a security descriptor as shown below can be configured.

```
<serviceSecurityConfig xmlns="http://www.globus.org/security/descriptor/service">
  <auth-method>
    <GSISecureConversation/>
    <GSISecureMessage/>
    <GSISecureTransport/>
  </auth-method>
  <authzChain>
    <pdp>
      <interceptor name="none"/>
    </pdp>
  </authzChain>
</serviceSecurityConfig>
```

- To configure the test resource to enforce authentication and use LocalCasPdp for authorization, a security descriptor file can be written and loaded into the resource object as shown below

```
<serviceSecurityConfig xmlns="http://www.globus.org/security/descriptor/service">
  <auth-method>
    <GSISecureConversation/>
    <GSISecureMessage/>
    <GSISecureTransport/>
  </auth-method>
  <methodAuthentication>
    <method name="getValue">
      <auth-method>
        <none/>
      </auth-method>
    </method>
  </methodAuthentication>
  <authzChain>
    <pdp>
      <interceptor name="pdp:org.globus.cas.impl.LocalCasPdp"/>
    </pdp>
  </authzChain>
</serviceSecurityConfig>
```

```
</authzChain>  
</serviceSecurityConfig>
```

Sample Resource code, where resourceSecDesc is the path to the above file.

```
public class TestResource implements SecureResource, ResourceIdentifier {  
  
    ResourceSecurityDescriptor desc;  
  
    public TestResource(int val, String resourceSecDesc) throws Exception {  
        this.key = new Integer(hashCode());  
        this.value = val;  
        this.desc = new ResourceSecurityDescriptor(resourceSecDesc);  
    }  
  
    public ResourceSecurityDescriptor getSecurityDescriptor() {  
  
        return this.desc;  
    }  
}
```

- The owner of the service should exist in the CAS database with the same DN used to run the service. Also, a user group to add all new CAS objects created is required. For the purpose of this sample, we assume that "testUser" exists on the server and a user group "testUserGroup". Refer to [Section 2.3, "Bootstrapping the CAS database"](#) for details on how to bootstrap the CAS server.
- A Service Action group with all operations in the service should be added to the CAS service. This needs to be done only once and can be done in the resource home. This example shows how a new group "testActionGp" can be added.

```
String casUrl = "local://host:port/wsrf/services/CASService";  
Util util = new Util(casUrl);  
System.out.println("Creating service action group");  
util.createServiceActionGroup("testUserGroup", "testActionGp");
```

Note that local invocations can be used, since the CAS server is expected to be cohosted.

- All operation on the service should be added to the service/action group.

```
util.addOperation(new String[] { "getValue", "add"},  
                  "testActionGp");
```

- A User group per resource created can be added to the CAS server. This example uses the resource key as part of the group name.

```
String newUserGroupName = "objGp-" + key.toString();  
util.createUserGroup(userGroup, newUserGroupName);
```

Example of using CAS Service for Web Service Policy Management

- The EPR of resource should be added as a CAS object. The EPR should be converted to string representation and added with default namespace.

```
String eprAsString = EPRUtil.getEPRAsString(epr);  
// The user group on which all rights are granted for this object can  
// be the new user group created in previous step.  
util.addResourceObject(eprAsString, "testUserGroup");
```

- With all CAS objects bootstrapped, permission for the new user group to perform actions in new service/action group on resource should be added to CAS server. Now any new user who needs to be allowed access just needs to be added to CAS server and then added to user group for the said resource.

```
// set permission for new user group, resource EPR and then new action  
// group.  
util.grantPermission(newUserGroupName, eprAsString,  
                    actionGroupName);
```

- On a create operation, when the list of authorized users is passed as parameter, the user should be added to the newly created user group.

```
util.addUserToGroup(newUserGroupName, "authorizedUserNick");
```

- With the above set up, a create call can be made by anyone, provided some authentication is used. `getValue` can be invoked insecurely also. `add` will require that the CAS server have policy about the user.

Chapter 8. Security Considerations

1. Security Considerations for CAS

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

Chapter 9. Debugging

1. Logging in Java WS Core

The following information applies to Java WS Core and all services built on Java WS Core.

Java WS Core server side has two types of loggers. One logger is used for development logging and by default writes to standard out. The other logger includes system administration information and is [CEDPs best practices](#)¹ compliant.

On client side, only developer logging is available and is configured using `log4j.properties`.

1.1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)² API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)³ as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)⁴.

1.1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁵. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

¹ <http://cedps.net/index.php/LoggingBestPractices>

² <http://jakarta.apache.org/commons/logging/>

³ <http://logging.apache.org/log4j/>

⁴ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁵ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

1.2. Configuring system administration logs

The specific logger to edit will be `log4j.logger.sysadmin` in `$GLOBUS_LOCATION/container-log4j.properties`. There you can configure the following properties:

```
log4j.appender.infoCategory=org.apache.log4j.RollingFileAppender
log4j.appender.infoCategory.Threshold=INFO
log4j.appender.infoCategory.File=var/containerLog
log4j.appender.infoCategory.MaxFileSize=10MB
log4j.appender.infoCategory.MaxBackupIndex=2
```

Above implies the logging file is rolling with each file size limited to 10MB and the logging information is stored in `$GLOBUS_LOCATION/var/containerLog`.

1.3. Sample log file

The [sample log file](#)⁶ contains many log entries for various scenarios in the Java WS container.


⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/sample-container-log.txt>

Chapter 10. Troubleshooting

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

1. Error Messages

Table 10.1. WS A&A Authorization Framework Error Messages

Error Code	Definition
<p>org.globus.cas.impl.databaseAccess.CasDBException, connection refused</p>	<p>If the CAS service fails with following error:</p> <pre> faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.userException faultSubcode: faultString: org.apache.commons.dbcp.DbcpException: Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TC you need to troubleshoot the connection to the CAS database.</pre>
<p>CAS clients fail with org.globus.cas.impl.databaseAccess.CasDBException</p>	<p>If CAS clients fail with database permission exceptions similar to:</p> <pre> [Caused by: ERROR: permission denied for relation service_type_action]; nested exception is:org.globus.cas.impl.databaseAccess.CasDBException:</pre> <p>, then there is something wrong with user permissions on the database.</p> <p> Note</p> <p>This is a specific instance of an error for the relation <i>service_type_action</i>. This error could be raised on any rel</p>

2. Credential Troubleshooting

2.1. Credential Errors

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

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

Table 10.2. Credential Errors

Error Code	Definition	Possible Solutions
Your proxy credential may have expired	Your proxy credential may have expired.	Use <code>grid-proxy-info</code> to check whether the proxy credential has actually expired. If it has, generate a new proxy with <code>grid-proxy-init</code> .
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.	Check the system clocks on the local and remote system.
Your end-user certificate may have expired	Your end-user certificate may have expired	Use <code>grid-cert-info</code> 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 <code>grid-proxy-destroy</code>) and creating a new one (with <code>grid-proxy-init</code>). Important: 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), <code>grid-proxy-init</code> will refuse to create a proxy certificate.	You can "fix" this by changing the permissions on the private key file. Important: However, you will still have a much more serious problem: it is 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	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 Installing GT 4.2.1 for details.
You may not trust the remote system's CA	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 Installing GT 4.2.1 for details.
There may be something wrong with the remote service's credentials	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 cannot find anything wrong with your credentials, check for the same conditions on the remote system (or ask a remote administrator to do so).

2.2. Some tools to validate certificate setup

2.2.1. grid-cert-diagnostics

The **grid-cert-diagnostics** program checks prints diagnostics about the user's certificates, and host security environment.

```
% grid-cert-diagnostics -p
```

2.2.2. Check that the user certificate is valid

```
openssl verify -CApath /etc/grid-security/certificates
-purpose sslclient ~/.globus/usercert.pem
```

2.2.3. Connect to the server using s_client

```
openssl s_client -ssl3 -cert ~/.globus/usercert.pem -key
~/.globus/userkey.pem -CApath /etc/grid-security/certificates
-connect <host:port>
```

Here `<host:port>` denotes the server and port you connect to.

If it prints an error and puts you back at the command prompt, then it typically means that the *server* has closed the connection, i.e. that the server was not happy with the client's certificate and verification. Check the SSL log on the server.

If the command "hangs" then it has actually opened a telnet style (but secure) socket, and you can "talk" to the server.

You should be able to scroll up and see the subject names of the server's verification chain:

```
depth=2 /DC=net/DC=ES/O=ESnet/OU=Certificate Authorities/CN=ESnet Root CA 1
verify return:1
depth=1 /DC=org/DC=DOEGrids/OU=Certificate Authorities/CN=DOEGrids CA 1
verify return:1
depth=0 /DC=org/DC=doegrids/OU=Services/CN=wiggum.mcs.anl.gov
verify return:1
```

In this case, there were no errors. Errors would give you an extra line next to the subject name of the certificate that caused the error.

2.2.4. Check that the server certificate is valid

Requires root login on server:

```
openssl verify -CApath /etc/grid-security/certificates -purpose sslserver
/etc/grid-security/hostcert.pem
```

Glossary

some terms not in the docs but wanted in glossary: [scheduler](#)

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](#).

S

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

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

T

transport-level security Uses transport-level security (TLS) mechanisms.

GT 4.2.1 CAS User's Guide

GT 4.2.1 CAS User's Guide

Introduction

The CAS User's Guide provides general end user-oriented information.

Building on the Globus Toolkit® Grid Security Infrastructure (GSI), CAS allows resource providers to specify course-grained access control policies in terms of communities as a whole, delegating fine-grained access control policy management to the community itself. Resource providers maintain ultimate authority over their resources but are spared day-to-day policy administration tasks (e.g. adding and deleting users, modifying user privileges).

Table of Contents

1. Using CAS	1
1. How it works	1
2. Basic procedure for using CAS	1
3. Using Authorization Service Interface	2
I. CAS User Commands	3
cas-proxy-init	4
cas-wrap	7
II. CAS Admin Commands	10
cas-enroll	11
cas-remove	15
cas-action	18
cas-group-admin	20
cas-group-add-entry	24
cas-group-remove-entry	27
cas-rights-admin	30
2. Troubleshooting	33
1. Credential Troubleshooting	33
2. Error Messages	36
3. Information Regarding CAS Licensing	37
Glossary	39

List of Tables

2.1. Credential Errors	34
2.2. WS A&A Authorization Framework Error Messages	36

Chapter 1. Using CAS

1. How it works

1. A CAS server is initiated for a community: a community representative acquires a GSI credential to represent that community as a whole, and then runs a CAS server using that community identity.
2. Resource providers grant privileges to the community. Each resource provider verifies that the holder of the community credential represents that community and that the community's policies are compatible with the resource provider's own policies. Once a trust relationship has been established, the resource provider then grants rights to the community identity, using normal local mechanisms (e.g. grid map files and disk quotas, file system permissions, etc).
3. Community representatives use the CAS to manage the community's trust relationships (e.g., to enroll users and resource providers into the community according to the community's standards) and grant fine-grained access control to resources. The CAS server is also used to manage its own access control policies; for example, community members who have the appropriate privileges may authorize additional community members to manage groups, grant permissions on some or all of the community's resources, etc.
4. When a user wants to access resources served by the CAS, that user makes a request to the CAS server. If the CAS server's database indicates that the user has the appropriate privileges, the CAS issues the user a GSI restricted *proxy credential* with an embedded policy giving the user the right to perform the requested actions.
5. The user then uses the credentials from the CAS to connect to the resource with any normal Globus tool (e.g. GridFTP). The resource then applies its local policy to determine the amount of access granted to the community, and further restricts that access based on the policy in the CAS credentials, This serves to limit the user's privileges to the intersection of those granted by the CAS to the user and those granted by the resource provider to the community.

2. Basic procedure for using CAS

A typical CAS user will use only two CAS-specific commands: **cas-proxy-init**, which contacts a CAS server and obtains a credential, and **cas-wrap**, which wraps a grid-enabled client program, causing that client program to use the credential that was previously generated using **cas-proxy-init**. For example, a day in the life of a CAS user might look something like this:

1. In the morning, the user runs:

```
% grid-proxy-init
% cas-proxy-init -t tag
```

The first line generates a Globus proxy credential; the second uses that credential to contact the CAS server and retrieve a CAS credential that includes all the permissions granted to the user by the community. The *tag* argument can be any string and is used to assign a name for that credential (**cas-wrap** uses this name to locate the credential).

2. Several times throughout the day, the user runs commands that look like:

```
% cas-wrap -t tag grid-enabled-program args
```

This runs the program **grid-enabled-program** with arguments *args*, using the CAS credential that had been created by and assigned the name *tag*.

For example:

```
% cas-wrap -t my-community gsincftp myhost.edu
```

looks for a CAS credential with the name "my-community" (e.g., a credential that had been created using "cas-proxy-init -t my-community") and then runs the command "gsincftp myhost.edu", causing the gsincftp program to use that CAS credential to authenticate.

3. At the end of the day, the user runs:

```
% cas-wrap -t tag grid-proxy-destroy
```

to destroy the CAS credential, and:

```
% grid-proxy-destroy
```

to destroy the Globus proxy credential. Or the user might simply let both credentials expire.

3. Using Authorization Service Interface

The user may access resources that are configured to contact a CAS server using the OGSA-AuthZ service interface. This interface allows the resource to pull down the user's rights from the CAS service and enforce the rights. Refer to [Section 7, "OGSA-AuthZ Service Interface"](#) for more details.

CAS User Commands

Name

cas-proxy-init -- Generate a CAS proxy

```
cas-proxy-init [common options] [ -p proxyfile | -t tag ]
```

Tool description

The **cas-proxy-init** command contacts a CAS server, obtains an assertion for the user, and embeds it in a credential. This credential can be used to access CAS-enabled services.

Options

Command-specific options

-b Generate a CAS credential that includes only those permissions specified in file *policyFileName* (the default *policyFile* is to generate a credential with all the user's permissions). Details about the template of the file is provided [here](#).

-u Choose a filename in which to store the CAS credential based on the value *tag*. Cannot be used with the **-p** option.

-w Specify the file in which to store the CAS credential. Cannot be used with the **-t** option.

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

- | | |
|---|--|
| -a, --anonymous | Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism. |
| -c, --serverCertificate <file> | Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism. |
| -debug | Runs the client with debug message traces and error stack traces. |
| -f, --descriptor <file> | Specifies a client security descriptor. Overrides all other security settings. |
| -help | Prints the usage message for the client. |
| -l, --contextLifetime <value> | Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism. |

<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

The following gets the assertion from the CAS server, generates a proxy with the assertion and writes it out to "casProxy".

```
cas-proxy-init -p casProxy
```

Requesting specific permissions from the CAS server

It is possible to request specific permissions from the CAS server using the `-f` option. This option causes **cas-proxy-init** to read a set of requested rights from a file.

This file should contain one or more resource identifiers:

Resource: *ResourceNamespace* | *ResourceName*

For each resource, there should be one or more action identifiers:

serviceType *action*

For example, if the client needed assertions for "file/read" service/action (permission) on two resources ("ftp://sample1.org" and "ftp://sample3.org", both in "FTPNamespace") but "directory/read" and "directory/write" permissions on the former resource only, the policy file should have the following entries:

Resource: FTPNamespace | ftp://sample1.org

file read

directory read

directory write

Resource: FTPNamespace | ftp://sample3.org

file read

To indicate any resource, the following wildcard notation should be used:

uri:samlResourceWildcard

To indicate any action, the following wildcard notation for *serviceType* and *action* should be used. Note that this should be the first (and clearly the only action) in the list of actions specified. All other actions in the list are ignored and if it is not the first, it is not treated as a wildcard.

uri:samlActionNSWildcard uri:samlActionWildcard

For example, if the client needs assertions for all resources and all actions, the policy file should look like:

Resource: uri:samlResourceWildcard

uri:samlActionNSWildcard uri:samlActionWildcard

If the client needs assertions for all actions on resource "FTPNamespace|ftp://sample1.org", the policy file should be as follows:

Resource: FTPNamespace | ftp://sample1.org

uri:samlActionNSWildcard uri:samlActionWildcard

Name

cas-wrap -- Runs program with CAS credentials

```
cas-wrap [common options] [ -p proxyfile | -t tag ]
```

Tool description

The **cas-wrap** command runs a grid-enabled program, causing it to use previously-generated CAS credentials.

This command invokes the given command with the given argument using the specified previously-generated CAS credential. For example:

```
casAdmin$ cas-wrap -t my-community gsincftp myhost.edu
```

will look for a credential generated by a previous execution of:

```
casAdmin$ cas-proxy-init -t my-community
```

and then set the environment to use that credential while running the command:

```
casAdmin$ gsincftp myhost.edu
```

The second form should be used if **cas-proxy-init** was run with the `-p` option. For example:

```
casAdmin$ cas-wrap -p /path/to/my/cas/credential gsincftp myhost.edu
```

will look for a credential generated by a previous execution of:

```
casAdmin$ cas-proxy-init -p /path/to/my/cas/credential
```

and then set the environment to use that credential while running the command:

```
casAdmin$ gsincftp myhost.edu
```

Options

Command-specific Options

`-p` Specify the file in which to store the CAS credential. Cannot be used with the `-t` option.

~~proxy~~
file

`-t` Choose a filename in which to store the CAS credential based on the value `tag`. Cannot be used with the `-p` option.

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

`-a, --anonymous` Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.

-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s cas-url	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z authorization	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <code><fqdn></code> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

Example of using cas-wrap to transfer a file.

```
cas-wrap -p casProxy globus-url-copy gsiftp://somehost.edu/some_file_path \  
file:///some_file_path
```

CAS Admin Commands

Name

cas-enroll -- Enroll a CAS Object

```
cas-enroll [common options] trustAnchor userGpName nickname authMethod authData cas-enroll [common options] namespace userGpName nickname basename comparisonAlg cas-enroll [common options] object userGpName objectName namespaceNick cas-enroll [common options] serviceType userGpName serviceTypeName
```

Tool description

This command line client is used to enroll a CAS Object, which includes trust anchors, namespaces, objects and service types.

Enrolling Trust Anchors

To enroll a trust anchor, the user must have `cas/enroll_trustAnchor` permission on that CAS server object (that is, the user must have permission to perform the `enroll_trustAnchor` action on the CAS service type).

The enroll operation allows the user to choose a user group to which `cas/grantAll` permission on the enrolled object should be granted. The nickname should be unique across the CAS database and is used to refer to this trust anchor.

To enroll trust anchors:

```
casAdmin$ cas-enroll [common options] trustAnchor userGpName nickname authMethod authData
```

where:

userGpName Indicates the user group to which `cas/grantAll` permission should be granted on this trust anchor entity.

nickname Indicates the trust anchor nickname.

authMethod Indicates the authentication method used by the trust anchor.

authData Indicates the data used for authentication, typically the DN.

Enrolling Namespaces

To enroll a namespace, the user must have `cas/enroll_namespace` permission (that is, the user must have permission to perform the `enroll_namespace` action on the cas service type).

The enroll operation allows the user to choose a userGroup to have `cas/grantAll` permission on the enrolled object. The comparison algorithm specified should be the name of the Comparison class that needs to be used to compare objects that belong to this namespace. The nickname should be unique across the CAS database and is used to refer to this user.

Also, two namespaces are added to the CAS database at boot up time, other than the inherent CAS Namespace:

- `FTPDirectoryTree` uses the `WildcardComparison` Algorithm and has the base URL set to the current directory.
- `FTPEXact` uses the `ExactComparison` Algorithm and has the base URL set to the current directory.

To enroll namespaces:

```
casAdmin$ cas-enroll [common options] namespace userGpName nickname basename comparisonAlg
```

where:

<i>userGpName</i>	Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.
<i>nickname</i>	Indicates the nickname of the namespace to be unenrolled. If the trust anchor nickname specified does not exist, an error is <i>not</i> thrown. If the unenroll operation is successful, all policy data on that trust anchor is purged.
<i>basename</i>	Indicates the base URL for the namespace.
<i>comparisonAlg</i>	Indicates the comparison algorithm to be used. Unless the standard comparison algorithms described below are used, the fully qualified name of the class that needs to be used should be given. The class needs to extend from the abstract class <code>org.globus.cas.impl.service.ObjectComparison</code> .

The two comparison classes provided as a part of the distribution are:

- `ExactComparison`: This class does a case-sensitive exact comparison of the object names. If `comparisonAlg` in the above method is set to `ExactComparison`, the class in the distribution is loaded and used.
- `WildcardComparison`: This class does wild card matching as described in [CAS Simple Policy Language](#)¹. It assumes that the wild card character is "*" and that the file separator is "/". If `comparisonAlg` in the above method is set to `WildcardComparison`, the class in the distribution is loaded and used.

Enrolling Objects

To enroll an object, the user must have cas/enroll_object permission (that is, the user must have permission to perform the enroll_object action on the cas service type).

The enroll operation allows the user to choose a userGroup to have cas/grantAll permission on the enrolled object. The name of the object and the namespace this object belongs to identify an object in the database and should be unique across the CAS database.

To enroll objects:

```
casAdmin$ cas-enroll [common options] object userGpName objectName namespaceNick
```

where:

<i>userGpName</i>	Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.
<i>objectName</i>	Indicates the name of the object.
<i>namespaceNick</i>	Indicates the nickname of the namespace to which this object belongs.

¹ http://www.globus.org/toolkit/docs/3.2/cas/CAS_policy_language_0.2.pdf

Enrolling Service Types

To enroll a service type, the user must have `cas/enroll_serviceType` permission (that is, the user must have permission to perform the `enroll_serviceType` action on the cas service type).

The enroll operation allows the user to choose a `userGroup` to have `cas/grantAll` permission on the enrolled service type. The service type name should be unique across the CAS database.

To enroll service types:

```
casAdmin$ cas-enroll [common options] serviceType userGpName serviceTypeName
```

where:

`userGpName` Indicates the user group to which `cas/grantAll` permission should be granted on this trust anchor entity.

`serviceTypeName` Indicates the service type name.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

<code>-a, --anonymous</code>	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"> • <code>msg</code> for GSI Secure Message, or • <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"> • <code>sig</code> for signature, or • <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here .

The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.

- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For detailed examples of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Name

cas-remove -- Remove a CAS object from the database

```
cas-remove [common options] trustAnchor nickname cas-remove [common options] namespace
nickname cas-remove [common options] object objName namespaceNick cas-remove [common
options] serviceType serviceTypeName
```

Tool description

Removing Trust Anchors

To remove a trust anchor, the user must have cas/remove permission on that trust anchor. The trust anchor must also be unused (that is, there may not be any users in the database that have this trust anchor or it may not be a part of any object group).

To remove trust anchors:

```
casAdmin$ cas-remove [options] trustAnchor nickname
```

where:

nickname Indicates the nickname of the trust anchor to be unenrolled.

If the trust anchor nickname specified does not exist, an error is *not* thrown. If the unenroll operation is successful, all policy data on that trust anchor is purged.

Removing Namespaces

To remove a namespace, the user must have cas/remove permission on that namespace. The namespace must also be unused — that is, there may not be any object in the database that belongs to this namespace.

```
casAdmin$ cas-remove [options] namespace nickname
```

where:

nickname Indicates the nickname of the namespace to be unenrolled.

If the namespace nickname specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that trust anchor is purged.

Removing Objects

To remove an object the user must have cas/remove permission on that object. The object must also be unused — that is, there may not be any object group in the database that this object belongs to.

```
casAdmin$ cas-remove [options] object objName namespaceNick
```

where:

objName Indicates the name of the object to be removed.

namespaceNick Indicates the nickname of the namespace to which this object belongs.

If the object specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that object is purged.

Removing Service Types

To remove a service type the user must have `cas/remove` permission on that service type. The service type must also be unused — that is, there may not be any service type to action mapping.

```
casAdmin$ cas-remove [options] serviceType serviceTypeName
```

where:

serviceTypeName Indicates the service type name.

If the service type specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that service type is purged.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

<code>-a, --anonymous</code>	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrf/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.

- v Prints the version number.
- x, --proxyFilename <value> Sets the proxy file to use as client credential.
- z *authorization* Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-action -- Maintains service types

```
cas-action [common_options] [ add | remove ] serviceTypeName actionName
```

Tool description

Use the **cas-action** command to add an action mapping to a service type or remove an action mapping from a service type.

To add an action mapping to a service type, the user must have `cas/create_group_entry` permission on the service type.

To remove a service type action mapping, the user must have `cas/delete_group_entry` permission on the service type.

If the group member being removed does not exist, an error is *not* thrown.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
-s <i>cas-url</i>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here .

The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.

- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For an example of using this command, see [Section 9, “Adding action mappings”](#).

Name

cas-group-admin -- Maintains user groups, object groups, or serviceAction groups

```
cas-group-admin [common options] [ user | object | serviceAction ] create userGpName groupName cas-  
group-admin [common options] [ user | object | serviceAction ] delete groupName
```

Tool description

Use **cas-group-admin** to create or delete user groups, object groups, or serviceAction groups. Note: to add or delete entries to these groups, see [\[fixme olink to other clients\]](#).

Adding user groups

To create a new user group the user must have cas/create_user_group permission (that is, the user must have permission to perform the create_user_group action on the cas service type). The user group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created user group. If the user group that is chosen to have cas/grantAll permission is the new group created, then the user making this request is added to the new group.

To add a user group:

```
casAdmin$ cas-group-admin [common options] user create userGpName groupName
```

where:

userGpName Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.

groupName Indicates the name of the user group being created.

Deleting user groups

To delete a user group, the user must have cas/delete_user_group entry permission on that user group. The group must be empty and also must not be referenced from other entities in the database (for example, it should not be a member of some object group).

If the user group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that user group is purged.

```
casAdmin$ cas-group-admin [common options] user delete groupName
```

where:

groupName Indicates the name of the user group to be deleted.

Creating An Object Group

To create a new object group, the user must have cas/create_object_group permission (that is, the user must have permission to perform the create_object_group action on the CAS service type). The object group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created object group.

```
casAdmin$ cas-group-admin [common options] object create userGpName groupName
```

where:

userGpName Indicates the user group to which cas/grantAll permission should be granted on this object group.

groupName Indicates the object group name.

Deleting An Object Group

To delete an object group, the user must have cas/delete_user_group entry permission on that object group. The group must be empty.

If the object group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that object group is purged.

```
casAdmin$ cas-group-admin [common options] object delete groupName
```

where:

groupName The name of the object group to be deleted.

Creating A Service/Action Group

To create a new service/action group, the user must have cas/create_serviceAction_group permission (that is, the user must have permission to perform the create_serviceAction_group action on the CAS service type). The serviceAction group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created serviceAction group.

```
casAdmin$ cas-group-admin [common options] serviceAction create userGpName groupName
```

where:

userGp-Name Indicates the user group to which cas/grantAll permission should be granted on this service/action group.

groupName Indicates the name of the service/action group being created.

Deleting A Service/Action Group

To delete a service/action group, the user must have cas/delete_user_group entry permission on that service/action group. The group must be empty and also must not be referenced from any other entity in the database. For example, it should not be a member of some object group.

If the service/action group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that service/action group is purged.

```
casAdmin$ cas-group-admin [common options] serviceAction delete groupName
```

where:

~~gp~~ Indicates the name of the service/action group to be deleted.

~~name~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s cas-url	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrf/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z authorization	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For examples of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Name

cas-group-add-entry -- Adds CAS objects to CAS groups

```
cas-group-add-entry [common options] user groupName nickname cas-group-add-entry  
[common options] object groupName objectSpecDesc objectSpec cas-group-add-entry [common  
options] serviceAction groupName serviceTypeName actionName
```

Tool description

Use **cas-group-add-entry** to add users to a user group, objects to an object group, or service/actions to service/action groups. Note: to add or delete groups, see [fixme olink to other clients].

Adding Member To A User Group

To add a user to a user group, the user must have cas/add_group_entry permission on that particular user group. Only user nicknames that exist in the CAS database can be valid members.

```
casAdmin$ cas-group-add-entry [common options] user groupName nickname
```

where:

~~gp~~ Indicates the user group name to which the member needs to be added.

~~nk~~

~~nk~~ Indicates the nickname of the user to be added to this group.

~~nk~~

Adding Member To An Object Group

To add a member (an object group can have the following CasObjects as members: object, user, user group, service type, namespace or trust anchor) to an object group, the user must have cas/add_group_entry permission on that particular object group.

```
casAdmin$ cas-group-add-entry [common options] object groupName objectSpecDesc objectSpec
```

where:

~~gp~~ Indicates the object group name to which the member needs to be added.

~~nk~~

~~ob~~ Indicates the type of CasObject. Can be one of the following options:

~~ob~~

~~ob~~ • trustAnchor

~~ob~~

~~ob~~ • user

• userGroup

• object

• namespace

• serviceType

~~o~~ Indicates the identifier for the CasObject the user is adding. Can be one of the following:

- ~~o~~ • nickname if adding a trustAnchor or user
- groupName if adding a userGroup
- objectNamespace objectName if adding an object
- nickname if adding a namespace
- serviceTypeName if adding a serviceType

Adding Service/Action To A Service/Action Group

To add a service/action to a serviceAction group, the user must have cas/add_group_entry permission on that particular serviceAction group (that is, the user must have permission to perform add_group_entry action on that service action group).

```
casAdmin$ cas-group-add-entry [common options] serviceAction groupName serviceTypeName act
```

where:

~~o~~ Indicates the service/action group to which the service/action needs to be added.

~~o~~

~~o~~ Indicates the service type name part of the mapping to be added to the group.

~~o~~

~~o~~

~~o~~

~~o~~ Indicates the action name part of the mapping to be added to the group.

~~o~~

~~o~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be:

	<ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For examples of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Name

cas-group-remove-entry -- Removing CAS objects from CAS groups

```
cas-group-remove-entry [common options] user groupName nickname cas-group-remove-  
entry [common options] object groupName objectSpec objectSpecDesc cas-group-remove-  
entry [common options] serviceAction groupName serviceTypeName actionName
```

Tool description

Use **cas-group-remove-entry** to remove users from a user group, objects from an object group, or service/actions from a service/action group. Note: to add or delete groups, see [\[fixme olink to other clients\]](#).

Removing User From A User Group

To remove a user from a user group, the user must have `cas/remove_group_entry` permission on that particular user group.

If the group member being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] user groupName nickname
```

where:

~~gp~~ Indicates the user group name from which the member needs to be removed.

~~nk~~

~~nk~~ Indicates the nickname of the user to be removed from this group.

~~nk~~

Removing Member From An Object Group

To remove an object from an object group the user must have `cas/remove_group_entry` permission on that particular object group:

If the group member being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] object groupName objectSpec objectSpecDesc
```

where:

~~gp~~ Indicates the object group name from which the member needs to be removed.

~~nk~~

~~ob~~ Indicates the type of CasObject. Can be one of the following options:

~~ob~~

~~ob~~ • trustAnchor

~~ob~~

~~ob~~ • user

~~ob~~

~~ob~~ • userGroup

~~ob~~

~~ob~~ • object

~~ob~~

~~ob~~ • namespace

- `serviceType`

~~o~~ Indicates the identifier for the CasObject the user is adding. Can be one of the following:

~~o~~
~~o~~

- `nickname` if adding a trustAnchor or user
- `groupName` if adding a userGroup
- `objectNamespace objectName` if adding an object
- `nickname` if adding a namespace
- `serviceTypeName` if adding a serviceType

Removing A Service/Action From A Service/Action Group

To remove a service/action from a service/action group, the user must have `cas/remove_group_entry` permission on that particular service/action group.

If the action being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] serviceAction groupName serviceTypeName
```

where:

~~o~~ Indicates the serviceAction group name from which the service/action needs to be removed.
~~o~~

~~o~~ Indicates the service type name part of the mapping to be removed from the group.
~~o~~
~~o~~
~~o~~

~~o~~ Indicates the action name part of the mapping to be removed from the group.
~~o~~
~~o~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

<code>-a, --anonymous</code>	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.

<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <code><fqdn></code> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-rights-admin -- Granting or revoking permissions

```
cas-rights-admin [common options] [ grant | revoke ] userGroupName objectSpecDesc objectSpec  
actionSpecDesc actionSpec
```

Tool description

Use **cas-rights-admin** to grant or revoke rights.

Granting Permissions To A User Group On An Object/Object Group

The user may grant permissions to a user group on an object or object group to perform a service action or service action group (that is, to perform any action that is a member of the service action group to which permission is granted), provided the user has both:

- cas/grant permission on the object or object group, and
- permission to perform the service action or service action group on the object or object group.

```
casAdmin$ cas-rights-admin [common options] grant userGroupName objectSpecDesc objectSpec
```

where:

~~⌘~~ Indicates the user group to be granted permission.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the identifier for the object or object group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type:

- ~~⌘~~
~~⌘~~ • object
~~⌘~~ • objectGroup

~~⌘~~ Indicates the identifier for action or action group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type:

- ~~⌘~~
~~⌘~~ • serviceAction
~~⌘~~ • serviceActionGp

Revoking A Policy In The CAS Database

The user may revoke a policy in the CAS database if the user has cas/revoke permission on the object or object group on which the policy is defined.

```
casAdmin$ cas-rights-admin [common options] revoke userGroupName objectSpecDesc objectSpec
```

where:

~~⌘~~ Indicates the user group for which you want to grant permission.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type of CasObject. Can be one of the following:

~~⌘~~
~~⌘~~
~~⌘~~

- trustAnchor
- user
- userGroup
- object
- namespace
- serviceType
- userGroup

~~⌘~~ Indicates the identifier for the object or object group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the identifier for the action or action group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type (serviceAction or serviceActionGp).

~~⌘~~
~~⌘~~
~~⌘~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.

- `-m, --securityMech <type>` Specifies the authentication mechanism. The value *type* can be:
- `msg` for GSI Secure Message, or
 - `conv` for GSI Secure Conversation.
- `-p, --protection <type>` Specifies the protection level. *type* can be:
- `sig` for signature, or
 - `enc` for encryption.
- `-s cas-url` Sets the CAS Service instance, where *cas-url* is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown [here](#).
- The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.
- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where *<fqdn>* is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For an example of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Chapter 2. Troubleshooting

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

1. Credential Troubleshooting

1.1. Credential Errors

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

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

Table 2.1. Credential Errors

Error Code	Definition	Possible Solutions
Your proxy credential may have expired	Your proxy credential may have expired.	Use <code>grid-proxy-info</code> to check whether the proxy credential has actually expired. If it has, generate a new proxy with <code>grid-proxy-init</code> .
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.	Check the system clocks on the local and remote system.
Your end-user certificate may have expired	Your end-user certificate may have expired	Use <code>grid-cert-info</code> 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 <code>grid-proxy-destroy</code>) and creating a new one (with <code>grid-proxy-init</code>). Important: 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), <code>grid-proxy-init</code> will refuse to create a proxy certificate.	You can "fix" this by changing the permissions on the private key file. Important: However, you will still have a much more serious problem: it is 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	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 Installing GT 4.2.1 for details.
You may not trust the remote system's CA	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 Installing GT 4.2.1 for details.
There may be something wrong with the remote service's credentials	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 cannot find anything wrong with your credentials, check for the same conditions on the remote system (or ask a remote administrator to do so) .

1.2. Some tools to validate certificate setup

1.2.1. grid-cert-diagnostics

The `grid-cert-diagnostics` program checks prints diagnostics about the user's certificates, and host security environment.

```
% grid-cert-diagnostics -p
```

1.2.2. Check that the user certificate is valid

```
openssl verify -CApath /etc/grid-security/certificates
-purpose sslclient ~/.globus/usercert.pem
```

1.2.3. Connect to the server using s_client

```
openssl s_client -ssl3 -cert ~/.globus/usercert.pem -key
~/.globus/userkey.pem -CApath /etc/grid-security/certificates
-connect <host:port>
```

Here `<host:port>` denotes the server and port you connect to.

If it prints an error and puts you back at the command prompt, then it typically means that the *server* has closed the connection, i.e. that the server was not happy with the client's certificate and verification. Check the SSL log on the server.

If the command "hangs" then it has actually opened a telnet style (but secure) socket, and you can "talk" to the server.

You should be able to scroll up and see the subject names of the server's verification chain:

```
depth=2 /DC=net/DC=ES/O=ESnet/OU=Certificate Authorities/CN=ESnet Root CA 1
verify return:1
depth=1 /DC=org/DC=DOEGrids/OU=Certificate Authorities/CN=DOEGrids CA 1
verify return:1
depth=0 /DC=org/DC=doegrids/OU=Services/CN=wiggum.mcs.anl.gov
verify return:1
```

In this case, there were no errors. Errors would give you an extra line next to the subject name of the certificate that caused the error.


1.2.4. Check that the server certificate is valid

Requires root login on server:

```
openssl verify -CApath /etc/grid-security/certificates -purpose sslserver
/etc/grid-security/hostcert.pem
```

2. Error Messages

Table 2.2. WS A&A Authorization Framework Error Messages

Error Code	Definition
<p>org.globus.cas.impl.databaseAccess.CasDBException, connection refused</p>	<p>If the CAS service fails with following error:</p> <pre> faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.userException faultSubcode: faultString: org.apache.commons.dbcp.DbcpException: Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TC you need to troubleshoot the connection to the CAS database.</pre>
<p>CAS clients fail with org.globus.cas.impl.databaseAccess.CasDBException</p>	<p>If CAS clients fail with database permission exceptions similar to:</p> <pre> [Caused by: ERROR: permission denied for relation service_type_action]; nested exception is:org.globus.cas.impl.databaseAccess.CasDBException:</pre> <p>, then there is something wrong with user permissions on the database.</p> <p> Note</p> <p>This is a specific instance of an error for the relation <i>service_type_action</i>. This error could be raised on any rel</p>

Chapter 3. Information Regarding CAS Licensing

This version of CAS uses the OASIS Security Assertion Markup Language (SAML) standard. Users should be aware that RSA Security has identified four patents it believes could be relevant to implementing certain operational modes of the SAML specifications. Previously, RSA required that users would execute a royalty-free reciprocal license to the RSA patents, and the Globus Alliance had established a license agreement with RSA covering usage of SAML in the Globus Toolkit.

On May 11, 2006, however, RSA issued the following statement:

RSA Intellectual Property Rights Statement

In previous correspondence dated December 6, 2004, January 20, 2003 and April 22, 2002, RSA Security Inc. ("RSA") disclosed that it is the assignee of U.S. Patent Nos. 6,085,320 and 6,189,098, both entitled "Client/Server Protocol for Proving Authenticity" and U.S. Patent Nos. 5,922,074 and 6,249,873, both entitled "Method of and Apparatus for Providing Secure Distributed Directory Services and Public Key Infrastructure" (collectively, the "RSA Patents"). At that time, RSA believed that these four patents could be relevant to practicing certain operational modes of the OASIS Security Assertion Markup Language ("SAML") specifications. In the correspondence, RSA offered to grant non-exclusive, royalty-free licenses on a non-discriminatory basis for the RSA Patents.

In the interest of encouraging deployment of SAML-based technologies, RSA hereby covenants, free of any royalty, that it will not assert any claims in the RSA Patents which may be essential to the SAML standard v1.0, 1.1 and 2.0 (hereinafter "NECESSARY CLAIMS") against any other entity with respect to any implementation conforming to the SAML standard v1.0, 1.1 and/or 2.0. This covenant shall become null and void with respect to any entity that asserts, either directly or indirectly (e.g., through an affiliate), any patent claims or threatens or initiates any patent infringement suit against RSA and/or its subsidiaries or affiliates. The revocation of the covenant shall extend to all prior use by the entity asserting the claim.

RSA will continue to honor existing license agreements for the RSA Patents and will continue to offer as an option to interested third parties the same licensing arrangement described in our previous correspondence. (The license agreement, along with instructions for obtaining and completing the license, are available on RSA's website www.rsasecurity.com)

Please refer to [published statement](#)¹

¹ <http://xml.coverpages.org/RSA-SAML-NonAssert.html#statement>

 **Important**

Users should always check with their legal counsel about the interpretation of the statement, but the statement may indicate that for most SAML-users no execution of any license will be required.

Glossary

some terms not in the docs but wanted in glossary: scheduler

C

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.

P

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

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

GT 4.2.1 CAS Developer's Guide

GT 4.2.1 CAS Developer's Guide

Introduction

This document is an overview of the CAS server and its functionality. It is intended to help CAS developers understand the features of the current implementation and throws light on permissions, assertion generation and querying capability in CAS. This document also contains information that is useful when writing clients for the CAS server.

The [Globus Toolkit 4.2.1 Developer's Guide](#) provides general information such as best practices and coding guidelines used.

Table of Contents

1. Before you begin	1
1. Feature summary	1
2. Tested platforms	1
3. Backward compatibility summary	1
4. Technology dependencies	1
5. Security Considerations for CAS	2
2. Usage scenarios	3
3. Setting up CAS for GridFTP	4
1. Overview	4
2. Files and directories in CAS	4
3. Configuring the CAS server to work with the GridFTP server	5
4. Transferring files using CAS and GridFTP	8
5. Managing policy	8
6. Authorization Enforcement at the GridFTP Server	9
4. Tutorials	10
5. Architecture and design overview	11
1. CAS Database Overview	11
2. CAS Permissions	15
3. CAS Requests	16
4. CAS Administrative Requests	16
5. CAS Query Requests	16
6. Assertion requests	16
7. OGSA-AuthZ Service Interface	17
6. APIs	19
1. Programming Model Overview	19
2. Component API	19
7. Services and WSDL	20
1. Protocol overview	20
2. Operations	20
3. CAS Resource Properties	21
4. Faults	21
5. WSDL and Schema Definition	21
I. CAS Admin Commands	22
cas-proxy-init	23
cas-wrap	26
cas-enroll	29
cas-remove	33
cas-action	36
cas-group-admin	38
cas-group-add-entry	42
cas-group-remove-entry	45
cas-rights-admin	48
II. CAS Query Commands	?
cas-whoami	52
cas-list-object	54
cas-get-object	56
cas-group-list-entries	58
cas-find-policies	60
query-cas-service	62
8. Semantics and syntax of domain-specific interface	64
9. Configuring	66

1. Configuration overview	66
2. Loading the CAS service at start up	66
3. Configuring the VO Description	67
4. Configuring the maximum assertion lifetime	67
5. Configuring database backend	67
6. Configuring security descriptor	67
7. Configuring with a GridFTP Server	68
8. Configuring CAS to manage policy for web service.	68
9. CAS auto-registration with default WS MDS Index Service	68
10. Registering CAS manually with default WS MDS Index Service	70
10. Environment variable interface	71
1. Environmental variables for CAS	71
11. Debugging	72
1. Development Logging in Java WS Core	72
2. Enabling verbose logging	72
3. Debugging info from CAS clients	73
12. Troubleshooting	74
1. Command run using cas-wrap does not pick up CAS proxy	74
2. Error Messages	75
13. Related Documentation	76
Glossary	77

List of Tables

5.1. User tables	12
5.2. Action tables	12
5.3. Resource Tables	14
5.4. Policy Statement Table	15
5.5. Request methods	17
9.1. Database parameters	67
9.2. Mapping from web services object to CAS object	68
12.1. WS A&A Authorization Framework Error Messages	75

Chapter 1. Before you begin

1. Feature summary

Features new in GT 4.2.1:

- Support for OGSA-AuthZ Authorization Service interface
- Support for managing web services policy.

Other Supported Features

- File-level access control for GridFTP
- Issuance of SAML authorization decisions

Deprecated Features

- None

2. Tested platforms

Tested Platforms for CAS

- Windows XP
- Linux (Red Hat 7.3)

Tested Containers for CAS

- Java WS Core container
- Tomcat 5.0.30

3. Backward compatibility summary

CAS has been updated to use the latest version of Java WS Core, which now supports the final version of WSRF/WSN specification.¹ This service is not compatible with the previous stable versions, GT 4.0.x

Fixed SAML assertions embedded in proxy to comply with RFC 3820 requirements. CAS assertions generated by default in GT 4.0.x will not be consumed by GT 4.2.x services that use assertions.

4. Technology dependencies

The CAS service depends on the following GT components:

- WS Authentication and Authorization
- Java WS Core

¹ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/rn/index.html#id2534919>

The CAS GridFTP authorization module depends on the following GT components:

- Non-WS Authentication and Authorization

The CAS service depends on the following 3rd party software:

- OpenSAML

The CAS GridFTP authorization module depends on the following 3rd party software:

- libxml

5. Security Considerations for CAS

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

Chapter 2. Usage scenarios

Typically in a VO a single CAS service is run and several clients contact the service to retrieve assertions about operations on resources. These assertions can be then presented to the resources to gain access. Users with some administrative rights can grant/revoke access to these resources.

GT ships with clients for users to retrieve and embed assertions in proxy and another to use that proxy to run some executable. Listed below are some helper methods in the client package that may be used for writing other clients for the CAS service. Refer to code and/or Java Docs for more information.

For more information, see the domain section [here](#).

Chapter 3. How to Set Up CAS to Use with GridFTP

This document is intended to be used with the [CAS](#) and [GridFTP](#) documents, to configure and use CAS with GridFTP.

1. Overview

CAS is used to administer access rights to files and directories and GridFTP server can be configured to enforce those rights. The following is an overview of the process:

1. A CAS administrator sets up rights on the CAS server.
2. A user retrieves those rights in the form of a signed assertion and presents it to the GridFTP server at the time of access.
3. If the assertion presented has the necessary permissions and the GridFTP server trusts the CAS server that issued it, access is allowed.

In a default Globus Toolkit installation, CAS is installed with a set of credentials associated with the CAS server. A CAS administrator has all rights on the CAS server and manages users. To be used with GridFTP, the administrator needs to add users, add the files and directories he wants protected and grant specific rights to the files.

The user then retrieves a CAS assertion from the server, for either specific files/directories or all his access rights stored in CAS. This is done using the client side command, [cas-proxy-init](#), which embeds the assertion in the user's proxy. Another client side command, [cas-wrap](#) can then be used to run the GridFTP command, [globus-url-copy](#), to ensure that the proxy is used.

Details about the CA who issues these credentials needs to be added to the CAS server as a trust anchor. Each user has a nick name, a subject DN and is associated with a trust anchor. A particular user is designated as superuser, so has permissions to add users and manager rights. This can be set up using the [bootstrap](#) step.

Users who are not administrators can be added using [admin command line tools](#). Permissions in a CAS database are granted to user groups, rather than users. So a user group needs to be created and users added to it prior to assigning permissions.

2. Files and directories in CAS

Files and directories are "objects" in CAS. Each object has a name and a namespace associated with it. The namespace determines the scope of the object and also the comparison method to use with object names. A predefined namespace, called FTPDirectoryTree is loaded in the CAS database upon bootstrap and this is the namespace expected for GridFTP objects.

For example, if permission on file "some_file_path" on host "somehost.edu" needs to be managed, it needs to be added as an object, with name "ftp://somehost.edu/some_file_path" and namespace "FTPDirectoryTree". This object "ftp://somehost.edu/some_file_path" will be recognized by the CAS-enabled GridFTP server at "somehost.edu" as referring to the file named "some_file_path". An object in the FTPDirectoryTree namespace with the name "ftp://somehost.edu/some_directory_path/*" will be recognized by the CAS-enabled GridFTP server at "somehost.edu" as referring to all files and directories under "some_directory_path".

In some cases, it may be desirable to have a GridFTP server recognize CAS assertions that use a hostname other than the server's fully qualified domain name. Starting the GridFTP server with the option "-H otherhost" will cause the GridFTP server to recognize objects with names that start with "ftp://otherhost/" instead.

The actions that are allowed on a file and a directory are listed in the following table:

Action	Result for a file	Result for a directory
read	Gives permission to read the file.	Gives permission to chdir to the directory.
lookup	Gives the right to get Unix stat() information.	Gives the right to chdir to and to list the contents of the directory.
write	Allows modification of an existing file.	Gives the right to chdir to the directory.
create	Allows creation of the file if it does not exist.	Allows creation of the directory if it does not exist and gives the right to chdir to the directory if it does exist.
delete	Allows deletion of the file.	Allows deletion of the directory, if empty; also gives the right to chdir to the directory.
chdir	Does not apply.	Allows making the directory the current default directory.

These actions are predefined and loaded onto CAS database when bootstrap is done. Apart from these actions on files/directories, there is another action "authz_assert" which is used to override the existing assertions (either from proxy or from a previous "authz_assert" command) with the assertions received over the GridFTP control channel. FTP clients can use the command "SITE AUTHZ_ASSERT" to send assertions to the GridFTP server.

The following is a summary of supported FTP commands and the permissions they require:

Typical Client Command	FTP Protocol Command	Rights Required
get	RETR	read
put	STOR	write, if file exists; create, if file does not exist
delete	DELE	delete
ls	LIST	lookup
chdir	CWD	any of: chdir, lookup, read, write, create, or delete
mkdir	MKD	create
rmdir	RMD	delete
rename	RNFR / RNT0	read and delete on old file; write on new file, if it exists, create on new file, if it does not exist

3. Configuring the CAS server to work with the GridFTP server

3.1. Step 1: Set up the CAS server.

This must be done by user "casAdmin".

3.2. Step 2: Run the CAS server with host credentials

Run the CAS server with host credentials `/O=Grid/OU=test/CN=foo.bar.edu`.



Note

The CA that issues these credentials should be trusted by the GridFTP server. Setting up of trusted CA is described [here](#)

3.3. Step 3: Enable CAS support in the GridFTP server

The GridFTP Server reads two files (`gsi-authz.conf` and `gsi-gaa.conf`) to determine how to perform certain authorization and mapping functions. If these files are not present (as is the case after a standard Globus Toolkit installation), the GridFTP server will not support CAS authorization (that is, the GridFTP server will ignore the CAS policy assertions in the user's credential and determine the user's permissions based solely on the user's identity).

Run **[`setup-globus-gaa-authz-callout`]** to create `gsi-authz.conf` and `gsi-gaa.conf` files that will cause the GridFTP server to honor CAS policy assertions. There are two ways to run this command:

1. To create the config files in the directory `/etc/grid-security` (where the GridFTP server looks for them by default), run the following as root:

```
$GLOBUS_LOCATION/setup/globus/setup-globus-gaa-authz-callout
```

2. To create the config files to another directory, run:

```
$GLOBUS_LOCATION/setup/globus/setup-globus-gaa-authz-callout -d mydir
```

Where 'mydir' is the path to the desired directory. You then must make sure the GridFTP server finds these files by setting the following environment variable *before* starting the GridFTP server:

- Set `GSI_AUTHZ_CONF` to `mydir/gsi-authz.conf`.
- Set `GSI_GAA_CONF` to `mydir/gsi-gaa.conf`.

By default, `setup-globus-gaa-authz-callout` will not overwrite an existing configuration file. Use the following options to overwrite existing GridFTP config files:

- Use the `-force` option to overwrite an existing `gsi-authz.conf` file
- Use the `-overwrite_gaa_config` option to overwrite an existing `gsi-gaa.conf` file.

3.4. Step 4: Configure the GridFTP server to trust the CAS server

The previous step configured the GridFTP server to understand CAS credentials. However, the GridFTP server will not allow a user authenticating with a CAS credential to perform any action that it would not allow the CAS server itself to perform. To configure the GridFTP server to trust a particular CAS server:

1. Create a local user account corresponding to the CAS server.
2. Use file permissions to allow that user account to have the desired level of file access.

3. Create a gridmap entry that maps the CAS server's distinguished name to that local account.

3.5. Step 5: Add a CAS administrator

Add a CAS admin, we'll use the CAS nickname, "casAdmin", and DN "/O=Grid/OU=test/CN=CAS Administrator". This credential is issued by a CA with certificate "/O=Grid/OU=test/CN=CA"

3.5.1. Using bootstrap to add a CAS administrator

This can be setup during [bootstrap](#) and below is the bootstrap file for the above scenario.

```
ta-name=defaultTrustAnchor
ta-authMethod=X509
ta-authData=/O=Grid/OU=test/CN=CA
user-name=superUser
user-subject=/O=Grid/OU=test/CN=CAS Administrator
userGroupname=superUserGroup
```

3.6. Step 6: Add a CAS user group

Since permissions are only on user groups, we need to add a user group, for example "readGroup". This can be done using [cas-group-admin](#).

```
cas-group-admin user create superUserGroup readGroup
```



Note

The superUserGroup here determined the user group that has permissions to manipulate the group that is being created i.e readGroup.

3.7. Step 7: Add a CAS user

Add the user who needs access to files. For example, add User1, with DN "/O=Grid/OU=test/CN=User 1" issued by the CA in [Step 4], using [cas-enroll]:

```
cas-enroll user superUserGroup User1 "/O=Grid/OU=test/CN=User 1" defaultTrustAnchor
```



Note

The superUserGroup here determined the user group that has permissions to manipulate the user that is being created i.e User1.

3.8. Step 8: Add user to the user group

To add the CAS user to the CAS user group, use [cas-group-add-entry](#):

```
cas-group-add-entry user readGroup User1
```

3.9. Step 9: Add CAS object

Add the file (on which you want to set permissions) as a CAS object with the name "ftp://foo.bar.edu/tmp/file1" and namespace "FTPDiretoryTree":

```
cas-enroll object superUserGroup "ftp://foo.bar.edu/tmp/file1" "FTPDirectoryTree"
```

 **Note**

The `superUserGroup` here determined the user group that has permissions to manipulate the object that is being created.

3.9.1. Giving permissions to all files in a directory

To give permissions to all files in a directory, create the CAS object with a wild card `*`. For example, object name `"ftp://foo.bar.edu/tmp/admin/*"`. If permission is given on this object, all files in that directory are affected.

For example, if you create a new user group called `"adminGroup"`, you give read permission to all users in that group as follows, any user in `"adminGroup"` can read any file in `"/tmp/admin"` on that server.

```
cas-rights-admin grant adminGroup object FTPDirectoryTree
    "ftp://foo.bar.edu/tmp/admin/*" serviceAction file read
```

3.10. Step 10: Add permission for users

Add permission for users in `"readGroup"` to be able to read object added in the previous step.

```
cas-rights-admin grant readGroup object FTPDirectoryTree
    "ftp://foo.bar.edu/tmp/file1" serviceAction file read
```

Now any user added to `readGroup` can read the file (that was added in Step 9).

4. Transferring files using CAS and GridFTP

Once the CAS server has been configured (as above), if `User1` wants to transfer the file `"/tmp/file1"` on `"ftp://foo.bar.edu"`, `User1` performs the following two steps:

1. Runs **cas-proxy-init** to get the CAS assertion:

```
cas-proxy-init -p casProxy
```

 **Note**

This gets the assertion from the CAS server, generates a proxy with the assertion and writes it out to `"casProxy"`.

2. Runs **cas-wrap** to transfer the file:

```
cas-wrap -p casProxy globus-url-copy gsiftp://somehost.edu/some_file_path
    file:///some_file_path
```

5. Managing policy

The CAS administrator can create groups of users and groups of objects to ease policy management.

Also, if the administrator wants to provide a set of rights, such as read and lookup, the administrator can create a service action group with these action as members. User groups can then be provided with access to all actions in the group.

Similarly the administrator can create groups of objects, which would be a group of GridFTP server and files on it, and grant access rights to users.

6. Authorization Enforcement at the GridFTP Server

The following describes how the GridFTP server processes a file transfer using CAS:

1. During the authentication phase, GridFTP server extracts the CAS assertion in the proxy credentials and stores it.
2. If the GridFTP server receives a CAS assertion over the control channel, it overwrites the old assertion (if any) with the new one.
3. Upon receiving a file transfer request, the GridFTP server looks for a CAS assertion.
4. If the assertion is present, checks whether the file being accessed has permission. If assertion has permission for the file, the access is allowed. Otherwise an error is thrown.
5. If no CAS assertion is found, the presence of user DN is checked in the gridmap file. If DN exists, then access is allowed. Otherwise an error is thrown.

Chapter 4. Tutorials

There are no tutorials available at this time.

Chapter 5. Architecture and design overview

The server essentially has users, actions, objects and policies governing the user's access to the objects for the purpose of performing specific actions. To better serve the requirements of a VO, the server allows grouping of users, actions and objects. This also facilitates specifying policies about them. The CAS server can be thought of as the front-end to a database that maintains state about such community permissions. The effect of each CAS request is either to modify this state or query it.

The server has two additional characteristics:

- Some query results are signed. Such signed results can be used for authorization at resources and other policy enforcement points that acknowledge such credentials.
- The same database is used to maintain information to control authorization decision for the CAS server.

The following topics contain related information:

- [CAS Database Overview](#)
- [CAS Permissions](#)
- [CAS Administrative Requests](#)
- [CAS Query Requests](#)

1. CAS Database Overview

The CAS database contains a number of tables to store information about users, resources (objects), actions and policies. This section describes each of those tables and their contents. The tables are categorized into tables used to identify and organize users in the database (`trust_anchor`, `user` and `user_group`), tables used to describe actions (`service_type`, `service_type_action`, `service_type_action_groups`), tables used to describe and organize resources or objects (`object`, `object_namespace` and `object_group`) and tables used to describe policies (`policy_statement`).

1.1. Tables relating to users

There are two categories of people in the CAS database, trust anchors and the users. The users may further be placed in user groups and the granularity of operations on the CAS database with respect to users is a user group.

Table 5.1. User tables

trust_anchor table	The trust_anchor table describes authorities that can generate credentials. It consists of tuples of {trust_anchor_nickname, authentication_method, authentication_data}. In general, the meaning of one of these tuples is that "database entries that refer to trust_anchor_nickname apply to the authority represented by authentication_data for authentication method authentication_method". The current implementation supports the following values in these fields: an authentication_method value of "x509" and an authentication_data value that is the certification authority's certificate. A trust_anchor_nickname value uniquely identifies an authentication_method and authentication_data across the database. For example, {globus_ca, x509, <contents of the globus CA cert>} associates the name globus_ca with the Globus <i>CA Certificate</i> .
user table	The user table consists of {user_nickname, trust_anchor_nickname, subject_name} tuples, which map raw authentication information into the symbolic names that appear in CAS user specifications. The intent of this tuple is to associate an internal name with a user that authenticates to the CAS service. The current implementation supports a subject_name which is the X509 distinguished name of a user. A subjectDN and trust anchor nickname uniquely identify an entry in the user_table. There is a one to one mapping between a user nickname and the combination of subjectDN and trust anchor nickname. For example, the tuple {user1, globus_ca, "/O=Globus/CN=User1 Name") indicates that the person who can authenticate as "/O=Globus/CN=User1 Name" using the authentication method of the trust anchor globus_ca has the permissions assigned to the user "user1" within the CAS database.
user_group table	This table maintains a list of all user groups in the CAS database.
user_group_entry table	The user_group_entry table consists of {group, user} tuples indicating that the specified user is a member of the group.

1.2. Tables to relating to actions

The database includes tables related to action specifications. Different services may define actions that have similar (or identical) names with different meanings and hence an action specification must include a service type in addition to the name of the action. The resource servers that receive CAS policy statements interpret the service types and actions. For example, a GridFTP server may honor policy statements that refer to the "file" service type and ignore policy statements for all other service types. In theory (because all this processing is done by the resource servers and not the CAS server), there's no need for the CAS server to keep track of allowable service types and actions. However, it is done to make it easier for administrators to detect and avoid errors while setting permissions.

The server also supports grouping of these service action mappings. Permissions may be granted to the service action groups or to a single service action.

Table 5.2. Action tables

service_type table	This table lists all the service types in the database.
service_type_action table	The service_type_action table consists of {service_type, action} tuples indicating that the specified action is valid for the specified service type. For example, a {service_type, action} of {file, read} means that read is a valid action for the service type file. This mapping is represented as "serviceType/action" in the current implementation.
service_action_group	This table lists all the service action group names.
service_action_group_entry	This table contains the following tuple (group, serviceType/action). The tuple indicates that the serviceType/action belongs to group.

1.3. Tables related to resources/objects

An object specification refers to an object or group of objects. An "object" may itself refer to either a single physical object (e.g. a file) or a collection of objects (e.g. all files within a directory). A given object is relevant within a defined namespace and the properties of the namespace apply to the object.

Table 5.3. Resource Tables

namespace_table	This table stores the following tuple (nickname, basename, comparisonAlg) indicating that objects in the namespace referred to by nickname in the CAS database are compared using the comparisonAlg and have a base URL of basename. A namespace uniquely identifies a single physical resource. For example, a namespace (ftpNS1, ftp://sample1.org/, wildcard) indicates that all object names within the realm of this namespace are to be compared using wildcard matching. Each comparison algorithm corresponds to a class in the CAS server code that implements an interface which defines routines for matching objects. The current implementation supports exact match and wildcard matching. Objects are represented as "objectNamespace objectName" in the current implementation.
object table	Stores the object name and the namespace that this object is in. For example, (/mydir/*, ftpNS1) implies that this object is within the ftpNs1 namespace, described above. Since this namespace has wild card matching /mydir/foo would match this object. CAS Objects can be either implicit (that is, those that are inherent to CAS) or explicit (that is, objects on other resources about which policies may be stored in the CAS database). While implicit objects may be of many types, external objects are always represented as type "object".
implicit objects	<p>It is sometimes convenient to treat some of the entities defined within the CAS server (such as users and groups) as objects. These implicit objects can be added to object groups and can appear in policy statements. Such policy statements govern access permissions to the CAS database. The types of implicit objects are:</p> <ul style="list-style-type: none"> • A user (a reference to an entry in the user table). This is used when granting permissions such as "may unenroll this user". • A user_group (a reference to an entry in the user_group table). This is used to grant permissions such as "may add users to this group". • A service_type (a reference to an entry in the service_type table). This is used when granting permissions such as "may add actions to this service type"; • An object_group (a reference to an entry in the object_group table). This is used when granting permissions such as "may add objects to this object group". • A namespace (reference to an entry in namespace table). This is used when granting permission like "may unenroll this namespace". • A trust anchor (reference to an entry in the trust_anchor_table). This is used when granting permissions like "may grant rights on this trust anchor". • The CAS server itself. This is used when granting permissions such as "may add new users to the CAS server". This is added at start up to the object_table.
object_group table	This table lists the names of object groups in the database.
object_group_entry table	The object_group_entry table consists of {object_group, objectSpecification, objectSpecDesc} tuples; this tuple indicates that the specified objectSpecification of the type objectSpecDesc is a member of the specified object group. The objectSpecification is an identifier for a object of type objectSpecDesc, i.e. an object, object group, user, user group, service type, namespace or trust anchor.

1.4. Tables relating to policy statements

The CAS server keeps track of policy statements, which are composed of three parts: a user specification, which denotes a user or set of users; an action specification, which denotes an operation (e.g., read a file) or a group of operations; and an object specification, which specifies an object or group of objects.

For example, if we were to specify a policy statement as an English sentence, "User1 may read ftp://myhost.edu/myfile", then the user specification would be "User1", the action specification would be "file/read", and the object specification would be ftp://myhost.edu/myfile. In reality, the CAS server maintains this information as entries in database tables and translates it into a policy language when responding to a query.

The policy_statement table consists of (userGroup, actionSpec, actionSpecDesc, objectSpec, objectSpecDesc) tuples corresponding to the policy statements implied by this relationship:

Table 5.4. Policy Statement Table

userGroup	A reference to an entry in the user_group table.
actionSpec	A reference to an entry in the service_action table or service_action_group table.
actionSpecDesc	Either a "serviceAction" or "serviceActionGroup" describing the actionSpec entry.
objectSpec	A reference to an entry in the object table, object_group_entry table, user table, user_group table, service_type table, namespace table or trust_anchor table.
objectSpecDesc	Either a "object", "objectGroup", "user", "userGroup", "serviceType", "namespace" or "trustAnchor" describing the objectSpec entry.

Each statement implies that users who belong to the userGroup are permitted to perform the service/action or all service/action(s) in the serviceActionGroup on the specified object or all objects in the said object group.

2. CAS Permissions

A user (U) is said to have permission to perform service/action S/A on object (O) if there is a statement in the policy_statement table that meets these three conditions:

1. The user element applies: U appears in the user table and the user element is either:
 - a user_group_specification referring to a user_group containing U, or
 - the community_specification.
2. The action element applies:
 - the action specification refers to service_type S and the action A.
 - the action specification refers to a service_action group that has service type S and action A as a member.
 - the action specification is superuser.
3. The object element applies: it's either:
 - An object that "matches" O - that is, the appropriate matching function (based on the namespace that the object belongs to) applied to O and the object_name yields a match, or
 - An object group that contains an object that "matches" O.

3. CAS Requests

CAS requests can be broadly classified into administrative requests and query requests. Each CAS request requires some set of permissions. These permission are assessed by looking up the authenticated user in the user table (to get the CAS nickname mapped to this user) and using that to check if the policy table has a permission defined for the operation as described in the previous section.

4. CAS Administrative Requests

Administrative requests typically modify the database and are used to add or remove CAS table entries.

- Enroll or unenroll trust anchors
- Enroll or unenroll users
- Create or delete namespaces
- Create or delete objects
- Create or delete service types
- Add or remove service type/action mappings
- Create or delete user, object or service action groups
- Add or remove entries from any of the above
- Define groups
- Grant or revoke permissions

The above request asserts permissions and performs the operation preserving the database consistency. For example, a policy can be defined (or a right can be granted) only on objects that exist in the CAS database and so on.

Creation of any CAS object (like a trust anchor, namespace, object, user, service type, user group, object group, service_action group) allows the client to choose a user group (irrespective of whether the client belongs to the group or not) to which all permissions on the newly created object is granted. In the case of the operation where the user creates a new user group, if the client chooses to grant all permissions to the newly created user group, then the user is added to the new group.

5. CAS Query Requests

Query request are classified into

- requests that return assertions that are typically signed and can be used by the client to authorize with some resource.
- requests that return information about the current state of the CAS database.

6. Assertion requests

The CAS server supports requests to retrieve policy information as signed policy assertions. These assertions can be presented at a resource by the client for authorization purposes. A policy assertion includes a list of policy statements, the distinguished name of the user that the permissions apply to, a validity period (a start and end time corresponding

to when the assertion is valid), and is signed by the CAS server. Each of the requests for policy assertions takes a lifetime argument (the desired lifetime of the policy assertion, in seconds) and the following is done to generate them.

1. The applicable user is identified as described in [CAS Permission](#).
2. The applicable set of policy statements for the user are identified as described in [CAS Permission](#).
3. If the set is not empty, a policy assertion is created or else null is returned.
4. The assertion lifetime is calculated as follows. If the requested lifetime is 0, the server's default lifetime is used, otherwise the minimum of the requested lifetime and the server's maximum lifetime is used.
5. The list of {service/action, object} permissions, the validity time (start time is the current time, end time is the current time plus the assertion lifetime), and the applicable user's subject name is formed into a assertion.
6. The assertion is signed and returned to the requester.

In the current implementation the Security Assertion Markup Language (SAML) standard defined by [OASIS](#)¹ is used for the requests and responses involved in obtaining a authorization assertions. [OpenSAML](#)², an open source implementation of the SAML 1.0 specification, has been used as a utility to generate and process SAML queries and assertions.

The methods for the requests are:

Table 5.5. Request methods

getMaximalAssertion	This is a self-request that any user in the CAS database may make. The set of policy statements returned is the complete set of the user's permissions, for all services other than the CAS service.
getUserAssertion	In this case, an additional argument specifies the requested user; the set of policy statements is the complete set of that user's permissions for all services other than the CAS service. This request requires cas/query or cas/superuser permission on the cas/server object.
getAssertion	This is a self-request that any user in the CAS database may make. In this case, the list of {service/action, object} permissions is determined as follows: For each requested permission, if there is a policy statement granting the request in the CAS database, then the requested (service/action, object) is added to the returned assertion as a decision statement.

Assertion generation is done based primarily based on the objects. This has implications in the case of maximal assertion and user assertion generation, where all applicable polices are returned. The service restricts polices only based on objects and does not make a distinction on service type. For example, if a non-implicit object has a CAS service type policy on it, it will be returned as a part of the assertion. This might be useful in case some other CAS server is itself being treated as an external resource and the CAS service types are used.

7. OGSA-AuthZ Service Interface

The CAS service also exposes a OGSA-AuthZ Service Interface that allows a client to query the server to ascertain rights. The interface allows for a signed SAML Request to be passed as parameter and pull down the rights as a SAML Response. The SAML Request allows for a query with a subject, resource and object to be passed as parameter. The

¹ <http://www.oasis-open.org>

² <http://www.opensaml.org>

server retrieves rights for the said parameters and returns SAML Authorization Assertions as part of response, if permission exists.

This functionality allows for a pull model, where the a resource can retrieve the rights of a subject from the CAS server. This is an alternative to the push model where the client retrieves assertion about itself and pushes it to the resource. Also, since the interface is a standard authorization service, a callout written to talk to any OGSA AuthZ compliant authorization service can be used.

The resource would need to be configured with information about the CAS Service (like location, CAS credential DN). Following is a list of steps that would be required to retrieve assertions from the CAS server using this interface.

1. Construct SAML Query containing the subject requesting access, the action and resource for which access is requested.
2. Construct SAML Request and sign it.
3. Use CAS's OGSA AuthZ interface and retrieve SAML Response for the request
4. Verify integrity of the SAML Response
5. Ensure the CAS server contacted is in a list of trusted servers (based on the DN of the signing entity)
6. Parse SAML Response to see if server returned an assertion with rights for access.
7. If no assertion was returned, deny access. If assertion was returned, permit access.



Note

Above is just a suggested list of steps. The decision retrieved from assertion might have to be augmented with other authorization schemes prior to permitting access to a resource.

A sample client **query-cas-service** shows how this interface can be used to retrieve assertions.

Chapter 6. APIs

1. Programming Model Overview

The CAS service allows for managing fine grained access policy of resources in a VO. The service has a back end database that stores data about users/resources/actions and the associated rights. It provides an administrative interface for managing the data and a query interface that allows users to retrieve this information. One of the operations in the query interface includes a means to get a signed SAML assertion that the client can present to a resource for authorization purposes.

2. Component API

Some relevant APIs:

- `org.globus.cas.CasPortType`
- `org.globus.cas.impl.CasConstants`
- `org.globus.cas.impl.client.CasProxyHelper`

Complete API:

- [Service API](#)¹
- [Common API](#)²
- [Client API](#)³
- [Util API](#)⁴

¹ http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_service_java/

² http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_common/

³ http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_client_java/

⁴ http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_utils_java/

Chapter 7. Services and WSDL

1. Protocol overview

This component is used to store and retrieve assertions about the rights a user has on some resource to perform some action on a service type. It uses the [Security Assertion Markup Language \(SAML\)](#)¹ to express an authorization query and return an assertion about the objects in the query. It also provides a WSDL interface for administrative tasks such as managing information about users and resources as well as granting and revoking rights on them.

2. Operations

- `addUser`: Adds a new user.
- `removeUser`: Removes a user.
- `addTrustAnchor`: Adds a new trust anchor.
- `removeTrustAnchor`: Removes a trust anchor.
- `createGroup`: Creates a new user, object or action group.
- `deleteGroup`: Deletes a user, object or action group.
- `createObject`: Creates a new object (resource).
- `deleteObject`: Deletes an object (resource).
- `createObjectNamespace`: Creates a new object namespace.
- `deleteObjectNamespace`: Deletes an object namespace.
- `manageObjectGroups`: Adds/deletes objects to an object group.
- `manageUserGroups`: Adds/deletes objects to an user group.
- `createServiceType`: Creates a new service type.
- `deleteServiceType`: Deletes service type.
- `manageServiceAction`: Adds/deletes service type and action mapping.
- `manageServiceActionGroups`: Creates/deletes a new service/action group.
- `grant`: Grants a user the right to perform service/action (or a group of service/actions) on a resource (or a group of resources).
- `revoke`: Revokes a user's right.
- `whoami`: Returns the CAS nickname associated with the user.
- `list`: Returns the list of users/objects/service/action types.

¹ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

- `findApplicablePolicy`: Returns all relevant policy for a said user/object/service/action.
- `getCasObject`: Returns the user/object/service/action represented by Object given a id.
- `getGroupMembers`: Returns all members for a given user/object/service/action group.
- `getAssertion`: Returns an assertion for a said query that contains the rights that the user for the action and resource specified in the query.
- `SAMLRequest`: Returns a SAML Response for a said SAML Rquest, which contains a SAML Query. This is the implementation of the OGSA AuthZ specification for authorization service.

3. CAS Resource Properties

- `ServerDN`: The DN from the credentials used by the CAS Service
- `VODescription`: This is a string that describes the VO relevant to CAS Service.

4. Faults

- `NoPermissionFault`: Throws if the client is not allowed to invoke the operation.
- `CasFault`: Throws if any other error occurs.

5. WSDL and Schema Definition

- [CAS WSDL](#)²
- [CAS schema file documentation](#)³

² http://viewcvs.globus.org/viewcvs.cgi/ws-cas/common/schema/cas/cas_flattened.wsdl?rev=1.2&only_with_tag=globus_4_2_0&content-type=text/vnd.viewcvs-markup

³ [../wsgram/schemas/cas_types.html](#)

CAS Admin Commands

Name

cas-proxy-init -- Generate a CAS proxy

```
cas-proxy-init [common options] [ -p proxyfile | -t tag ]
```

Tool description

The **cas-proxy-init** command contacts a CAS server, obtains an assertion for the user, and embeds it in a credential. This credential can be used to access CAS-enabled services.

Options

Command-specific options

-b Generate a CAS credential that includes only those permissions specified in file *policyFileName* (the default *policyFile* is to generate a credential with all the user's permissions). Details about the template of the file is provided [here](#).

-u Choose a filename in which to store the CAS credential based on the value *tag*. Cannot be used with the *-p* option.

-w Specify the file in which to store the CAS credential. Cannot be used with the *-t* option.

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

- | | |
|--------------------------------|--|
| -a, --anonymous | Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism. |
| -c, --serverCertificate <file> | Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism. |
| -debug | Runs the client with debug message traces and error stack traces. |
| -f, --descriptor <file> | Specifies a client security descriptor. Overrides all other security settings. |
| -help | Prints the usage message for the client. |
| -l, --contextLifetime <value> | Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism. |

<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

The following gets the assertion from the CAS server, generates a proxy with the assertion and writes it out to "casProxy".

```
cas-proxy-init -p casProxy
```

Requesting specific permissions from the CAS server

It is possible to request specific permissions from the CAS server using the `-f` option. This option causes **cas-proxy-init** to read a set of requested rights from a file.

This file should contain one or more resource identifiers:

Resource: *ResourceNamespace* | *ResourceName*

For each resource, there should be one or more action identifiers:

serviceType *action*

For example, if the client needed assertions for "file/read" service/action (permission) on two resources ("ftp://sample1.org" and "ftp://sample3.org", both in "FTPNamespace") but "directory/read" and "directory/write" permissions on the former resource only, the policy file should have the following entries:

Resource: FTPNamespace | ftp://sample1.org

file read

directory read

directory write

Resource: FTPNamespace | ftp://sample3.org

file read

To indicate any resource, the following wildcard notation should be used:

uri:samlResourceWildcard

To indicate any action, the following wildcard notation for *serviceType* and *action* should be used. Note that this should be the first (and clearly the only action) in the list of actions specified. All other actions in the list are ignored and if it is not the first, it is not treated as a wildcard.

uri:samlActionNSWildcard uri:samlActionWildcard

For example, if the client needs assertions for all resources and all actions, the policy file should look like:

Resource: uri:samlResourceWildcard

uri:samlActionNSWildcard uri:samlActionWildcard

If the client needs assertions for all actions on resource "FTPNamespace|ftp://sample1.org", the policy file should be as follows:

Resource: FTPNamespace | ftp://sample1.org

uri:samlActionNSWildcard uri:samlActionWildcard

Name

cas-wrap -- Runs program with CAS credentials

```
cas-wrap [common options] [ -p proxyfile | -t tag ]
```

Tool description

The **cas-wrap** command runs a grid-enabled program, causing it to use previously-generated CAS credentials.

This command invokes the given command with the given argument using the specified previously-generated CAS credential. For example:

```
casAdmin$ cas-wrap -t my-community gsincftp myhost.edu
```

will look for a credential generated by a previous execution of:

```
casAdmin$ cas-proxy-init -t my-community
```

and then set the environment to use that credential while running the command:

```
casAdmin$ gsincftp myhost.edu
```

The second form should be used if **cas-proxy-init** was run with the `-p` option. For example:

```
casAdmin$ cas-wrap -p /path/to/my/cas/credential gsincftp myhost.edu
```

will look for a credential generated by a previous execution of:

```
casAdmin$ cas-proxy-init -p /path/to/my/cas/credential
```

and then set the environment to use that credential while running the command:

```
casAdmin$ gsincftp myhost.edu
```

Options

Command-specific Options

`-p` Specify the file in which to store the CAS credential. Cannot be used with the `-t` option.

~~proxy~~
file

`-t` Choose a filename in which to store the CAS credential based on the value *tag*. Cannot be used with the `-p` option.

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

`-a, --anonymous` Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.

<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <code><fqdn></code> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

Example of using cas-wrap to transfer a file.

```
cas-wrap -p casProxy globus-url-copy gsiftp://somehost.edu/some_file_path \  
file:///some_file_path
```

Name

cas-enroll -- Enroll a CAS Object

```
cas-enroll [common options] trustAnchor userGpName nickname authMethod authData cas-enroll [common options] namespace userGpName nickname basename comparisonAlg cas-enroll [common options] object userGpName objectName namespaceNick cas-enroll [common options] serviceType userGpName serviceTypeName
```

Tool description

This command line client is used to enroll a CAS Object, which includes trust anchors, namespaces, objects and service types.

Enrolling Trust Anchors

To enroll a trust anchor, the user must have `cas/enroll_trustAnchor` permission on that CAS server object (that is, the user must have permission to perform the `enroll_trustAnchor` action on the CAS service type).

The enroll operation allows the user to choose a user group to which `cas/grantAll` permission on the enrolled object should be granted. The nickname should be unique across the CAS database and is used to refer to this trust anchor.

To enroll trust anchors:

```
casAdmin$ cas-enroll [common options] trustAnchor userGpName nickname authMethod authData
```

where:

userGpName Indicates the user group to which `cas/grantAll` permission should be granted on this trust anchor entity.

nickname Indicates the trust anchor nickname.

authMethod Indicates the authentication method used by the trust anchor.

authData Indicates the data used for authentication, typically the DN.

Enrolling Namespaces

To enroll a namespace, the user must have `cas/enroll_namespace` permission (that is, the user must have permission to perform the `enroll_namespace` action on the cas service type).

The enroll operation allows the user to choose a userGroup to have `cas/grantAll` permission on the enrolled object. The comparison algorithm specified should be the name of the Comparison class that needs to be used to compare objects that belong to this namespace. The nickname should be unique across the CAS database and is used to refer to this user.

Also, two namespaces are added to the CAS database at boot up time, other than the inherent CAS Namespace:

- `FTPDirectoryTree` uses the `WildcardComparison` Algorithm and has the base URL set to the current directory.
- `FTPEXact` uses the `ExactComparison` Algorithm and has the base URL set to the current directory.

To enroll namespaces:

```
casAdmin$ cas-enroll [common options] namespace userGpName nickname basename comparisonAlg
```

where:

<i>userGpName</i>	Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.
<i>nickname</i>	Indicates the nickname of the namespace to be unenrolled. If the trust anchor nickname specified does not exist, an error is <i>not</i> thrown. If the unenroll operation is successful, all policy data on that trust anchor is purged.
<i>basename</i>	Indicates the base URL for the namespace.
<i>comparisonAlg</i>	Indicates the comparison algorithm to be used. Unless the standard comparison algorithms described below are used, the fully qualified name of the class that needs to be used should be given. The class needs to extend from the abstract class <code>org.globus.cas.impl.service.ObjectComparison</code> .

The two comparison classes provided as a part of the distribution are:

- `ExactComparison`: This class does a case-sensitive exact comparison of the object names. If `comparisonAlg` in the above method is set to `ExactComparison`, the class in the distribution is loaded and used.
- `WildcardComparison`: This class does wild card matching as described in [CAS Simple Policy Language](#)¹. It assumes that the wild card character is "*" and that the file separator is "/". If `comparisonAlg` in the above method is set to `WildcardComparison`, the class in the distribution is loaded and used.

Enrolling Objects

To enroll an object, the user must have cas/enroll_object permission (that is, the user must have permission to perform the enroll_object action on the cas service type).

The enroll operation allows the user to choose a userGroup to have cas/grantAll permission on the enrolled object. The name of the object and the namespace this object belongs to identify an object in the database and should be unique across the CAS database.

To enroll objects:

```
casAdmin$ cas-enroll [common options] object userGpName objectName namespaceNick
```

where:

<i>userGpName</i>	Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.
<i>objectName</i>	Indicates the name of the object.
<i>namespaceNick</i>	Indicates the nickname of the namespace to which this object belongs.

¹ http://www.globus.org/toolkit/docs/3.2/cas/CAS_policy_language_0.2.pdf

Enrolling Service Types

To enroll a service type, the user must have `cas/enroll_serviceType` permission (that is, the user must have permission to perform the `enroll_serviceType` action on the cas service type).

The enroll operation allows the user to choose a `userGroup` to have `cas/grantAll` permission on the enrolled service type. The service type name should be unique across the CAS database.

To enroll service types:

```
casAdmin$ cas-enroll [common options] serviceType userGpName serviceTypeName
```

where:

`userGpName` Indicates the user group to which `cas/grantAll` permission should be granted on this trust anchor entity.

`serviceTypeName` Indicates the service type name.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

<code>-a, --anonymous</code>	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"> • <code>msg</code> for GSI Secure Message, or • <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"> • <code>sig</code> for signature, or • <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here .

The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.

- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For detailed examples of using this command, see [Chapter 6, Example of CAS Server Administration](#) .

Name

cas-remove -- Remove a CAS object from the database

```
cas-remove [common options] trustAnchor nickname cas-remove [common options] namespace
nickname cas-remove [common options] object objName namespaceNick cas-remove [common
options] serviceType serviceTypeName
```

Tool description

Removing Trust Anchors

To remove a trust anchor, the user must have cas/remove permission on that trust anchor. The trust anchor must also be unused (that is, there may not be any users in the database that have this trust anchor or it may not be a part of any object group).

To remove trust anchors:

```
casAdmin$ cas-remove [options] trustAnchor nickname
```

where:

nickname Indicates the nickname of the trust anchor to be unenrolled.

If the trust anchor nickname specified does not exist, an error is *not* thrown. If the unenroll operation is successful, all policy data on that trust anchor is purged.

Removing Namespaces

To remove a namespace, the user must have cas/remove permission on that namespace. The namespace must also be unused — that is, there may not be any object in the database that belongs to this namespace.

```
casAdmin$ cas-remove [options] namespace nickname
```

where:

nickname Indicates the nickname of the namespace to be unenrolled.

If the namespace nickname specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that trust anchor is purged.

Removing Objects

To remove an object the user must have cas/remove permission on that object. The object must also be unused — that is, there may not be any object group in the database that this object belongs to.

```
casAdmin$ cas-remove [options] object objName namespaceNick
```

where:

objName Indicates the name of the object to be removed.

namespaceNick Indicates the nickname of the namespace to which this object belongs.

If the object specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that object is purged.

Removing Service Types

To remove a service type the user must have cas/remove permission on that service type. The service type must also be unused — that is, there may not be any service type to action mapping.

```
casAdmin$ cas-remove [options] serviceType serviceTypeName
```

where:

serviceTypeName Indicates the service type name.

If the service type specified does not exist, an error is *not* thrown. If the remove operation is successful, all policy data on that service type is purged.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s <i>cas-url</i>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrf/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.

- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-action -- Maintains service types

```
cas-action [common options] [ add | remove ] serviceTypeName actionName
```

Tool description

Use the **cas-action** command to add an action mapping to a service type or remove an action mapping from a service type.

To add an action mapping to a service type, the user must have `cas/create_group_entry` permission on the service type.

To remove a service type action mapping, the user must have `cas/delete_group_entry` permission on the service type.

If the group member being removed does not exist, an error is *not* thrown.

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
-s <i>cas-url</i>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here .

The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.

- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For an example of using this command, see [Section 9, “Adding action mappings”](#).

Name

cas-group-admin -- Maintains user groups, object groups, or serviceAction groups

```
cas-group-admin [common options] [ user | object | serviceAction ] create userGpName groupName cas-  
group-admin [common options] [ user | object | serviceAction ] delete groupName
```

Tool description

Use **cas-group-admin** to create or delete user groups, object groups, or serviceAction groups. Note: to add or delete entries to these groups, see [\[fixme olink to other clients\]](#).

Adding user groups

To create a new user group the user must have cas/create_user_group permission (that is, the user must have permission to perform the create_user_group action on the cas service type). The user group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created user group. If the user group that is chosen to have cas/grantAll permission is the new group created, then the user making this request is added to the new group.

To add a user group:

```
casAdmin$ cas-group-admin [common options] user create userGpName groupName
```

where:

userGpName Indicates the user group to which cas/grantAll permission should be granted on this trust anchor entity.

groupName Indicates the name of the user group being created.

Deleting user groups

To delete a user group, the user must have cas/delete_user_group entry permission on that user group. The group must be empty and also must not be referenced from other entities in the database (for example, it should not be a member of some object group).

If the user group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that user group is purged.

```
casAdmin$ cas-group-admin [common options] user delete groupName
```

where:

groupName Indicates the name of the user group to be deleted.

Creating An Object Group

To create a new object group, the user must have cas/create_object_group permission (that is, the user must have permission to perform the create_object_group action on the CAS service type). The object group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created object group.

```
casAdmin$ cas-group-admin [common options] object create userGpName groupName
```

where:

userGpName Indicates the user group to which cas/grantAll permission should be granted on this object group.

groupName Indicates the object group name.

Deleting An Object Group

To delete an object group, the user must have cas/delete_user_group entry permission on that object group. The group must be empty.

If the object group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that object group is purged.

```
casAdmin$ cas-group-admin [common options] object delete groupName
```

where:

groupName The name of the object group to be deleted.

Creating A Service/Action Group

To create a new service/action group, the user must have cas/create_serviceAction_group permission (that is, the user must have permission to perform the create_serviceAction_group action on the CAS service type). The serviceAction group name should be unique across the CAS database. The create operation allows the user to choose a user group to have cas/grantAll permission on the created serviceAction group.

```
casAdmin$ cas-group-admin [common options] serviceAction create userGpName groupName
```

where:

userGp-Name Indicates the user group to which cas/grantAll permission should be granted on this service/action group.

groupName Indicates the name of the service/action group being created.

Deleting A Service/Action Group

To delete a service/action group, the user must have cas/delete_user_group entry permission on that service/action group. The group must be empty and also must not be referenced from any other entity in the database. For example, it should not be a member of some object group.

If the service/action group specified does not exist, an error is *not* thrown. If the delete operation is successful, all policy data on that service/action group is purged.

```
casAdmin$ cas-group-admin [common options] serviceAction delete groupName
```

where:

gp Indicates the name of the service/action group to be deleted.

~~gp~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s cas-url	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrf/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z authorization	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For examples of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Name

cas-group-add-entry -- Adds CAS objects to CAS groups

```
cas-group-add-entry [common options] user groupName nickname cas-group-add-entry
[common options] object groupName objectSpecDesc objectSpec cas-group-add-entry [common
options] serviceAction groupName serviceTypeName actionName
```

Tool description

Use **cas-group-add-entry** to add users to a user group, objects to an object group, or service/actions to service/action groups. Note: to add or delete groups, see [fixme olink to other clients].

Adding Member To A User Group

To add a user to a user group, the user must have cas/add_group_entry permission on that particular user group. Only user nicknames that exist in the CAS database can be valid members.

```
casAdmin$ cas-group-add-entry [common options] user groupName nickname
```

where:

~~gp~~ Indicates the user group name to which the member needs to be added.

~~nk~~

~~nk~~ Indicates the nickname of the user to be added to this group.

~~nk~~

Adding Member To An Object Group

To add a member (an object group can have the following CasObjects as members: object, user, user group, service type, namespace or trust anchor) to an object group, the user must have cas/add_group_entry permission on that particular object group.

```
casAdmin$ cas-group-add-entry [common options] object groupName objectSpecDesc objectSpec
```

where:

~~gp~~ Indicates the object group name to which the member needs to be added.

~~nk~~

~~ob~~ Indicates the type of CasObject. Can be one of the following options:

~~ob~~

~~ob~~ • trustAnchor

~~ob~~

• user

• userGroup

• object

• namespace

• serviceType

~~o~~ Indicates the identifier for the CasObject the user is adding. Can be one of the following:

- ~~g~~
 - nickname if adding a trustAnchor or user
 - groupName if adding a userGroup
 - objectNamespace objectName if adding an object
 - nickname if adding a namespace
 - serviceTypeName if adding a serviceType

Adding Service/Action To A Service/Action Group

To add a service/action to a serviceAction group, the user must have cas/add_group_entry permission on that particular serviceAction group (that is, the user must have permission to perform add_group_entry action on that service action group).

```
casAdmin$ cas-group-add-entry [common options] serviceAction groupName serviceTypeName act
```

where:

~~g~~ Indicates the service/action group to which the service/action needs to be added.

~~h~~

~~s~~ Indicates the service type name part of the mapping to be added to the group.

~~v~~

~~t~~

~~h~~

~~a~~ Indicates the action name part of the mapping to be added to the group.

~~h~~

~~h~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be:

	<ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For examples of using this command, see [Chapter 6, Example of CAS Server Administration](#).

Name

cas-group-remove-entry -- Removing CAS objects from CAS groups

```
cas-group-remove-entry [common options] user groupName nickname cas-group-remove-  
entry [common options] object groupName objectSpec objectSpecDesc cas-group-remove-  
entry [common options] serviceAction groupName serviceTypeName actionName
```

Tool description

Use **cas-group-remove-entry** to remove users from a user group, objects from an object group, or service/actions from a service/action group. Note: to add or delete groups, see [\[fixme olink to other clients\]](#).

Removing User From A User Group

To remove a user from a user group, the user must have `cas/remove_group_entry` permission on that particular user group.

If the group member being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] user groupName nickname
```

where:

~~gp~~ Indicates the user group name from which the member needs to be removed.

~~nk~~

~~nk~~ Indicates the nickname of the user to be removed from this group.

~~nk~~

Removing Member From An Object Group

To remove an object from an object group the user must have `cas/remove_group_entry` permission on that particular object group:

If the group member being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] object groupName objectSpec objectSpecDe
```

where:

~~gp~~ Indicates the object group name from which the member needs to be removed.

~~nk~~

~~ob~~ Indicates the type of CasObject. Can be one of the following options:

~~ob~~

~~ob~~ • trustAnchor

~~ob~~

• user

• userGroup

• object

• namespace

- `serviceType`

~~o~~ Indicates the identifier for the CasObject the user is adding. Can be one of the following:

- ~~g~~
 - `nickname` if adding a trustAnchor or user
 - `groupName` if adding a userGroup
 - `objectNamespace objectName` if adding an object
 - `nickname` if adding a namespace
 - `serviceTypeName` if adding a serviceType

Removing A Service/Action From A Service/Action Group

To remove a service/action from a service/action group, the user must have `cas/remove_group_entry` permission on that particular service/action group.

If the action being removed does not exist, an error is *not* thrown.

```
casAdmin$ cas-group-remove-entry [common options] serviceAction groupName serviceTypeName
```

where:

~~g~~ Indicates the serviceAction group name from which the service/action needs to be removed.
~~g~~

~~s~~ Indicates the service type name part of the mapping to be removed from the group.
~~v~~
~~t~~
~~g~~

~~s~~ Indicates the action name part of the mapping to be removed from the group.
~~t~~
~~g~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

<code>-a, --anonymous</code>	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
<code>-c, --serverCertificate <file></code>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
<code>-debug</code>	Runs the client with debug message traces and error stack traces.
<code>-f, --descriptor <file></code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.

<code>-l, --contextLifetime <value></code>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
<code>-m, --securityMech <type></code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
<code>-p, --protection <type></code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
<code>-s cas-url</code>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
<code>-v</code>	Prints the version number.
<code>-x, --proxyFilename <value></code>	Sets the proxy file to use as client credential.
<code>-z authorization</code>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <code><fqdn></code> is the fully qualified domain name of the host on which the CAS service is up.

 **Note**

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-rights-admin -- Granting or revoking permissions

```
cas-rights-admin [common options] [grant|revoke] userGroupName objectSpecDesc objectSpec  
actionSpecDesc actionSpec
```

Tool description

Use **cas-rights-admin** to grant or revoke rights.

Granting Permissions To A User Group On An Object/Object Group

The user may grant permissions to a user group on an object or object group to perform a service action or service action group (that is, to perform any action that is a member of the service action group to which permission is granted), provided the user has both:

- cas/grant permission on the object or object group, and
- permission to perform the service action or service action group on the object or object group.

```
casAdmin$ cas-rights-admin [common options] grant userGroupName objectSpecDesc objectSpec
```

where:

~~⌘~~ Indicates the user group to be granted permission.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the identifier for the object or object group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type:

- ~~⌘~~
- object
 - objectGroup

~~⌘~~ Indicates the identifier for action or action group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type:

- ~~⌘~~
- serviceAction
 - serviceActionGp

Revoking A Policy In The CAS Database

The user may revoke a policy in the CAS database if the user has cas/revoke permission on the object or object group on which the policy is defined.

```
casAdmin$ cas-rights-admin [common options] revoke userGroupName objectSpecDesc objectSpec
```

where:

~~⌘~~ Indicates the user group for which you want to grant permission.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type of CasObject. Can be one of the following:

~~⌘~~
~~⌘~~
~~⌘~~

- trustAnchor
- user
- userGroup
- object
- namespace
- serviceType
- userGroup

~~⌘~~ Indicates the identifier for the object or object group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the identifier for the action or action group.

~~⌘~~
~~⌘~~

~~⌘~~ Indicates the type (serviceAction or serviceActionGp).

~~⌘~~
~~⌘~~
~~⌘~~

Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.

- `-m, --securityMech <type>` Specifies the authentication mechanism. The value *type* can be:
- `msg` for GSI Secure Message, or
 - `conv` for GSI Secure Conversation.
- `-p, --protection <type>` Specifies the protection level. *type* can be:
- `sig` for signature, or
 - `enc` for encryption.
- `-s cas-url` Sets the CAS Service instance, where *cas-url* is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown [here](#).
- The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.
- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where *<fqdn>* is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Usage

For an example of using this command, see [Chapter 6, Example of CAS Server Administration](#).

CAS Query Commands

The CAS Query commands do not alter the state of the database and any CAS user who has cas/query permissions may use the commands to retrieve data from the CAS server.

The following queries can be run against the CAS server. These are typically used by CAS clients (who may not be administrators).

The user need cas/query permissions to perform these operations—that is, the user must have permission to query on the cas server object.

Name

cas-whoami -- Getting a user's CAS identity.

cas-whoami [*options*]

Tool description

The **cas-whoami** command returns the CAS user nick of the client.

Command options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s <i>cas-url</i>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z <i>authorization</i>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> .

If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.

Alternatively, an environment variable can be set as shown [here](#).

If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-list-object -- Getting object list

cas-list-object [*options*] *type*

Tool description

The **cas-list-object** command returns a list of CasObjects in the database of the requested type.

Command Options

~~Use~~ Use one of the following to indicate the type of of CasObjects you want listed:

- trustAnchor
- user
- userGroup
- object
- objectGroup
- objectGroup
- namespace
- serviceType
- serviceAction
- serviceActionGp

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.

- `-m, --securityMech <type>` Specifies the authentication mechanism. The value *type* can be:
- `msg` for GSI Secure Message, or
 - `conv` for GSI Secure Conversation.
- `-p, --protection <type>` Specifies the protection level. *type* can be:
- `sig` for signature, or
 - `enc` for encryption.
- `-s cas-url` Sets the CAS Service instance, where *cas-url* is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown [here](#).
- The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where *Host* and *Port* are the host and port where the container with the CAS service is running.
- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-get-object -- Getting CAS object

cas-get-object [*options*] *type name*

Tool description

The **cas-get-object** command returns the particular object of the said type and name.

Command Options

te Use one of the following to indicate the type of of CasObjects you want to get:

- trustAnchor
- user
- userGroup
- object
- objectGroup
- namespace
- serviceType
- serviceAction
- serviceActionGp

me Use one of the following to indicate the name of the specific CAS object you want to get:

- *nickname* (if getting trustAnchor, user, userGroup, or namespace)
- *objectNamespace objectName* (if getting object or objectGroup)
- *serviceTypeName* (if getting serviceType, serviceAction, or serviceActionGp)

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.

-f, --descriptor <i><file></i>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <i><value></i>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <i><type></i>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• <code>msg</code> for GSI Secure Message, or• <code>conv</code> for GSI Secure Conversation.
-p, --protection <i><type></i>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• <code>sig</code> for signature, or• <code>enc</code> for encryption.
-s <i>cas-url</i>	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <i><value></i>	Sets the proxy file to use as client credential.
-z <i>authorization</i>	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <i><fqdn></i> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-group-list-entries -- Getting group members

cas-group-list-entries [*options*] *type name*

Tool description

The **cas-group-list-entries** command returns a list of group members.

Command Options

o Use one of the following to indicate the type of group for which you want a list of members:

- user
- object
- serviceType

n The name of the group.

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.

- `-s cas-url` Sets the CAS Service instance, where `cas-url` is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown [here](#).
The instance URL typically looks like `http://Host:Port/wsrp/services/CASService`, where `Host` and `Port` are the host and port where the container with the CAS service is running.
- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
Alternatively, an environment variable can be set as shown [here](#).
If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

cas-find-policies -- Getting policy information

```
cas-find-policies [options] [-c cas-url] type name
```

Tool description

The **cas-find-policies** command returns all applicable policies, both policies that are implicit to the CAS server and those that are external.

Command options

-c cas-url The URL of the CAS service.

type Use one of the following to indicate the type of CasObjects:

- trustAnchor
- user
- userGroup
- object
- objectGroup
- namespace
- serviceType
- serviceAction
- serviceActionGp

name Use the type of name corresponding to the appropriate CasObject:

- *nickname* (for trustAnchors, users, or namespaces)
- *groupName* (for userGroups, objectGroups, or serviceActionGps)
- *objectNamespace/objectName* (for objects)
- *serviceTypeName (or) serviceType/Action* (for serviceTypes or serviceActions)

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.

-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.
-s cas-url	Sets the CAS Service instance, where <i>cas-url</i> is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown here . The instance URL typically looks like <code>http://Host:Port/wsrp/services/CASService</code> , where <i>Host</i> and <i>Port</i> are the host and port where the container with the CAS service is running.
-v	Prints the version number.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z authorization	Specifies the type of authorization used, such as <code>self</code> or <code>host</code> . If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value. Alternatively, an environment variable can be set as shown here . If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/<fqdn></code> , where <code><fqdn></code> is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport, then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Name

query-cas-service -- Query CAS Service (using OGSA AuthZ interface)

query-cas-service [*options*] *assertionFilename*

Tool description

The **query-cas-service** command returns a SAML Response containing SAML Assertions with user rights for a given SAML Query. This client uses the OGSA AuthZ interface and writes out the retrieved assertion to a file.

Command options

~~⌘~~ File to write assertions to.

~~⌘~~

~~⌘~~

~~⌘~~

~~⌘~~

~~⌘~~

Common Options

Important

If you have an asterisk (*) in your command, you might need to escape it with a backslash (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none">• msg for GSI Secure Message, or• conv for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none">• sig for signature, or• enc for encryption.

- `-s cas-url` Sets the CAS Service instance, where `cas-url` is the URL of the CAS service instance. Alternatively, an environment variable can be set as shown [here](#).
- The instance URL typically looks like `http://Host:Port/wsrf/services/CASService`, where `Host` and `Port` are the host and port where the container with the CAS service is running.
- `-v` Prints the version number.
- `-x, --proxyFilename <value>` Sets the proxy file to use as client credential.
- `-z authorization` Specifies the type of authorization used, such as `self` or `host`.
- If you cannot use a standard method for authorization, you can use the specific CAS server's identity as the value.
- Alternatively, an environment variable can be set as shown [here](#).
- If none of the above are set, host authorization is done by default and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.



Note

If the service being contacted is using GSI Secure Transport , then the container credentials configured for the service will be used, even if service/resource level credentials are configured. Hence authorization needs to be done based on the DN of the container credentials.

Chapter 8. Semantics and syntax of domain-specific interface

- To get a handle for the CAS service port type, use the `org.globus.cas.impl.client.CasClientSetup` class.

Sample Code:

To get a handle to a CAS service with instance URL *instanceURL* and identity *serviceIdentity*:

```
CasClientSetup clientSetup = new CasClientSetup();  
  
CommunityAuthorizationServicePortType casPort =  
  
    clientSetup.getCASPort(instanceURL, serverIdentity);
```

- To generate a proxy with embedded CAS assertions, use the API in the class `org.globus.cas.impl.client.CasProxyHelper`. The class `org.globus.cas.impl.client.ClientParams` is used to pass in appropriate parameters and the datatype `org.globus.cas.impl.client.ResourceActionsMap` is used to represent the resource/actions mapping for which assertions are requested on.

Listed below is sample code that uses the client side util API to generate a proxy with CAS assertions embedded in it.

1. The `ClientParams` class is used to construct the parameter. If the default constructor is used and none of the values are set then the requested assertion lifetime is set to 24 hours, the default proxy file is used and the proxy containing the embedded assertions is named with a ".cas" extension at the end of the proxy file.

```
ClientParams clientParams = new ClientParams();
```

2. The following is used to set assertion lifetime. If not set then 24 hours is used.

```
clientParams.setAssertionLifetime(lifetime);
```

3. Set the file name of the proxy to use. If not set then the default credential is used.

```
clientParams.setProxyFileName(proxyFilename);
```

4. Set the file name that the proxy with CAS assertions will be written to. If not set then original proxy file name is appended with a tag.

```
clientParams.setCasProxyFileName(casProxyFilename);
```

5. Set the extension to append to the original proxy filename. If not set then the extension ".cas" is used. The extension is only used if a filename for the CAS proxy is not set.

```
clientParams.setCasProxyTag(tag);
```

6. Set the resource/actions for which the assertion is requested on. It uses an array of data type *ResourceActionsMap* (explained below).

```
clientParams.setResourceActionsMap(resActions);
```

7. The *ResourceActionsMap* datatype is used to represent the resource and the actions on the resource for which the permissions are required. It uses a *String* to represent the resource and a vector of strings to represent the actions.

The resource should be of the form "*objectNamespace|objectName*". The action should be of the form "*serviceType actionName*".

8. Create an instance of the *Helper* class:

```
CasProxyHelper casProxyHelper = new CasProxyHelper(instanceURL, serverIdentity);
```

Where:

- *instanceURL* is the URL used to contact the CAS service.
- *serverIdentity* is the expected identity of the server. If null, host authorization is used.

9. Generate the proxy with CAS assertions:

```
String casProxyFilename = casProxyHelper.getCasProxy(clientParams);
```

This method contacts the CAS service, retrieves assertions, embeds the assertions in a proxy credential and returns the path to the proxy file.

Chapter 9. Configuring

1. Configuration overview

The CAS service can be configured with the following :

- server start up configuration
- a description of the VO the CAS service serves
- the maximum lifetime of the assertions it can issue
- information about the back end database it uses. Any database with a JDBC driver and reasonable SQL support can be used. The schema that works with Derby database, MySQL and PostGres is distributed and can be found at `$GLOBUS_LOCATION/etc/globus_cas_service/casDbSchema`.
- the security settings of the service and 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.

2. Loading the CAS service at start up

By default, the CAS service is not loaded at start up. To change this behavior, uncomment the *loadOnStartup* property set in `$GLOBUS_LOCATION/etc/globus_cas_service/server-config.wsdd` as shown below.

Once the *loadOnStartup* property is uncommented, the following happens at start up:

1. The CAS service is loaded.
2. The database connection pool is initialized.
3. The service registers itself to the default MDS Index Service.

```
<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>
```

3. Configuring the VO Description

To change the VO description, set the parameter `voDescription` in `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` to the desired values.

4. Configuring the maximum assertion lifetime

To change the maximum assertion lifetime set the parameters `maxAssertionLifetime` in `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` to the desired values.

5. Configuring database backend

To alter the configuration of the database back end edit the `databaseConfiguration` section of `$GLOBUS_LOCATION/etc/globus_cas_service/jndi-config.xml` as described below. If you are using the default Derby installation, the only parameter to change is the `connectionURL` to replace `GLOBUS_LOCATION` with the actual location of your toolkit installation.

Table 9.1. Database parameters

<code>driver</code>	The JDBC driver to be used
<code>connectionURL</code>	The JDBC connection url to be used when connecting to the database
<code>userName</code>	The user name to connect to the database as
<code>password</code>	The corresponding database password
<code>activeConnections</code>	The maximum number of active connections at any given instance
<code>onExhaustAction</code>	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)
<code>maxWait</code>	The maximum time in milliseconds that the pool will wait for a connection to be returned
<code>idleConnections</code>	The maximum number of idle connections at any given time

6. Configuring security descriptor

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 which 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 back end database to determine if the call is permitted.

To alter the security descriptor configuration refer to [Security Descriptors](#). The file to be changed is `$GLOBUS_LOCATION/etc/globus_cas_service/security-config.xml`.

 **Note**

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

 **Important**

If the service is configured to use GSI Secure Transport, then container credentials are used for the handshake, irrespective of whether service level credentials are specified.

7. Configuring with a GridFTP Server

CAS is used to administer access rights to files and directories and the GridFTP server can be configured to enforce those rights.

For detailed information about configuring CAS for use with a GridFTP server, see [How to Set up CAS with GridFTP](#).

8. Configuring CAS to manage policy for web service.

The CAS server can be used to administer rights for access to web services. The mapping from CAS objects to the web service resource is shown on this table:

Table 9.2. Mapping from web services object to CAS object

Object	EPR of WS resource as string. The OGSA-AuthZ specification defines how a EPR can be represented as a string and a utility for such is provided at <code>org.globus.wsrf.impl.security.EPRUtil</code> .
Object namespace	The object namespace is used to get both a comparison algorithm and the basename. For web services policy we need exact comparison and also don't have any base name. An implicit namespace <code>casDefaultNs</code> with the required properties is added to the service.
Service type	The OGSA-AuthZ specification defines a service type to use for web services operation as "http://www.gridforum.org/namespaces/2003/06/ogsa-authorization/saml/action/operation" This is defined as a constant in <code>org.globus.wsrf.impl.security.authorization.SAMLAuthorizationConstants</code> and is added implicitly.
Action	This is the actual operation on the webservice. For example method "add" on Counter Service.

An example scenario is described [here](#).

9. CAS auto-registration with default WS MDS Index Service

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

 **Note**

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

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

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

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

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

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

9.1. Configuring resource properties

By default, the `VoDescription` resource property (which describes the virtual organization relevant to the CAS Service) is sent to the default Index Service.

You can configure which resource properties are sent in the registration.xml file, `$GLOBUS_LOCATION/etc/globus_cas_service/registration.xml`. The following is the relevant section of the file:

```
<Content xsi:type="agg:AggregatorContent "
  xmlns:agg="http://mds.globus.org/aggregator/types">
  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
  <agg:GetResourcePropertyPollType
  xmlns:cas="http://www.globus.org/07/2004/cas">
  <!-- Specifies that the index should refresh information
  every 8 hours (28800000ms) -->
  <agg:PollIntervalMillis>28800000</agg:PollIntervalMillis>
```

```
<!-- specifies that all Resource Properties should be
collected from the RFT factory -->

<agg:ResourcePropertyName>cas:VoDescription</agg:ResourcePropertyName>

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

10. Registering CAS manually with default WS MDS Index Service

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

Chapter 10. Environment variable interface

1. Environmental variables for CAS

All CAS client programs use the following environment variables to determine the appropriate URL to connect to and server identity to expect. In all cases, the command line options takes precedence over the environment variables.

- The URL is determined using this algorithm:
 - If the `-c` command line option was specified, the URL specified with that option is used.
 - Otherwise, the `CAS_SERVER_URL` environment variable must be set, and its value is used.
- The server identity (i.e. the expected subject name of the CAS server certificate) is determined as follows:
 - If the `-s` command line option was specified, the value specified with that option is used as the identity
 - Otherwise, if the `CAS_SERVER_IDENTITY` environment variable is set, the value of that variable is used as the expected server identity. Ensure that the value is enclosed within double quotes if there are spaces in the DN. *The double quotes are required by the CAS scripts when they are run from a Windows shell, although the shell does not require it even if the value has spaces.*
 - If neither is set, host authorization is done and the expected server credential is `cas/<fqdn>`, where `<fqdn>` is the fully qualified domain name of the host on which the CAS service is up.

Chapter 11. Debugging

Log output from the CAS server is a useful tool for debugging issues. Because CAS is built on top of Java WS Core, developer debugging is the same as described in [Chapter 10, Debugging](#). For information about sys admin logs, [Chapter 9, Debugging](#).

1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)¹ API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)² as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)³.

1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁴. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

2. Enabling verbose logging

As described in the above section, configuration files need to be edited to enable logging at different levels. For example, to see all logging for the CAS server, the following lines need to be added:

```
# Server code
    log4j.category.org.globus.cas.impl.service=DEBUG
# Database access module
    log4j.category.org.globus.cas.impl.databaseAccess=DEBUG
```

¹ <http://jakarta.apache.org/commons/logging/>

² <http://logging.apache.org/log4j/>

³ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁴ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

3. Debugging info from CAS clients

Debugging information from CAS clients can be obtained by using the `-debug` option on the command line.

Chapter 12. Troubleshooting

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

1. Command run using cas-wrap does not pick up CAS proxy

Some commands require environment variables to be passed explicitly to the command (e.g java). If running such command through cas-wrap, place the command in a script (passing in the environment variable) and execute the script through cas-wrap.

For example, to run `java -DX509_USER_PROXY=$X509_USER_PROXY org.some.java.Client`, create a script `client-script` with following contents:


```
java
    -DX509_USER_PROXY=$X509_USER_PROXY org.some.java.Client
```

Ensure that the script has executable permissions and run:

```
cas-wrap -t sometag client-script
```

2. Error Messages

Table 12.1. WS A&A Authorization Framework Error Messages

Error Code	Definition
<p>org.globus.cas.impl.databaseAccess.CasDBException, connection refused</p>	<p>If the CAS service fails with following error:</p> <pre> faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.userException faultSubcode: faultString: org.apache.commons.dbcp.DbcpException: Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TC you need to troubleshoot the connection to the CAS database.</pre>
<p>CAS clients fail with org.globus.cas.impl.databaseAccess.CasDBException</p>	<p>If CAS clients fail with database permission exceptions similar to:</p> <pre> [Caused by: ERROR: permission denied for relation service_type_action]; nested exception is:org.globus.cas.impl.databaseAccess.CasDBException:</pre> <p>, then there is something wrong with user permissions on the database.</p> <p> Note</p> <p>This is a specific instance of an error for the relation <i>service_type_action</i>. This error could be raised on any rel</p>

Chapter 13. Related Documentation

- [A Community Authorization Service For Group Collaboration](#)¹
- [Using CAS to Manage Role-Based VO Sub-groups](#)²
- [The Community Authorization Service: Status and Future](#)³
- [Associated Standards](#)

¹ http://www.globus.org/alliance/publications/papers/CAS_2002_Revised.pdf

² <http://www.globus.org/alliance/publications/papers/CAS-group-CHEP03.pdf>

³ http://www.globus.org/alliance/publications/papers/CAS_update_CHEP_03-final.pdf

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.

T

transport-level security Uses transport-level security (TLS) mechanisms.

GT 4.2.1 Migrating Guide for WS A & A CAS

Table of Contents

1. Migrating CAS from GT4	1
2. Migrating CAS from GT3	1
3. Migrating CAS from GT2	1

<titleabbrev>Migrating Guide</titleabbrev>

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

1. Migrating CAS from GT4

This version is not compatible with the GT4 version of CAS because of protocol changes. To migrate to this version, this component needs to be installed completely independent of any current GT4 CAS installs.

2. Migrating CAS from GT3

This version is not compatible with the GT3 version of CAS because of protocol changes. To migrate to this version, this component needs to be installed completely independent of any current GT3 CAS installs.

3. Migrating CAS from GT2

This version is not compatible with the GT2 version of CAS because of protocol changes. To migrate to this version, this component needs to be installed completely independent of any current GT2.x CAS installs.

GT 4.2.1 CAS Quality Report

Table of Contents

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

<titleabbrev>Quality Report</titleabbrev>

1. Test coverage reports

There are no reports available at this time.

2. Code analysis reports

There are no reports available at this time.

3. Outstanding bugs

- [CAS does not work with some versions of MySQL](#)¹.
- [schema-invalid CAS assertion](#)²
- [Error in using CAS service for web-service policy management](#)³
- [Callout contacting CAS service is not using proper credential to secure calls.](#)⁴

4. Bug Fixes

- [Bug 6249](#)⁵ Upgrade Derby version

5. Performance reports

There are no performance reports available at this time.

¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=1636

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5940

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=6311

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=6312

⁵ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6249

GT 4.2.1 CAS Release Notes

Table of Contents

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

<titleabbrev>Release Notes</titleabbrev>

1. Component Overview

The Community Authorization Service (CAS) allows a virtual organization to express policy regarding resources distributed across a number of sites. A CAS server issues assertions to the virtual organization users, granting them fine-grained access rights to resources. Servers recognize and enforce the assertions. CAS is designed to be extensible to multiple services and is currently supported by the GridFTP server and web services.

2. Feature summary

Features new in GT 4.2.1:

- Support for OGSA-AuthZ Authorization Service interface
- Support for managing web services policy.

Other Supported Features

- File-level access control for GridFTP
- Issuance of SAML authorization decisions

Deprecated Features

- None

3. Summary of Changes in CAS

No changes have been made since last stable release, 4.0.x.

4. Internationalization

The CAS service code has been internationalized.

5. Bug Fixes

- [Bug 6249](#):¹ Upgrade Derby version

6. Known Problems

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

6.1. Limitations

No known limitations

6.2. Outstanding bugs

- [CAS does not work with some versions of MySQL](#)².
- [schema-invalid CAS assertion](#)³
- [Error in using CAS service for web-service policy management](#)⁴
- [Callout contacting CAS service is not using proper credential to secure calls](#).⁵

7. Technology dependencies

The CAS service depends on the following GT components:

- WS Authentication and Authorization
- Java WS Core

The CAS GridFTP authorization module depends on the following GT components:

- Non-WS Authentication and Authorization

The CAS service depends on the following 3rd party software:

- OpenSAML

The CAS GridFTP authorization module depends on the following 3rd party software:

- libxml

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=6249

² http://bugzilla.globus.org/globus/show_bug.cgi?id=1636

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5940

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=6311

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=6312

8. Tested platforms

Tested Platforms for CAS

- Windows XP
- Linux (Red Hat 7.3)

Tested Containers for CAS

- Java WS Core container
- Tomcat 5.0.30

9. Backward compatibility summary

CAS has been updated to use the latest version of Java WS Core, which now supports the final version of WSRF/WSN specification.⁶ This service is not compatible with the previous stable versions, GT 4.0.x

Fixed SAML assertions embedded in proxy to comply with RFC 3820 requirements. CAS assertions generated by default in GT 4.0.x will not be consumed by GT 4.2.x services that use assertions.

10. Associated Standards

Associated standards for CAS:

- Simple Assertion Markup Language⁷
- OGSA AuthZ Authorization Service⁸

11. For More Information

Click [here](#) for more information about this component.

⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/rn/index.html#id2534919>

⁷ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

⁸ <https://forge.gridforum.org/projects/ogsa-authz/>