

# **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.<sup>1</sup> 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

---

<sup>1</sup> <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 / 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 "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"> <li>• A user (a reference to an entry in the user table). This is used when granting permissions such as "may unenroll this user".</li> <li>• 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".</li> <li>• 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";</li> <li>• 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".</li> <li>• A namespace (reference to an entry in namespace table). This is used when granting permission like "may unenroll this namespace".</li> <li>• 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".</li> <li>• 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.</li> </ul>
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](#)<sup>1</sup> is used for the requests and responses involved in obtaining a authorization assertions. [OpenSAML](#)<sup>2</sup>, 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

<sup>1</sup> <http://www.oasis-open.org>

<sup>2</sup> <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](#)<sup>1</sup>
- [Common API](#)<sup>2</sup>
- [Client API](#)<sup>3</sup>
- [Util API](#)<sup>4</sup>

---

<sup>1</sup> [http://www.globus.org/api/javadoc-4.2.1/globus\\_wsrf\\_cas\\_service\\_java/](http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_service_java/)

<sup>2</sup> [http://www.globus.org/api/javadoc-4.2.1/globus\\_wsrf\\_cas\\_common/](http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_common/)

<sup>3</sup> [http://www.globus.org/api/javadoc-4.2.1/globus\\_wsrf\\_cas\\_client\\_java/](http://www.globus.org/api/javadoc-4.2.1/globus_wsrf_cas_client_java/)

<sup>4</sup> [http://www.globus.org/api/javadoc-4.2.1/globus\\_wsrf\\_cas\\_utils\\_java/](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 \(SAML\)](#)<sup>1</sup> 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.

---

<sup>1</sup> [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=security](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](#)<sup>2</sup>
- [CAS schema file documentation](#)<sup>3</sup>

---

<sup>2</sup> [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](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)

<sup>3</sup> [../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 *policyFileName* Generate a CAS credential that includes only those permissions specified in file *policyFileName* (the default is to generate a credential with all the user's permissions). Details about the template of the file is provided [here](#).

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

-w *generatedCredentialFile* 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 &lt;type&gt;</code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
<code>-p, --protection &lt;type&gt;</code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
<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 <a href="#">here</a> .  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 &lt;value&gt;</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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <i>&lt;fqdn&gt;</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 &lt;file&gt;</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 &lt;file&gt;</code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime &lt;value&gt;</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 &lt;type&gt;</code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
<code>-p, --protection &lt;type&gt;</code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
<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 <a href="#">here</a> .  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 &lt;value&gt;</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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <code>&lt;fqdn&gt;</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 serviceName
```

## 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](#)<sup>1</sup>. 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.

---

<sup>1</sup> [http://www.globus.org/toolkit/docs/3.2/cas/CAS\\_policy\\_language\\_0.2.pdf](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 &lt;file&gt;</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 &lt;file&gt;</code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime &lt;value&gt;</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 &lt;type&gt;</code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"> <li>• <code>msg</code> for GSI Secure Message, or</li> <li>• <code>conv</code> for GSI Secure Conversation.</li> </ul>
<code>-p, --protection &lt;type&gt;</code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"> <li>• <code>sig</code> for signature, or</li> <li>• <code>enc</code> for encryption.</li> </ul>
<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 <a href="#">here</a> .

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 &lt;file&gt;</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 &lt;file&gt;</code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.
<code>-l, --contextLifetime &lt;value&gt;</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 &lt;type&gt;</code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
<code>-p, --protection &lt;type&gt;</code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
<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 <a href="#">here</a> .  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"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
-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 <a href="#">here</a> .

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"><li>• msg for GSI Secure Message, or</li><li>• conv for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• sig for signature, or</li><li>• enc for encryption.</li></ul>
-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 <a href="#">here</a> .  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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <i>&lt;fqdn&gt;</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~~

~~f~~

~~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:

- 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 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~~

• 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 &lt;file&gt;</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 &lt;file&gt;</code>	Specifies a client security descriptor. Overrides all other security settings.
<code>-help</code>	Prints the usage message for the client.

<code>-l, --contextLifetime &lt;value&gt;</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 &lt;type&gt;</code>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
<code>-p, --protection &lt;type&gt;</code>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
<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 <a href="#">here</a> .  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 &lt;value&gt;</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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <code>&lt;fqdn&gt;</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"><li>• msg for GSI Secure Message, or</li><li>• conv for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• sig for signature, or</li><li>• enc for encryption.</li></ul>
-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 <a href="#">here</a> .  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 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>&lt;file&gt;</i>	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime <i>&lt;value&gt;</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>&lt;type&gt;</i>	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>msg</code> for GSI Secure Message, or</li><li>• <code>conv</code> for GSI Secure Conversation.</li></ul>
-p, --protection <i>&lt;type&gt;</i>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• <code>sig</code> for signature, or</li><li>• <code>enc</code> for encryption.</li></ul>
-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 <a href="#">here</a> .  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>&lt;value&gt;</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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <i>&lt;fqdn&gt;</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

**te** Use one of the following to indicate the type of group for which you want a list of members:

- user
- object
- serviceType

**me** 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"><li>• msg for GSI Secure Message, or</li><li>• conv for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• sig for signature, or</li><li>• enc for encryption.</li></ul>

- `-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"><li>• msg for GSI Secure Message, or</li><li>• conv for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• sig for signature, or</li><li>• enc for encryption.</li></ul>
-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 <a href="#">here</a> .  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 <a href="#">here</a> .  If none of the above are set, host authorization is done by default and the expected server credential is <code>cas/&lt;fqdn&gt;</code> , where <code>&lt;fqdn&gt;</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"><li>• msg for GSI Secure Message, or</li><li>• conv for GSI Secure Conversation.</li></ul>
-p, --protection <type>	Specifies the protection level. <i>type</i> can be: <ul style="list-style-type: none"><li>• sig for signature, or</li><li>• enc for encryption.</li></ul>

- `-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.

---

# 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](#)<sup>1</sup> 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](#)<sup>2</sup> as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)<sup>3</sup>.

### 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](#)<sup>4</sup>. 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
```

---

<sup>1</sup> <http://jakarta.apache.org/commons/logging/>

<sup>2</sup> <http://logging.apache.org/log4j/>

<sup>3</sup> [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))

<sup>4</sup> <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> <b>Note</b></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](#)<sup>1</sup>
- [Using CAS to Manage Role-Based VO Sub-groups](#)<sup>2</sup>
- [The Community Authorization Service: Status and Future](#)<sup>3</sup>
- [Associated Standards](#)

---

<sup>1</sup> [http://www.globus.org/alliance/publications/papers/CAS\\_2002\\_Revised.pdf](http://www.globus.org/alliance/publications/papers/CAS_2002_Revised.pdf)

<sup>2</sup> <http://www.globus.org/alliance/publications/papers/CAS-group-CHEP03.pdf>

<sup>3</sup> [http://www.globus.org/alliance/publications/papers/CAS\\_update\\_CHEP\\_03-final.pdf](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.