

## **GT 4.2.0 CAS Public Interfaces**

DRAFT

## GT 4.2.0 CAS Public Interfaces

DRAFT

---

# Table of Contents

1. APIs .....	1
1. Programming Model Overview .....	1
2. Component API .....	1
2. Services and WSDL .....	2
1. Protocol overview .....	2
2. Operations .....	2
3. CAS Resource Properties .....	3
4. Faults .....	3
5. WSDL and Schema Definition .....	3
I. CAS Admin Commands .....	4
cas-proxy-init .....	5
cas-wrap .....	8
cas-enroll .....	11
cas-remove .....	15
cas-action .....	18
cas-group-admin .....	20
cas-group-add-entry .....	24
cas-group-remove-entry .....	27
cas-rights-admin .....	30
II. CAS Query Commands .....	?
cas-whoami .....	34
cas-list-object .....	36
cas-get-object .....	38
cas-group-list-entries .....	40
cas-find-policies .....	42
query-cas-service .....	44
3. Semantics and syntax of domain-specific interface .....	46
4. Configuring .....	48
1. Configuration overview .....	48
2. Loading the CAS service at start up .....	48
3. Configuring the VO Description .....	49
4. Configuring the maximum assertion lifetime .....	49
5. Configuring database backend .....	49
6. Configuring security descriptor .....	49
7. Configuring with a GridFTP Server .....	50
8. Configuring CAS to manage policy for web service. ....	50
9. CAS auto-registration with default WS MDS Index Service .....	50
10. Registering CAS manually with default WS MDS Index Service .....	52
5. Environment variable interface .....	53
1. Environmental variables for CAS .....	53
A. Errors .....	54
Glossary .....	58

## List of Tables

4.1. Database parameters .....	49
4.2. Mapping from web services object to CAS object .....	50
A.1. Java WS A&A Errors .....	55
A.2. WS A&A Authorization Framework Error Messages .....	57

DRAFT

# Chapter 1. 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-4.2.0/globus\\_wsrf\\_cas\\_service\\_java/](http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_cas_service_java/)

<sup>2</sup> [http://www.globus.org/api/javadoc-4.2.0/globus\\_wsrf\\_cas\\_common/](http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_cas_common/)

<sup>3</sup> [http://www.globus.org/api/javadoc-4.2.0/globus\\_wsrf\\_cas\\_client\\_java/](http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_cas_client_java/)

<sup>4</sup> [http://www.globus.org/api/javadoc-4.2.0/globus\\_wsrf\\_cas\\_utils\\_java/](http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_cas_utils_java/)

# Chapter 2. 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 a 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

DRAFT

# 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 *certificate* 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. *value* 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

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

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

## Requesting specific permissions from the CAS server

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

This file should contain one or more resource identifiers:

Resource: *ResourceNamespace* | *ResourceName*

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

*serviceType* *action*

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

Resource: FTPNamespace|ftp://sample1.org

file read

directory read

directory write

Resource: FTPNamespace|ftp://sample3.org

file read

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

uri:samlResourceWildcard

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

uri:samlActionNSWildcard uri:samlActionWildcard

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

Resource: uri:samlResourceWildcard

uri:samlActionNSWildcard uri:samlActionWildcard

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

Resource: FTPNamespace|ftp://sample1.org

uri:samlActionNSWildcard uri:samlActionWildcard

# Name

cas-wrap -- Runs program with CAS credentials

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

## Tool description

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

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

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

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

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

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

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

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

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

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

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

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

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

## Options

### Command-specific Options

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

*proxy-  
file*

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

### Common Options

#### Important

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

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

- `-c, --serverCertificate <file>` Specifies the server's *certificate* 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. *value* 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

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
```

DRAFT

# Name

cas-enroll -- Enroll a CAS Object

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

## Tool description

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

### Enrolling Trust Anchors

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

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

To enroll trust anchors:

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

where:

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

*nickname* Indicates the trust anchor nickname.

*authMethod* Indicates the authentication method used by the trust anchor.

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

### Enrolling Namespaces

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

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

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

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

To enroll namespaces:

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

where:

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

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

- `ExactComparison`: This class does a case-sensitive exact comparison of the object names. If `comparisonAlg` in the above method is set to `ExactComparison`, the class in the distribution is loaded and used.
- `WildcardComparison`: This class does wild card matching as described in [CAS Simple Policy Language](#)<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 (\).

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate < <i>file</i> >	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor < <i>file</i> >	Specifies a client security descriptor. Overrides all other security settings.
-help	Prints the usage message for the client.
-l, --contextLifetime < <i>value</i> >	Sets the lifetime of the client security context. <i>value</i> is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech < <i>type</i> >	Specifies the authentication mechanism. The value <i>type</i> can be: <ul style="list-style-type: none"> <li>• <code>msg</code> for GSI Secure Message, or</li> <li>• <code>conv</code> for GSI Secure Conversation.</li> </ul>
-p, --protection < <i>type</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/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>• 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 `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 [\[olink to other clients\]](#).

## 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](#).

DRAFT

# 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 [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:

- ~~o~~ • nickname if adding a trustAnchor or user
- groupName if adding a userGroup
- objectNamespace objectName if adding an object
- nickname if adding a namespace
- serviceName 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 serviceName act
```

where:

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

~~o~~

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

~~o~~

~~o~~

~~o~~

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

~~o~~

~~o~~

## Options

### Important

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

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

- 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 [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:

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

~~n~~

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

~~n~~

### Removing Member From An Object Group

To remove an object from an object group the user must have cas/remove\_group\_entry permission on that particular object group:

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

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

where:

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

~~n~~

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

~~t~~

- ~~t~~ • trustAnchor

~~t~~

- user

- userGroup

- object

- namespace

- `serviceType`

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

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

## Removing A Service/Action From A Service/Action Group

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

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

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

where:

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

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

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

## Options

### Important

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

- |   |  |
|---|--|
| <code>-a, --anonymous</code>                      | Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.    |
| <code>-c, --serverCertificate &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.   |

- `-l, --contextLifetime <value>` Sets the lifetime of the client security context. *value* 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-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.

DRAFT

# 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/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 <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.

DRAFT

# 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

**type** 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

**name** 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>• msg for GSI Secure Message, or</li><li>• conv 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>• 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 <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

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

- user
- object
- serviceType

**n** The name of the group.

## Common Options

### Important

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

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



### Note

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

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



### Note

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

## Chapter 3. 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 4. 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 4.1. 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 4.2. 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 5. 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.

# Appendix A. Errors

DRAFT

**Table A.1. Java WS A&A Errors**


Error Code	Definition	Possible Solutions
[JWSSEC-248] Secure container requires valid credentials	This error occurs when <code>globus-start-container</code> is run without any valid credentials. Either a proxy certificate or service/host certificate needs to be configured for the container to start up.	<ol style="list-style-type: none"> <li data-bbox="805 300 1334 604">1. If you are not looking to start up a container that uses GSI Secure Transport, which is used by the container by default, use <code>globus-start-container -nosec</code>. You will be able to use insecure clients and services. However, this also implies that if you have not configured individual services with credentials, you will not be able to securely access the service.</li> <li data-bbox="805 615 1334 804">2. If you are running a personal container, generate a proxy certificate with <code>grid-proxy-init</code>. If the proxy certificate is not in the default location, configure the container security descriptor as described in <a href="#">Configuring Container Security Descriptor</a>.</li> <li data-bbox="805 835 1334 930">3. If you want to use host certificates, configure the container security descriptor as described <a href="#">Configuring Credentials</a>.</li> </ol>
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-250] Failed to load certificate/key file]	This error occurs if the file path to the container certificate and key configured are invalid.	<ol style="list-style-type: none"> <li data-bbox="805 963 1334 1163">1. The path to the container certificate and key are configured in <code>\$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml</code>. This file is loaded as described [here - fixme link]. Ensure that the path is correct.</li> </ol>
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-249] Failed to load proxy file]	This error occurs if container proxy file configured is invalid.	<ol style="list-style-type: none"> <li data-bbox="805 1194 1334 1394">1. The path to the container proxy certificates are configured in <code>\$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml</code>. This file is loaded as described [here - fixme link]. Ensure that the path is correct.</li> </ol>
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-245] Error parsing file: "etc/globus_wsrf_core/global_security_descriptor.xml" [Caused by: ...]	This error occurs if the container security descriptor configured is invalid.	<ol style="list-style-type: none"> <li data-bbox="805 1425 1334 1520">1. The container security descriptor should conform to the <a href="#">Container Security Descriptor Schema</a>.<sup>1</sup></li> <li data-bbox="805 1551 1334 1604">2. Refer to the "Caused by: " section for details on the specific element that is not correct.</li> </ol>

<sup>1</sup> [http://www.globus.org/toolkit/docs/4.2.0/security/container\\_security\\_descriptor.xsd](http://www.globus.org/toolkit/docs/4.2.0/security/container_security_descriptor.xsd)

Error Code	Definition	Possible Solutions
[JGLOBUS-77] Unknown CA	This error occurs if the CA certificate for the credentials being used is not installed correctly.	<ol style="list-style-type: none"><li data-bbox="805 243 1325 401">1. If this issue occurs on the server side, the container is not configured with CA certificates. The container looks for trusted certificates in the default location as described <a href="#">Java CoG Toolkit FAQ</a><sup>2</sup></li><li data-bbox="805 428 1325 520">2. On the server side, the trusted certificates can be configured as described in <a href="#">Trusted Certificates</a></li><li data-bbox="805 548 1325 640">3. On the client side, trusted certificates can be configured as described in <a href="#">Configuring Trusted Credentials</a></li></ol>

<sup>2</sup> <http://www.globus.org/cog/distribution/1.2/FAQ.TXT>

**Table A.2. WS A&A Authorization Framework Error Messages**

Error Code	Definition
org.globus.cas.impl.databaseAccess.CasDBException, connection refused	<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>
CAS clients fail with org.globus.cas.impl.databaseAccess.CasDBException	<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>

# Glossary

## C

certificate                      A public key plus information about the certificate owner bound together by the digital signature of a CA. In the case of a CA certificate, the certificate is self signed, i.e. it was signed using its own private key.

## T

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

DRAFT