CAS Simple Policy Language

Document version 0.2

Laura Pearlman, Von Welch

laura@isi.edu, welch@mcs.anl.gov

1 Introduction

This document describes the simple policy language that will be used for the initial implementation of the Community Authorization Service (CAS).

2 Language Requirements

The requirements of the language:

1. It must be able to describe files and file actions appropriate for a GridFTP server.
2. It must be simple to create and parse.

3 Language Format

The policy language allows specification of a policy containing one or more Rights. A policy is encoded as a printable ASCII string terminated by a NUL (ASCII code 0 character). Each line of the file contains either a delimiter or an attribute-value pair. All whitespace (except carriage returns) is ignored.

Each Right is delimited by curly brackets (‘{‘ and ‘}’) and contains one or more Objects on which the Rights apply and one or more Actions which are granted on the Object(s).

Each collection of Objects in a Right is identified by an Object Type. Currently this is always “File” indicating they describe Files (either an individual file or directory tree).

Each collection of Actions contains a Service Type. Currently this is always “File” indicating these are normal Unix file type actions.

The formal language description in EBNF format is:

```
Policy ::= Right+

Right ::= ``{\n" Objects Services ``}\n"

Objects ::= ObjectNameType ObjectName+

Services ::= ServiceType ServiceAction+

ObjectNameType ::= ``OBJECT_NAME_TYPE=" Value ``\n"

ObjectName ::= ``OBJECT_NAME=" Value ``\n'
```

ServiceType ::= ‘‘SERVICE_TYPE=” Value ‘‘\n”

ServiceAction ::= ‘‘SERVICE_ACTION=” Value ‘‘\n”

Value ::= ‘‘[^\n]+”

Things to note about the format:

1. One currently can’t have a value ending with whitespace.

4 Values

4.1 Object Name Types

For the purposes of CAS AlphaR1 the only Object Name Type will be "wildcard".

4.2 Object Names

For the purposed of CAS AlphaR1 the object names for the "wildcard" object name type will follow these rules:

1. They will represent Unix files, Unix directories, or subtrees in the Unix filesystem.
2. All names will begin at the root of the filesystem (i.e. will start with a slash).
3. Names may not include ".." as a component referring to the parent directory.
4. Subtrees will end in "/*" which indicates all file and directories under the listed path.
5. The standard Unix file path delimiter, forward slash (/), will be used to delimit directory components.

4.3 Service Types

For the purposes of CAS AlphaR1 the only supported service type will be "file".

4.4 Service Actions

For the purposes of CAS AlphaR1 the following services will be supported for the "file" service type:

Action	Rights given by action
read	For a File object, gives permission to read the file. For a Directory object, gives permission to chdir to the directory
lookup	For a File, gives the right to get Unix stat() information. For a directory, gives the right to chdir to and to list the contents of the directory
write	For a File, allows modification of an existing file. For a directory, gives the right to chdir to the directory.
create	For a File, allows creation of the file if it does not exist. For a directory, allows creation of the directory if it does not exist and gives the

	right to chdir to the directory if it does exist
delete	For a File, allows deletion of the file. For a Directory, allows deletion of the directory, if empty; also gives the right to chdir to the directory.
chdir	For a File, this right is meaningless. For a Directory, allows making the directory the current default directory.

5 Supported FTP Command and needed rights

FTP Client Comand	**FTP Protocol Command**	**Needed Rights**
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 / RNTO	*read* and *delete* on old file *write* on new file, if it exists *create* on new file, if it does not exist

6 Examples

The following policy contains four entries granting rights to files on the gridftp server on *myserver.edu*. The first gives read and lookup rights to the file "/etc/gridmap". The second gives lookup, read and write access to the files "/tmp/foo" and "/tmp/bar". The third give read, write, and lookup access to the subtree "/home/user". The fourth gives chdir access to the whole directory tree,

```
{
OBJECT_NAME_TYPE=wildcard
OBJECT_NAME=ftp://myserver.edu/etc/grid-security/gridmap
SERVICE_TYPE=file
SERVICE_ACTION=read
SERVICE_ACTION=lookup
}
{
OBJECT_NAME_TYPE=wildcard
OBJECT_NAME=ftp://myserver.edu/tmp/foo
OBJECT_NAME=ftp://myserver.edu/tmp/bar
SERVICE_TYPE=file
SERVICE_ACTION=read
SERVICE_ACTION=write
SERVICE_ACTION=lookup
}
{
OBJECT_NAME_TYPE=wildcard
OBJECT_NAME=ftp://myserver.edu/home/user/*
SERVICE_TYPE=file
SERVICE_ACTION=read
SERVICE_ACTION=lookup
}
{
OBJECT_NAME_TYPE=wildcard
OBJECT_NAME=ftp://myserver.edu/*
SERVICE_TYPE=file
SERVICE_ACTION=chdir
}
```

With these rights the user would be able to perform the following actions:

- get or ls /etc/grid-security/gridmap
- get or ls /tmp/foo
- put /tmp/foo if it already exists
- get or ls /tmp/bar
- put /tmp/bar if it already exists

- list the contents of any directory under /home/user
- get any file under /home/user
- cd to any directory

7 See Also

The FTP RFC: http://www.cis.ohio-state.edu/Services/rfc/rfc-text/rfc0959.txt

The CAS Webpage: http://www.globus.org/security/CAS/