# GridFTP w/ Java Authorization Developer Documentation

## Introduction

For a general overview of the GridFTP w/ Java Authorization extension to Globus GridFTP server, please refer to the User Documentation.

The Java Authorization API builds on the work done by the Community Authorization Service. A short overview of the GridFTP request system follows.

When a GridFTP client (hereby called a “requester”) first connects to a secure GridFTP server, authentication is performed. This involves taking the requester’s X.509 certificate and verifying a) the certificate is valid, and b) the CA that issued the certificate is trusted by the GridFTP server. For more information on X.509 certificates, please refer to one of the many security documents on the subject.

The C authz implementation that is part of the Java Authorization package works with Globus’ standard authentication to log the user in. As part of the login process, the C authz implementation maps the incoming user to the same user that started the GridFTP server control process. For example, if user “gridftp” started the GridFTP server, then all incoming users are mapped to user “gridftp”. Thus, it is critical that for a secure GridFTP server setup, you use a user with a minimum of privileges on the system. Ideally, this would be a new user specifically created for ftp access.

After the incoming requester is mapped to the user, a data process is created that handles the requester’s commands. If you are familiar with regular ftp, they are essentially the same commands, since GridFTP is at its core a standard FTP server with grid extensions. See the following table for a list of acceptable requester commands.

Table of requester operations

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

A request in both the Java Authorization API and in CAS consists of a tuple: the requester identity, the requester operation (one of the above actions), and the requester target, always a URI with the ftp protocol that points to a resource on the local GridFTP filesystem.

The C authz implementation provided as part of the Java Authorization extension creates a new object (as specified in the configuration) that implements the Authorization interface and calls the authorize method on the interface when a GridFTP request is made. The Java Authorization API allows a Java developer to implement authorization using information in the request tuple.

## Java Authorization API

The API is extremely simple, consisting of the following interface:

**public** **interface** Authorize {

**public** **boolean** authorize(String identity, String operation, String target);

}

To implement Java Authorization for GridFTP, simply write a Java implementation using the above interface. The authorize method takes three arguments: requester identity, requester operation, and requester target. The authorize method returns true if the request is allowed, and false if the request is denied.

To make it easy for a developer to implement a new authorization policy, there is a convenience class that one can extend and implement. The interface is as follows:

**public** **abstract** **class** AbstractAuthCallout **implements** Authorize {

…

**public** **abstract** **boolean** authorizeOperation(String identity, GridFTPOperation.Operation operation, String target);

}

To implement the code, simply extend AbstractAuthCallout and implement authorizeOperation. Note that the operation parameter is of type GridFTPOperation.Operation, which is an enum that defines the allowed operations for improved type safety (same as the operations in the above table). In order to use the new class, modify $GRIDFTP\_JAVA\_AUTH\_CONF with the new classpath and classname of your class. The only other requirement the class needs to fulfill is that it needs a default constructor. For an example, see org.cagrid.authorization.callout.gridftp.db.DatabaseAuthCallout.

For convenience, the setup package provides two working authorization checks. One check is a database authorization check. Tuples are stored in the database. The implementation verifies that the given tuple representing the request exists in the database. It allows the request if the tuple exists, and denies the request if it doesn’t exist.

The other check uses Grid Grouper to perform authorization. The implementation takes a grid grouper configuration, such as a simple membership query, and performs the configured check for the supplied requester identity.

Please see the attached Javadoc for API details (TODO link).

# Works Cited

1. How To Set Up CAS to Use with GridFTP. *Globus Toolkit.* [Online] http://www.globus.org/toolkit/docs/4.0/security/cas/WS\_AA\_CAS\_HOWTO\_Setup\_GridFTP.html.