iRODS GridFTP Module

Documentation

SPONSORED BY THE

Federal Ministry
of Education
and Research

Content

Author: Christian Loeschen, TU Dresden

Dresden, 07/15/2010

1 Introduction

This is the documentation of the GridFTP module, developed for the iRODS Data-management System. Using this module it is possible to access files on GridFTP servers, transfer them from/to iRODS and manage files/directories on GridFTP servers. GSI proxy certificates or every iRODS user are supported, or the iRODS server can be run using a single certificate. The GridFTP microservices can be used with the `irule` command or it is possible to automate their usage within iRODS actions.

It is developed within the German DGrid Integrationsprojekt 2 (DGI2) founded by the German Federal Ministry of Education and Research (BMBF).

2 Installation

1. Copy the GridFTP module to `<irods_dir>/modules`.
2. Ensure in `info.txt` Enabled is set to: `yes`.
3. The Globus GridFTP and GSI Libraries must be available on your system.
4. Adjust the include flags and library paths in Makefile, if necessary.
5. Run `irodssetup`.
6. Environment variable `LD_LIBRARY_PATH` must be set to Globus Toolkit libraries (`<globus_dir>/lib`) to make iRODS starting properly.

3 Usage

GSI certificate information, esp. CRLs in `/etc/grid-security` must be available on iRODS host and up to date, that certificate validation works.

3.1 Available Microservices

`msiGridftpChmod(*MODE,*DST)*`	Changes permissions for a file. Unix like permissions are used (600, 755, 644).
`msiGridftpCksm(*SRC,*OUT)*`	Returns the checksum of a file.
`msiGridftpCp(*SRC,*DST)`	Copies a file on the server from SRC to DST
`msiGridftpDel(*DST)`	Deletes a file.
`msiGridftpExist(*DST,*OUT)`	Checks whether a file exists.
`msiGridftpFuncTest(*SRC,*DST,*RES)`	Test microservice that tests all the GridFTP module functions.
`msiGridftpGet(*SRC,*DST)`	Gets a file from a GridFTP server to iRODS.
`msiGridftpLs(*SRC,*OUT)`	Returns a list of the directory entries.
`msiGridftpMkdir(*DST)`	Creates a directory.
`msiGridftpModtime(*DST,*OUT)`	Returns the last modification time of a file.

`msiGridftpMv(*SRC,*DST)*`	Moves a file on the server from SRC to DST
`msiGridftpPut(*SRC,*DST)`	Puts an iRODS file to a GridFTP server.
`msiGridftpRmdir(*DST)`	Deletes a directory.
`msiGridftpSize(*SRC,*OUT)`	Returns the size of a file.
`msiGridftpTest()`	A test microservice to verify the module is integrated correctly.

*see 3.5.2.2

3.2 Using Proxy Certificates

The microservices make use of user proxy certificates to authenticate the user on GridFTP servers. Therefore the user needs to upload a valid proxy certificate to iRODS:

```
~> grid-proxy-init
Your identity: /C=DE/O=GridGermanyOU=Technische Universitaet
Dresden/OU=ZIH/CN=Christian Loeschen
Enter GRID pass phrase for this identity:
Creating proxy ........................................... Done
Your proxy is valid until: Wed May  5 02:58:46 2010
~> iput /tmp/x509up_u1000 user-proxy
~> ils
/tempZone/home/rods:
      user-proxy
```

The proxy certificate can be created using `voms-proxy-init`, too. The certificate must reside within the iRODS home directory of the user and be named "user-proxy". It can be renamed or a path could be added in the `proxy_file` variable within `gridftp.h` header file. The module must be compiled after that to take affect.

The GridFTP module is also able to work with a single server certificate. Therefore the iRODS server must be started with the userid of a valid certificate.

3.3 Common Usage Scenarios

Some common scenarios how to use GridFTP microservices with the irule command, combined within Rules are described within this section.

This one puts a single file from the iRODS user directory to a GridFTP server:

```
~> irule -v "msiGridftpPut(*SRC,*DST)"
"*SRC=/tempZone/home/rods/foo.bar%*DST=gsiftp://helena.zih.tu-
dresden.de/pnfs/zih.tu-dresden.de/data/ghep/foo.bar" ""
```

This one changes the access rights for a file:

```
~> irule -v "msiGridftpChmod(*MODE,*DST)"
"*MODE=644%*DST=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar" ""
```

But now let's do some more advanced stuff. In the next exampled, a directory is listed, and the output is processed in following microservices:

```
~> irule -v "msiGridftpLs(*DIR,
*FILES)##forEachExec(*FILES,writeLine(stdout,*FILES),nop)|nop"
"*DIR=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/" "ruleExecOut"

~> irule -v "msiGridftpLs(*DIR,
*FILES)##writeLine(stdout,)##forEachExec(*FILES,msiGridftpSize(*FILES,
*SIZE)##msiGridftpModtime(*FILES,*TIME)##writeLine(stdout,*TIME  *SIZE
*FILES),nop)|nop" "*DIR=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/" "ruleExecOut"

~> irule -v "msiGridftpLs(*DIR,
*FILES)##forEachExec(*FILES,msiGridftpDel(*FILES),nop)|nop"
"*DIR=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/" ""

~> irule -v "msiGridftpLs(*DIR,
*FILES)##forEachExec(*FILES,msiGridftpGet(*FILES,*DEST),nop)|nop"
"*DIR=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/%*DEST=/tempZone/home/rods/" ""
```

This is a simple check whether a file exists with some output:

```
irule -v "msiGridftpExist(*DST,*OUT)##ifExec(*OUT ==
YES,writeLine(stdout,*DST exists),nop,writeLine(stdout,*DST not
exists),nop)|nop" "*DST=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar" "ruleExecOut"
```

It is also possible to add some actions in `<irods_dir>/server/config/reConfigs/core.irb`:

```
acGridftpMv(*SRC,*DST)||msiGridftpCp(*SRC,*DST)##msiGridftpDel(*SRC)|
nop
```

So one can use gridftp move, though msiGridftpMv won't work, as on my test machine:

```
~> irule -v "acGridftpMv(*SRC,*DST)" "*SRC=gsiftp://helena.zih.tu-
dresden.de/pnfs/zih.tu-dresden.de/data/ghep/foo
%*DST=gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar" ""
```

3.4 Performing a module test

`msiGridftpTest` is just a test microservice, that indicates whether the GridFTP module is integrated correctly into iRODS. When it's output look like this, the GridFTP module is available:

```
~> irule --test "msiGridftpTest" *A=0 *B
Level 0: DEBUG:     msiGridftpTest log
```

With the `msiGridftpFuncTest` microservice a full test is run testing all GridFTP module functions. Two input parameters are necessary:

– *SRC: an existing iRODS file to copy to the GridFTP server for testing

– *DST: an existing directory on the GridFTP server to write the test to

```
~> irule -v "msiGridftpFuncTest(*SRC, *DST, *RES)"
"*SRC=/tempZone/home/rods/foo.bar%*DST=gsiftp://helena.zih.tu-
dresden.de/pnfs/zih.tu-dresden.de/data/ghep" "*RES"
rcExecMyRule: msiGridftpFuncTest(*SRC, *DST, *RES)
outParamDesc: *RES
ExecMyRule completed successfully.    Output

  output index: 0
    label: *RES
    type: STR_PI
    str content:
gridftp mkdir: ok
gridftp put: ok
gridftp cp: ok
gridftp ls: ok
gridftp exist: ok
gridftp size: ok
gridftp modtime: ok
gridftp cksm: FAILED
gridftp chmod: FAILED
gridftp get: ok
gridftp mv: FAILED
gridftp del: ok
gridftp rmdir: ok
```

As one can see, my GridFTP test server doesn't support checksum, chmod and move functions.

3.5 Additional Tips

3.5.1 How to enable a user to verify he is using a valid proxy certificate

For this purpose it is necessary to put a link from `grid-proxy-info` into `<irods_dir>/server/bin/cmd/`. `grid-proxy-info` comes with Globus Toolkit and is usually located in `<globus_dir>/bin/`. Now the user needs to know the real path on the file-system to his proxy certificate. This is for example `<irods_dir>/Vault/home/<username>/user-proxy`. Using `iexecmd` the user can verify his proxy certificate:

```
~> iexecmd "grid-proxy-info -f /opt/iRODS/Vault/home/rods/user-proxy"
subject  : /C=DE/O=GridGermany/OU=Technische Universitaet
Dresden/OU=ZIH/CN=Christian Loeschen/CN=203199236
issuer   : /C=DE/O=GridGermany/OU=Technische Universitaet
Dresden/OU=ZIH/CN=Christian Loeschen
identity : /C=DE/O=GridGermany/OU=Technische Universitaet
Dresden/OU=ZIH/CN=Christian Loeschen
type     : RFC 3820 compliant impersonation proxy
strength : 512 bits
path     : /opt/iRODS/Vault/home/rods/user-proxy
timeleft : 0:00:00
```

As one can see, my proxy certificate is expired.

Alternatively a small script can be put in `<irods_dir>/server/bin/cmd`:

```
~> cat /opt/iRODS/server/bin/cmd/test-proxy

#!/bin/sh
/opt/globus/bin/grid-proxy-info -f /opt/iRODS/Vault$1
```

Now a user can issue `iexecmd` with the irods path to his certificate, but without the zone prefix:

```
~> iexecmd "test-proxy /home/rods/user-proxy"
```

3.5.2 Possible adjustments in `gridftp.h` file

To take effect of changes of one of the following options the GridFTP module must be recompiled.

3.5.2.1 Enable debugging

Defining `GRIDFTP_DEBUG` makes the GridFTP microservices more verbose. Then in `<rods_dir>/server/log/rodsLog*` additional information about microservice execution can be found.

3.5.2.2 Disable feature check

Because some GridFTP server (like my dCache test machine) don't understand every GridFTP command, some of them won't work. Before executing a command, the GridFTP server is asked whether the command is supported. Because there seem to be different feature mappings within different libraries, this function may return wrong values. When there occur any problems, define `GRIDFTP_DISABLE_FEATURE_CHECK` to prevent the feature check. Then the feature check always returns true.

3.5.2.3 Set user proxy file location

`char *proxy_file` points to the user proxy file for GSI authentication within the iRODS user home directory. It's default value is "/user-proxy". You can change this, and also add additional paths, but don't forget the leading slash.

Pay attention to apply changes also to 3.5.1.

3.5.2.4 Build a GridFTP command line client

As a fall-out of the development of the GridFTP module, a simple command line GridFTP client is arised. If `GRIDFTP_IRODS_MODULE` is defined, the client is build. In the `console_client/` directory the makefile for building the client is located. Change to this directory and issue a `make`. Therefore, no iRODS server is necessary. After building the binary is in this directory, too. To see a brief introduction about its functions you can simply type

```
~> gridftp-client
usage: gridftp
        get|put|cp|mv <src-url> <dst-url>
        ls|del|rmdir|mkdir|exist|size|cksm|modtime|feat <url>
        chmod <mode> <url>
```

Anything else works as usual:

```
~> gridftp-client put /bin/sh gsiftp://helena.zih.tu-
dresden.de/pnfs/zih.tu-dresden.de/data/ghep/foo.bar
~> gridftp-client ls gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/
foo.bar
~> gridftp-client size gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar
606864
~> gridftp-client modtime gsiftp://helena.zih.tu-
dresden.de/pnfs/zih.tu-dresden.de/data/ghep/foo.bar
Wed Jul  7 17:45:09 2010
~> gridftp-client del gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar
~> gridftp-client exist gsiftp://helena.zih.tu-dresden.de/pnfs/zih.tu-
dresden.de/data/ghep/foo.bar
file does not exist
```

4 Further Information

4.1 License

This software is released under BSD license.

4.2 Contact

You can contact the author and developer via email: *christian.loeschen@tu-dresden.de*

4.3 Links

IRODS	www.irods.org/
Globus Toolkit	www.globus.org/toolkit/
DGI2 homepage	http://dgi.d-grid.de/index.php?id=456&L=1
Project homepage	http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/grid_computing/iRODS

Appendix A: Files

```
info.txt                      iRODS module information file
Makefile                      module makefile
README                        information about GridFTP module

microservices/include/
    gridftp.h                 module header file
    gridftpMS.h               header file with all microservice
                              functions
    microservices.header      microservice table header for iRODS
    microservices.table       microservice table for iRODS

microservices/obj/

microservices/src/
    gridftp.c                 main file for GridFTP console client
    gridftp_cp.c              microservice source files
    gridftp_feat.c
    gridftp_modtime.c
    gridftp_rmdir.c
    gridftp_chmod.c
    gridftp_del.c
    gridftp_functional_test.c
    gridftp_ls.c
    gridftp_mv.c
    gridftp_size.c
    gridftp_cksm.c
    gridftp_exist.c
    gridftp_get.c
    gridftp_mkdir.c
    gridftp_put.c
    gridftp_tools.c           module helper functions

console_client/
    makefile                  makefile for the GridFTP console client
```

Appendix B: API Functions

I Microservice functions (gridftpMS.h)

```
/*
 *    iRODS Microservice function that changes the access permissions for
 *    a file or directory on a GridFTP server.
 *
 *    @param mode  msParam_t containing changed access rights in UNIX style (like "644")
 *    @param url   msParam_t containing GridFTP URL
 *    @param rei   iRODS ruleExecutionInfo structure
 *    @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpChmod(msParam_t *mode, msParam_t *url, ruleExecInfo_t *rei);


/*
 *    iRODS Microservice function that requests the checksum of a file on
 *    a GridFTP server.
 *
 *    @param url   msParam_t containing GridFTP file URL
 *    @param out   msParam_t to write requested checksum to
 *    @param rei   iRODS ruleExecutionInfo structure
 *    @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpCksm(msParam_t *url, msParam_t *out, ruleExecInfo_t *rei);


/*
 *    iRODS Microservice function that copies a file from one GridFTP
 *    location to another.
 *
 *    @param url   msParam_t containing GridFTP file URL to copy file from
 *    @param out   msParam_t containing GridFTP file URL to copy file to
 *    @param rei   iRODS ruleExecutionInfo structure
 *    @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpCp(msParam_t *srcUrl, msParam_t *dstUrl, ruleExecInfo_t *rei);


/*
 *    iRODS Microservice function that deletes a file from a GridFTP server.
 *
 *    @param url   msParam_t containing GridFTP file URL
 *    @param rei   iRODS ruleExecutionInfo structure
 *    @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpDel(msParam_t *url, ruleExecInfo_t *rei);


/*
 *    iRODS Microservice function that requests the existence of a file
 *    on a GridFTP server.
 *
 *    @param url   msParam_t containing GridFTP URL
 *    @param out   msParam_t to write result to
 *    @param rei   iRODS ruleExecutionInfo structure
 *    @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpExist(msParam_t *url, msParam_t *out, ruleExecInfo_t *rei);
```

```
/*
 *     iRODS Microservice function that performs a full module test on a
 *     GridFTP server.
 *
 *     @param rodsFile     msParam_t containing iRODS file name to use for testing
 *     @param testUrl      msParam_t containing GridFTP directory URL to write test
 *     @param out          msParam_t to write test result to
 *     @param rei          iRODS ruleExecutionInfo structure
 *     @return             1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpFuncTest(msParam_t *rodsFile, msParam_t *testUrl, msParam_t *out,
ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that gets a file from a GridFTP server
 *     to local iRODS.
 *
 *     @param srcUrl       msParam_t containing GridFTP file URL to get file from
 *     @param rodsFile     msParam_t containing iRODS file name to put file to
 *     @param rei          iRODS ruleExecutionInfo structure
 *     @return             1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpGet(msParam_t *srcUrl, msParam_t *rodsFile, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that does a directory listing on a
 *     GridFTP server.
 *
 *     @param url   msParam_t containing GridFTP directory URL
 *     @param out   msParam_t to write requested directory listing to
 *     @param rei   iRODS ruleExecutionInfo structure
 *     @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpLs(msParam_t *url, msParam_t *out, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that creates a directory on a GridFTP server.
 *
 *     @param url   msParam_t containing GridFTP directory URL
 *     @param rei   iRODS ruleExecutionInfo structure
 *     @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpMkdir(msParam_t *url, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that requests the modification date of a
 *     file on a GridFTP server.
 *
 *     @param url   msParam_t containing GridFTP file URL
 *     @param out   msParam_t to write requested modification time to
 *     @param rei   iRODS ruleExecutionInfo structure
 *     @return      1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpModtime(msParam_t *url, msParam_t *out, ruleExecInfo_t *rei);
```

```
/*
 *     iRODS Microservice function that moves a file from one GridFTP location
 *     to another.
 *
 *     @param url    msParam_t containing GridFTP file URL to move file from
 *     @param out    msParam_t containing GridFTP file URL to move file to
 *     @param rei    iRODS ruleExecutionInfo structure
 *     @return       1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpMv(msParam_t *srcUrl, msParam_t *dstUrl, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that puts a file from local iRODS to a
 *     GridFTP server.
 *
 *     @param rodsFile      msParam_t containing existing iRODS file name
 *     @param dstUrl        msParam_t containing GridFTP file URL to put file to
 *     @param rei           iRODS ruleExecutionInfo structure
 *     @return              1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpPut(msParam_t *rodsFile, msParam_t *dstUrl, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that deletes a directory on a GridFTP server.
 *
 *     @param url    msParam_t containing GridFTP directory URL
 *     @param rei    iRODS ruleExecutionInfo structure
 *     @return       1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpRmdir(msParam_t *url, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that requests the size of a file on a
 *     GridFTP server.
 *
 *     @param url    msParam_t containing GridFTP file URL
 *     @param out    msParam_t to write requested size to
 *     @param rei    iRODS ruleExecutionInfo structure
 *     @return       1 on success, on error -344000 (EXEC_CMD_ERROR)
 */
int msiGridftpSize(msParam_t *url, msParam_t *out, ruleExecInfo_t *rei);


/*
 *     iRODS Microservice function that checks the correct module integration.
 *     Does only a little ouput.
 *
 *     @param rei    iRODS ruleExecutionInfo structure
 *     @return       1
 */
int msiGridftpTest(ruleExecInfo_t *rei);
```

II Internal module functions (gridftp.h)

```
/*******************************
 *    main gridftp functions
 ******************************/


/*
 *    Changes the access permissions for a file or directory on a GridFTP server.
 *
 *    @param mode         UNIX like permission set (like "644")
 *    @param gridftpURL   GridFTP URL for this operation
 *    @return             0 on success, 1 on failure
 */
int gridftp_chmod(char *mode, char *gridftpURL);


/*
 *    Requests the checksum of a file on a GridFTP server.
 *
 *    @param gridftpURL   GridFTP URL to request file checksum
 *    @return             0 on success, 1 on failure
 */
int gridftp_cksm(char *gridftpURL);


/*
 *    Copies a file from one GridFTP location to another.
 *    GridFTP location.
 *
 *    @param srcGridftpURL      source GridFTP URL to copy from
 *    @param dstGridftpURL      destination GridFTP URL to copy to
 *    @return                   0 on success, 1 on failure
 */
int gridftp_cp(char *srcGridftpURL, char *dstGridftpURL);


/*
 *    Deletes a file from a GridFTP server.
 *
 *    @param gridftpURL   GridFTP URL for file to delete
 *    @return             0 on success, 1 on failure
 */
int gridftp_del(char *gridftpURL);


/*
 *    Requests the existence of a file on a GridFTP server.
 *
 *    @param gridftpURL   GridFTP URL to request file existence
 *    @return             0 on success, 1 on failure
 */
int gridftp_exist(char *gridftpURL);


/*
 *    Requests the supported features of a GridFTP server.
 *
 *    @param gridftpURL   GridFTP URL to request supported features
 *    @return             0 on success, 1 on failure
 */
int gridftp_feat(char *gridftpURL);
```

```
/*
 *     Gets a file from a GridFTP server to local (iRODS).
 *     This get function is used within the commandline client
 *     and the iRODS GridFTP file driver.
 *
 *     @param srcGridftpURL      source GridFTP URL to get file
 *     @param dstFile            destination file name
 *                               (with real path on the filesystem)
 *     @return                   0 on success, 1 on failure
 */
int gridftp_get(char *srcGridftpURL, char *dstFile);


/*
 *     Gets a file from a GridFTP server to local iRODS.
 *
 *     @param srcGridftpURL      source GridFTP URL to get file
 *     @param dstIrodsFile       destination iRODS file name
 *     @return                   0 on success, 1 on failure
 */
int gridftp_module_get(char *srcGridftpURL, char *dstIrodsFile);


/*
 *     Does a directory listing on a GridFTP server.
 *
 *     @param gridftpURL   GridFTP URL for directory listing
 *     @return             0 on success, 1 on failure
 */
int gridftp_ls(char *gridftpURL);


/*
 *     Creates a directory on a GridFTP server.
 *
 *     @param gridftpURL   GridFTP URL of directory to create
 *     @return             0 on success, 1 on failure
 */
int gridftp_mkdir(char *gridftpURL);


/*
 *     Requests the modification date of a file on a GridFTP server.
 *
 *     @param gridftpURL   GridFTP URL to request file modification date
 *     @return             0 on success, 1 on failure
 */
int gridftp_modtime(char *gridftpURL);


/*
 *     Moves a file from one GridFTP location to another.
 *
 *     @param srcGridftpURL      source GridFTP URL to move from
 *     @param dstGridftpURL      destination GridFTP URL to move to
 *     @return                   0 on success, 1 on failure
 */
int gridftp_mv(char *srcGridftpURL, char *dstGridftpURL);
```

```
/*
 *     Puts a file from local (iRODS) to a GridFTP server.
 *     This put function is used within the commandline client
 *     and the iRODS GridFTP file driver.
 *
 *     @param srcFile            source file
 *                               (with real path on the filesystem)
 *     @param dstGridftpURL      destination GridFTP URL to put this file
 *     @return                   0 on success, 1 on failure
 */
int gridftp_put(char *srcFile, char *dstGridftpURL);


/*
 *     Puts a file from local iRODS to a GridFTP server. This one is used within
 *     This put function is used within iRODS microservice.
 *
 *     @param srcIrodsFile       source iRODS file
 *     @param dstGridftpURL      destination GridFTP URL to put this file
 *     @return                   0 on success, 1 on failure
 */
int gridftp_module_put(char *srcIrodsFile, char *dstGridftpURL);


/*
 *     Deletes a directory on a GridFTP server.
 *
 *     @param gridftpURL   GridFTP URL of directory to remove
 *     @return             0 on success, 1 on failure
 */
int gridftp_rmdir(char *gridftpURL);


/*
 *     Requests the size of a file on a GridFTP server.
 *
 *     @param gridftpURL   GridFTP URL to request file size
 *     @return             0 on success, 1 on failure
 */
int gridftp_size(char *gridftpURL);



/***************************
 *     helper functions
 ***************************/


/*
 *     Sets the user proxy certificate for use within microservices.
 *
 *     @param rsCommiRODS  server communication handle
 *     @return             0 on success, 1 on failure
 */
int setUserProxy(rsComm_t *rsComm);


/*
 *     Prints an informational message to stdout/rodsLog, if
 *     Prints GRIDFTP_DEBUG is defined.
 *
 *     @param msg          message to print
 */
void printInfo(char *msg, ...);
```

```c
/*
 *     Prints an error message to stdout/rodsLog.
 *
 *     @param msg          error message to print
 */
void printErrorMessage(char *msg, ...);


/*
 *     Prints the messages from an error object to stdout/rodsLog.
 *
 *     @param errObject    error message to print
 */
void printErrorObject(globus_object_t *errObject);


/*
 *     Checks the error state of a globus_result_t object returned by
 *     a globus_ftp_client operation. When it contains an error, the
 *     error message is printed to stdout/rodsLog.
 *
 *     @param result       globus_result_t object, that was returned
 *                         by a globus_ftp_client operation
 *     @return             1, if the errObject contains an error, otherwise 0
 */
inline int checkResult(globus_result_t result);


/*
 *     Requests whether a single feature is supported by a GridFTP server.
 *
 *     @param handle       an initialized handle
 *     @param url          GridFTP URL of the requested server
 *     @param feature      feature to request
 */
globus_ftp_client_tristate_t gridftp_feat_check(
             globus_ftp_client_handle_t *handle,
             char *url,
             globus_ftp_client_probed_feature_t feature);
```