# UNICORE UFTP SERVER

UNICORE Team

| | |
|---|---|
| Document Version: | 1.2.0 |
| Component Version: | 1.2.0 |
| Date: | 15 12 2011 |

# Contents

This is the UFTP user manual providing information on running and using the UNICORE UFTP server *uftpd*. Please note also the following places for getting more information:

UNICORE Website: http://www.unicore.eu

Support list: unicore-support@lists.sf.net

Developer's list: unicore-devel@lists.sf.net

# 1 UNICORE UFTP

UFTP is a data streaming library and file transfer tool based with the following features:

- dynamic firewall port opening using a pseudo FTP connection, written by Tim Pohlmann (Juelich Supercomputing Centre, 2010)
- parallel input/output streams based on code from the JPARSS library, Copyright (c) 2001 Southeastern Universities Research Association, Thomas Jefferson National Accelerator Facility
- optional encryption of the data streams using a symmetric key algorithm

It is integrated into UNICORE version 6.4.1 and later, and allows to transfer data from client to server (and vice versa), as well as providing data staging between UFTP-enabled UNICORE sites.

The server part, called *uftpd*, listens on two ports (which may be on two different network interfaces):

- the command port receives control commands
- the listen port accepts data connections from clients.

The uftpd server is "controlled" by UNICORE/X via the command port, and receives/sends data directly from/to a user's client machine or another UFTP enabled UNICORE server. Data connnections are made to the "listen" port, which has to be accessible from external machines. Firewalls have to treat the "listen" port as an FTP port.

A UFTP file transfer works as follows:

- the UNICORE/X server sends a request to the command port. This request notifies the UFTPD server about the upcoming transfer and contains the following information
- the client's IP address
- the source/target file name
- whether to send or receive data

- a "secret", i.e. a string the client will send to authenticate itself
- how many data connections will be opened
- the user and group id for who to create the file (in case of send mode)
- an optional key to encrypt/decrypt the data
- the UFTPD server will now accept an incoming connection from the announced IP address, provided the supplied "secret" matches the expectation.
- if everything is OK, the requested number of data connections from the client can be opened. Firewall transversal will be negotiated using a pseudo FTP protocol.
- the file is sent/received using the requested number of data connections
- to access the requested file, uftpd attempts to switch its user id to the requested one prior to reading/writing the file. This uses a C library which is accessed from Java via the Java native interface (JNI). See also the installation section below.

---

**IMPORTANT SECURITY NOTE**
The UNICORE UFTPD server is running with root privileges. Make sure to read and understand the section below on protecting the command socket. Otherwise, users logged on to the UFTPD machine can possibly read and write other user's files.

---

# 2 Installation and use

## 2.1 Prerequisites

- Java 1.6 (or later) runtime is required
- the server "listen" port needs to be accessible through your firewalls, declaring it an "FTP" port
- the UFTPD server needs access to the target file systems
- a server certificate for the UFTPD server is strongly recommended for production use (see the section on SSL below)
- UNICORE 6.4.2 or later

**NOTE ON PATHS**
The UNICORE UFTP server is distributed either as a platform independent and portable tar.gz or zip bundle, or as an installable, platform dependent package such as RPM.
Depending on the installation package, the paths to various files are different. If installing using distribution-specific package the following paths are used:

```
CONF=/etc/unicore/uftpd
SBIN=/usr/sbin
BIN=/usr/bin
LOG=/var/log/unicore/uftpd
LIB=/usr/share/unicore/uftpd/lib
```

If installing using the portable bundle, all UFTP files are installed under a single directory. Path prefixes are as follows, where INST is the directory where UFTP was installed:

```
CONF=INST/conf
SBIN=INST/bin
BIN=INST/bin
LOG=INST/log
LIB=INST/lib
```

These variables (CONF, SBIN, BIN and LOG) are used throughout the rest of this manual.

## 2.2 C library for switching user ID

It may be required to re-compile the libuftp-unix.so library on your system. The one supplied with the distribution has been compiled on a 64bit Linux system. The folder LIB/native contains the required header and C source file, as well as an exemplary make file. Run "make install" to build the library, which will build and install the library into the LIB folder. If any problems occur during this procedure, please consult UNICORE support.

## 2.3 Starting and stopping the UFTPD server

In the SBIN directory, start/stop and status scripts are provided:

- `unicore-uftpd-start.sh` starts the server
- `unicore-uftpd-stop.sh` stops the server
- `unicore-uftpd-status.sh` checks the server status

The parameters such as server host/port, control host/port, and others are configured in the `CONF/uftpd.conf` file

In a production scenario with multiple users, the uftpd server needs to be started as root. This is necessary to be able to set the correct file permissions.

## 2.4 Configuration parameters

The following variables can be defined in the configuration file (uftpd.conf):

```
SERVER_HOST     : the interface where the server listens for client data
                  connections

SERVER_PORT     : the port where the server listens for client data
                  connections

CMD_HOST        : the interface where the server listens for control commands

CMD_PORT        : the port where the server listens for control commands

UFTPD_MEM       : the maximum memory allocated to the UFTPD server

MAX_CONNECTIONS : the maximum number of parallel TCP streams per client

BUFFER_SIZE     : the size of the buffer (in kilobytes) for reading/writing local

SSL_CONF        : File containing SSL settings for the command port

ACL             : File containing the list of server DNs that are allowed
                  access to the command port
```

As usual if you set the SERVER_HOST to be "0.0.0.0", the server will bind to all the available network interfaces.

If possible, use an "internal" interface for the Command socket. If that is not possible, make sure the Command socket is protected by a firewall!

We strongly recommend enabling SSL for the Command socket. Please refer to the next section.

## 2.5 Protecting the Command socket

Using SSL for the Command port ensures that only trusted users (rather, trusted UNICORE servers) can issue commands to the UFTPD server. To further limit the set of trusted users, an access control list (ACL) file is used.

In production settings where users can log in to the UFTPD server, SSL should ALWAYS be enabled.

### 2.5.1 SSL setup

**IMPORTANT SECURITY NOTE**
Without SSL enabled, users logged in to the UFTPD server can potentially read or write files with root privileges.

As usual for SSL, you need a keystore containing the UFTPD server's certificate, and a truststore containing certificate authorities that should be trusted. Keystore and truststore can be the same file.

The following properties can be set in the `CONF/udtpd-ssl.conf` file.

- `javax.net.ssl.keyStore` Keystore file containing the UFTPD key
- `javax.net.ssl.keyStorePassword` Keystore password
- `javax.net.ssl.keyStoreType` (optional) JKS or PKCS12, default is JKS
- `javax.net.ssl.trustStore` Truststore file containing the trusted CAs
- `javax.net.ssl.trustStorePassword` Truststore password

If the `javax.net.ssl.keyStore` property is NOT set, SSL will be disabled.

If the `javax.net.ssl.keyStorePassword` property is not set, the keystore password will be queried from the command line. If you want to use this, you will need to adapt the start script, avoiding the use of "nohup".

Please refer also to the UNICORE FAQ for keystore related questions.

### 2.5.2 ACL setup

The access control list contains the distinguished names of those certificates that should be allowed access.

The "ACL" setting in `CONF/uftpd.conf` is used to specify the location of the ACL file

```
export ACL=conf/uftpd.acl
```

The default ACL contains the certificate DN of the UNICORE/X server from the UNICORE core server bundle. In production, you need to replace this by the actual DNs of your UNICORE/X server(s).

The ACL entries are expected in RFC2253 format. To get the name from a certificate in the correct format using openssl, you can use the following OpenSSL command:

```
$> openssl x509 -in your_server.pem -noout -subject -nameopt RFC2253
```

### 2.6 Firewall configuration

NOTE

Please consult the firewall documentation on how to enable an "FTP" service on your firewall (or operating system).

With Linux iptables, you may use rules similar to the following:

```
iptables -A INPUT -p tcp -m tcp --dport $SERVER_PORT -j ACCEPT
iptables -A INPUT -p tcp -m helper --helper ftp-$SERVER_PORT -j  ↩
    ACCEPT
```

where $SERVER_PORT is the SERVER_PORT defined in uftpd.conf. The first rule allows anyone to access port $SERVER_PORT. The second rule activates the iptables connection tracking FTP module on port $SERVER_PORT.

On some operating systems it may be required to load additional kernel modules to enable connection tracking, for example on CentOS:

```
modprobe nf_conntrack_ipv4
modprobe nf_conntrack_ftp ports=$SERVER_PORT
```

## 3 UNICORE Integration

To enable a UNICORE/X server for the UFTP filetransfer the following settings have to be made.

### 3.1 Enabling the UFTP service

In the wsrflite.xml file, add a service definition for the UFTP filetransfer service:

```
<service name="FileTransferUFTP" wsrf="true" persistent="true">
   <interface class="de.fzj.unicore.uas.fts.FileTransfer" />
   <implementation class="de.fzj.unicore.uas.fts. ↩
       FileTransferHomeImpl"/>
</service>
```

### 3.2 Enabling the UFTP protocol

For those storages where you want UFTP enabled, you need to add it to the list of supported protocols. This is done in the UNICORE/X main config file `uas.config`.

```
# supported protocols
uas.sms.protocols=BFT UFTP RBYTEIO SBYTEIO
```

Please check the UNICORE/X manual for further details.

### 3.3 Configuration options

The UNICORE/X server needs to know the UFTPD server settings (i.e. host and port of both the command and the main listener socket). In the `uas.config` file, set the following properties:

```
# Listener (pseudo-FTP) socket
uftp.server.host=...
uftp.server.port=...

# Command socket
uftp.command.host=...
uftp.command.port=...

# Full path to the 'uftp.sh' client executable
uftp.client.executable=...

# How many parallel streams to use per file transfer
uftp.streams=2

# (optional) file read/write buffer in kbytes
uftp.buffersize=128
```

### 3.4 UFTP servers with multiple interfaces

If your UFTP server is on a machine with multiple network interfaces, you can give a comma-separated list of host names, like so:

```
uftp.server.host=net1.domain.org,net2.domain.org
```

The UFTP client in this case will try to connect to the hosts in the specified order, and will use the first "working" one.

### 3.5 Enabling data encryption

If you wish to encrypt the data sent/received by UFTPD (for example in data staging), the following property can be used:

```
# enable data encryption
uftp.encryption=true
```

This will encrypt data with a symmetric key using the Blowfish algorithm. Note that this costs some performance due to the additional CPU load.

### 3.6 Disabling SSL on the command port

While not recommended, it may be sometimes useful to disable SSL for communicating with the UFTPD, e.g. while setting up and testing. To do this add a property in `uas.config` (no UNICORE/X restart is required)

```
uftp.command.ssl.disable=true
```

### 3.7 Enabling "local" UFTP mode on the UNICORE/X server

In case the UNICORE/X server has direct access to the target file system, and you're not using the Perl TSI, it can be an interesting option to run the UFTP filetransfer client code directly in the UNICORE/X server instead of passing it to the TSI. This means more load in the UNICORE/X process. To enable local mode, edit `uas.config` and set

```
# enable local client mode
uftp.client.local=true
```

## 4 Testing the UFTP server without UNICORE

Testing as described in this section works only if SSL is not enabled. Therefore, you should run these tests as a non-root user. Enable SSL and restart the UFTPD server with root privileges once you are finished with these tests.

The UFTP distribution contains two scripts that allow you to test the UFP functionality without using UNICORE. Making a data transfer involves two steps:

- invoke `uftp-job.sh` to "announce" an upcoming transfer to the UFTPD server
- invoke `uftp.sh` to initiate the actual transfer

Note, in case you installed from an RPM or DEB package, these files are located in `/usr/bin`.

The following shell commands "transfer" the file `.bashrc` to the /tmp directory.

Assuming you installed from RPM/DEB:

```
. /etc/unicore/uftpd/uftpd.conf
unicore-uftpd-job.sh -c localhost -f ~/.bashrc -s true -x my_secret ↩
     -n 2 -u unicore -g unicore
uftp.sh -r -f /tmp/test -L $SERVER_PORT -l $SERVER_HOST -x  ↩
    my_secret -n 2
```

This should create a file `/tmp/test` identical to `~/.bashrc`. Check the console output and the UFTPD log file `LOG/uftpd.log` in case of errors.

After the transfer finished, check that indeed

```
md5sum /tmp/test ~/.bashrc
```

gives the correct checksum for the newly created file.

It is also possible to enable encryption "manually", by appending "-E <key>" to the commands above, where "key" is a sequence of 12 characters (really a base64-encoded 64 bit key).