Table of Contents

Enable HTTP protocol for

Configuration

The `xrootd` daemon can export both the `xrootd` protocol and the `https` protocol; this page covers the configuration changes required for enabling HTTPS.

For clients to successfully read from the regional redirector, `HTTPS` must be enabled for the data servers and the site-level redirector.

Versions and Platforms

We strongly suggest utilizing RHEL7 and XRootD 4.9.1.

The latest version of `xrootd-lcmaps` (`1.3` or later) has features required by CMS and doesn't support RHEL6.

Xrootd Config Changes

Verify you have the most up-to-date version of the `sec.protocol` line:

```
sec.protocol /usr/lib64 gsi \
   -certdir:/etc/grid-security/certificates \
   -cert:/etc/grid-security/xrd/xrdcert.pem \
   -key:/etc/grid-security/xrd/xrdkey.pem \
   -crl:1 \
   -authzfun:libXrdLcmaps.so \
   -authzfunparms:--lcmapscfg,/etc/xrootd/lcmaps.cfg,--loglevel,0 \
   -gmapopt:10 \
   -gmapto:0
```

In the `xrootd` daemon specific configuration, enable HTTPS:

```
if exec xrootd
  xrd.protocol http:1094 /usr/lib64/libXrdHttp-4.so
  http.cadir /etc/grid-security/certificates
  http.cert /etc/grid-security/xrd/xrdcert.pem
  http.key /etc/grid-security/xrd/xrdkey.pem
  http.secxtractor /usr/lib64/libXrdLcmaps.so
  http.listingdeny yes
  http.staticpreload http://static/robots.txt /etc/xrootd/robots.txt
  http.desthttps yes
fi
```

Updates to Auth database contents

The `/etc/xrootd/Authfile` should contain these lines:

```
g /cms /store lr
```

Other VOs may need additional authorizations; note that the VOMS attributes are mapped to the XRootD group structure. The above example only covers reads; writes for CMS are covered later in the document.

In particular, it should **not** have a line that looks like this:

```
u * /store lr
```

Create robots.txt

Add the file `robots.txt` to `/etc/xrootd` with these contents:

```
User-agent: *
Disallow: /
```

We recommend doing this in order to prevent search engines from trying to index your site.

PhEDEx Configuration

Update `storage.xml` to have a `davs` protocol. Add a line that looks like this:

```
  <lfn-to-pfn protocol="davs"  destination-match=".*" chain="direct" path-match="(.*)" result="da
```

The PhEDEx file export agent should be changed to additionally export the `davs` protocol:

```
### AGENT LABEL=exp-pfn PROGRAM=Toolkit/Transfer/FileExport
 -db          ${PHEDEX_DBPARAM}
 -nodes       ${PHEDEX_NODE}
 -storagemap  ${PHEDEX_MAP}
 -protocols   'srmv2,direct,davs'
```

(the existing `protocols` argument needs to be changed). Verify you change the configuration of both the Debug and Prod agents. The agent must be restarted for the change to take effect.

Testing the configuration

Make sure you see file not found from any browser session, even if you have your certificate loaded. This should fail because a VOMS extension isn t present in your browser; we want to require VOMS.

From the terminal, generate a CMS VOMS proxy and attempt to use `davix-get` to copy from your XRootD host:

```
davix-get https://cmstest2.rcac.purdue.edu:1094/store/user/goughes/test.root -P grid
```

Do not use `curl` as it does not support proxy certificates.

Enable HTTP-based Writes

Xrootd Authorization changes

The primary changes are to the `Authfile`; you will need to add several `a` (all) authorizations to where users need to be able to write. Here's an example:

```
t writecmsdata /store/backfill/          a  \
               /store/data/              a  \
               /store/generator/         a  \
               /store/group/             a  \
               /store/hidata/            a  \
               /store/mc/                a  \
               /store/PhEDEx_LoadTest07/ a  \
               /store/relval/            a  \
               /store/temp/              a  \
               /store/unmerged/          a

t readcmsdata  /store/                   lr

# cmsprod and PhEDEx have full access to managed CMS data, and read for CMS
u cmsprod  writecmsdata readcmsdata
u cmsphedex writecmsdata readcmsdata

# lcgadmin can write into /store/user/sam and /store/unmerged/SAM.
u lcgadmin readcmsdata /store/user/sam/ a /store/unmerged/SAM a

# CMS users have full access to their own directory, and read for CMS
# While xrootd allows the user to *attempt* any operation - even in other user's
# home directories - the underlying filesystem also has its internal permissions and will further
# limit things.
g /cms /store/user a /store/temp/ a readcmsdata
```

Notes:

- This guide assumes that you have an underlying filesystem (HDFS, Lustre, etc) implementing filesystem permissions for `/store/user` directories. If not (pure Xrootd, files are written as the same user), you will need one line per CMS username.
- List authorizations from most specific to least specific.
- The first two columns must be unique; if multiple authorizations are needed for a user, add them to the same line.
- `t` is a template
- `u` lines are for users as mapped by LCMAPS (such as `cmsprod`). These should correspond to Unix usernames at your site.
- `g` lines refer to VOMS groups (such as `/cms`). These do **not** correspond to Unix group names at your site.
- If you have `u *`, recall this allows ALL users, including unauthenticated. This includes random web spiders!

The upstream documentation has further information on the `Authfile` format.

Xrootd Checksum changes

In order for the checksumming functionality to work, additional RPM upgrades may be needed:

- `xrootd-4.9.1` or later is needed (as of April 2019, this is in the OSG release repo)
- `xrootd-hdfs-2.1.3` or later. Currently in the OSG release; only necessary for HDFS sites.

Additionally, we need to enable checksums in the configuration file:

```
xrootd.chksum max 2 md5 adler32 crc32
# Below line is only necessary for HDFS sites
ofs.ckslib * /usr/lib64/libXrdHdfs.so
```

CMS-specific changes

- Update `storage.xml` to have a `davs` protocol. Add a line that looks like this:

  ```
  <lfn-to-pfn protocol="davs"  destination-match=".*" chain="direct" path-match="(.*)" res
  ```
- Update the `site-local-config.xml` to point the stageout protocol to `davs`. Inside the `<local-stage-out>` stanza, convert the catalog protocol to `davs`. Example:

  ```
   <catalog url="trivialcatalog_file://etc/cvmfs/SITECONF/PhEDEx/storage.xml?protocol=davs
  ```

It is best to test this out on a few worker nodes before deploying widely. SITECONF update example from Caltech:
https://gitlab.cern.ch/SITECONF/T2_US_CALTECH/commit/edc766852de59923c075061e04e8e8d572e94276

Testing the configuration

From the terminal, generate a CMS VOMS proxy and attempt to use `davix-get` to copy from your XRootD host:

```
davix-put /tmp/hello_world.txt https://cmstest2.rcac.purdue.edu:1094/store/user/goughes/test.root
```

OR

```
gfal-copy  /tmp/hello_world.txt https://transfer-1.ultralight.org:1094//store/temp/user/localtest
Copying file:///tmp/hello_world.txt   [DONE]  after 4s
```

Do not use `curl` as it does not support proxy certificates.

Enable Third-Party-Copy

The OSG has third-party-copy support for XRootD; this section walks through enabling the support.

Install RPMs

You will need the XRootD `4.9.1` (or later) release.

If you are a HDFS site, you will also to have `xrootd-hdfs` version 2.1.3 or later; this is only in OSG 3.4.

Configuration Changes

If you have a line like this in your XRootD config, remove it:

```
xrootd.fslib /usr/lib64/libXrdOfs.so
```

This is no longer necessary and breaks the configuration for the third party copy daemon.

Add the following lines for the `xrootd` daemon:

```
if exec xrootd
# Enable third-party-copy
http.exthandler xrdtpc libXrdHttpTPC.so

# Pass the bearer token to the Xrootd authorization framework.
http.header2cgi Authorization authz
fi

# Enable Macroons-based mappings; if no token is present, then the GSI certificate will be used.
ofs.authlib libXrdMacaroons.so
```

Follow the Macaroons support configuration section below to finalize the Macaroons pieces.

Testing Setup

You may utilize the FTS server at https://fts3-devel.cern.ch:8446.

```
fts-transfer-submit --overwrite -s https://fts3-devel.cern.ch:8446 \
    https://xrootd-local.unl.edu:1094//user/uscms01/pnfs/unl.edu/data4/cms/store/mc/RunIISpring18
    davs://transfer-8.hep.caltech.edu:1094//store/user/bbockelm/test/writes_new/scitokens.8 \
    --blocking --verbose
```

Changes

Change your `storage.xml` to provide a mapping for the `davs` protocol:

```
  <lfn-to-pfn protocol="davs"  destination-match=".*" chain="direct" path-match=".*/LoadTest07_Ne
```

Update as appropriate for your site. Remember to synchronize this with CVMFS and any other place you store your `storage.xml`.

Next, startup a dedicated transfer agent in the Debug instance:

```
### AGENT LABEL=download-davs PROGRAM=Toolkit/Transfer/FileDownload
 -db              ${PHEDEX_DBPARAM}
 -nodes           ${PHEDEX_NODE}
 -delete          ${PHEDEX_CONF}/FileDownloadDelete
 -validate        ${PHEDEX_CONF}/FileDownloadVerify,-d
 -accept          'T2_US_Nebraska'
 -verbose
 -backend         FTS3
 -service       https://fts3-devel.cern.ch:8446
 -protocols       davs
 -batch-files     10
 -max-active-files 30
 -jobs            3
```

Make sure your primary download agent ignores the same link so there aren't two agents attempting transfers on the same link.

Patch PhEDEx with the following:
https://patch-diff.githubusercontent.com/raw/dmwm/PHEDEX/pull/1123.patch.

Macaroons Support

Macaroons are a token format that allows delegation of authorization and attenuation. They are a convenient way to enable a user to upload or download a single file without having a grid proxy and are used by FTS to perform third party copies.

Install the plugin

This is installed by default with XRootD 4.9.1.

Configuration

Add the following lines to your xrootd configuration:

```
http.exthandler xrdmacaroons libXrdMacaroons.so
macaroons.secretkey /etc/xrootd/macaroon-secret
ofs.authlib libXrdMacaroons.so libXrdAccSciTokens.so
```

The secret key is a symmetric key necessary to verify macaroons; the same key must be deployed to all XRootD servers in your cluster (so puppetize its distribution).

The secret key must be base64-encoded. The most straightforward way to generate this is the following:

```
openssl rand -base64 -out /etc/xrootd/macaroon-secret 64
```

NOTE: The current implementation is sensitive to errant newline characters. Use the `openssl` command above and try to avoid editing with a text editor.

Usage

To generate a macaroon for personal use, you can run:

```
$ macaroon-init -v
Usage: macaroon-init URL validity_min ACTIVITY ...

$ macaroon-init https://host.example.com//path/to/directory/ 60 DOWNLOAD,UPLOAD
```

(the `macaroon-init` CLI can be found as part of the `x509-scitokens-issuer-client` package). This will generate a macaroon with 60 minutes of validity that has upload and download access to the path specified at `/path/to/directory`, provided that your X509 identity has that access.

The output will look like the following:

```
Querying https://host.example.com//path/to/directory/ for new token.
Validity: PT60M, activities: DOWNLOAD,UPLOAD,READ_METADATA.
Successfully generated a new token:
{
  "macaroon":"MDAxY2xvY2F0aW9uIFQyX1VTX05lYnJhc2thCjAwMzRpZGVudGlmaWVyIGMzODU3MjQ3LThjYzItNGI0YS0
}
```

The contents of the `macaroon` key is your new security token. Anyone you share it with will be able to read and write from the same path. You can use this token as a bearer token for HTTPS authorization. For example, it can authorize the following transfer:

```
curl -v
```

```
-H 'Authorization: Bearer MDAxY2xvY2F0aW9uIFQyX1VTX05lYnJhc2thCjAwMzRpZGVudGlmaWVyIGMzODU3Mj
https://host.example.com//path/to/directory/hello_world
```

Enable

The OSG has support for SciTokens within XRootD; this section walks through enabling the support.

Install RPMs

You will need the XRootD `4.9.1` (or later) release and the `xrootd-scitokens` RPM.

If you are a HDFS site, you will also to have `xrootd-hdfs` version 2.1.3 or later; this is only in OSG 3.4.

Configuration Changes

Add the following lines for the `xrootd` daemon:

```
if exec xrootd
# Pass the bearer token to the Xrootd authorization framework.
http.header2cgi Authorization authz
fi

# Enable Macroons- and SciTokens-based mappings; if no token is present, then the GSI certificate
ofs.authlib libXrdMacaroons.so libXrdAccSciTokens.so
```

Create `scitokens.cfg`

Create `/etc/xrootd/scitokens.cfg` with the following contents:

```
[Issuer CMS]

issuer = https://scitokens.org/cms
base_path = /
map_subject = False
default_user = cmsprod
```

Adjust `base_path` to be wherever /store lives inside your HDFS. The issuer enabled at `https://scitokens.org/cms` will be able to issue tokens for anything inside that base path. Additionally, the `default_user` should be set to the username that should own all CMS files written via HTTP.

At Nebraska, all files are written inside `/user/uscms01/pnfs/unl.edu/data4/store`, so we have set `base_path = /user/uscms01/pnfs/unl.edu/data4/`

Testing Setup

To test the SciTokens support:

- Install `x509-scitokens-issuer-client` via yum. It's currently in the HCC repo or available in source form at https://github.com/scitokens/x509-scitokens-issuer (it's working its way into the `osg-contrib` to make this step easier).
- Run `x509-scitoken-init https://cmsweb.cern.ch` with a CMS VOMS proxy in your environment (do not use a host certificate). Should print a big nasty token to stdout. Copy that.
- Running `x509-scitoken-init https://cmsweb.cern.ch` did not work, but running `x509-scitoken-init https://scitokens.org/cms` does work.
- Try to `curl` something with the token from the previous step set in the `Authorization` header. Example:

```
curl --capath /etc/grid-security/certificates -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5c
```

For third-party-copy support: you may utilize the FTS server at https://fts3-devel.cern.ch:8446.

```
fts-transfer-submit --overwrite -s https://fts3-devel.cern.ch:8446 \
    https://xrootd-local.unl.edu:1094//user/uscms01/pnfs/unl.edu/data4/cms/store/mc/RunIISpring18
    davs://transfer-8.hep.caltech.edu:1094//store/user/bbockelm/test/writes_new/scitokens.8 \
    --file-metadata '{"source-issuer": "https://scitokens.org/cms", "dest-issuer": "https://scito
    --blocking --verbose
```

Reference Material

Nebraska currently runs the following set of RPMs plus dependencies:

- xrootd-4.9.0-rc1
- xrootd-lcmaps-1.4.0-1.20180809.1.hcc.el7.x86_64
- xrootd-hdfs-2.1.3-1.hcc.el7.x86_64
- xrootd-cmstfc-1.5.2-1.osg34.el7.x86_64
- xrootd-scitokens-0.5.0-1.hcc.el7.x86_64

RPMs with `hcc` in the release number are from the HCC repo while those with `osg34` are from OSG 3.4 release repo. The experimental xrootd build contains the checksum feature slated for Xrootd 4.9.0.

This is a lightly cleaned-up version of the main Xrootd configuration file:

```
# Port specifications; only the redirector needs to use a well-known port
# "any" will cause rooted to bind to any available port.  Change as needed for firewalls.
xrd.port 1094

all.sitename T2_US_Nebraska

# The roles this server will play.
all.role server

# The known managers
all.manager xrootd-local.unl.edu:1213

# Allow any path to be exported; this is further refined in the authfile.
all.export / nostage

# Hosts allowed to use this xrootd cluster
cms.allow host *

### Standard directives
# Simple sites probably don't need to touch these.
# Logging verbosity
xrootd.trace emsg login stall redirect
ofs.trace none
xrd.trace conn
cms.trace all

# Integrate with CMS TFC, placed in /etc/storage.xml
oss.namelib /usr/lib64/libXrdCmsTfc.so file:/etc/xrootd/storage.xml?protocol=hadoop

# Turn on authorization
ofs.authorize 1
acc.authdb /etc/xrootd/Authfile
acc.audit deny grant

# Enable verification of write/update commands
sec.level all compatible

# Security configuration
sec.protocol /usr/lib64 gsi -certdir:/etc/grid-security/certificates -cert:/etc/grid-security/xrd

xrootd.seclib /usr/lib64/libXrdSec.so
ofs.osslib /usr/lib64/libXrdHdfs.so

if exec xrootd
xrd.protocol http:1094 /usr/lib64/libXrdHttp.so
http.cadir /etc/grid-security/certificates
http.cert /etc/grid-security/xrd/xrdcert.pem
http.key /etc/grid-security/xrd/xrdkey.pem
http.secxtractor /usr/lib64/libXrdLcmaps.so
http.listingdeny yes
http.staticpreload http://static/robots.txt /etc/xrootd/robots.txt
#http.selfhttps2http yes

# Enable third-party-copy
http.exthandler xrdtpc libXrdHttpTPC.so

# Pass the bearer token to the Xrootd authorization framework.
http.header2cgi Authorization authz

fi

# Enable SciTokens-based mappings; if no token is present, then the GSI certificate will be used.
ofs.authlib libXrdMacaroons.so libXrdAccSciTokens.so

cms.delay startup 10
cms.fxhold 60s

xrd.report xrootd.t2.ucsd.edu:9931 every 30s all sync
xrootd.monitor all auth flush io 30s ident 5m mbuff 2048 window 5s dest files io info user xrootd

xrootd.async off

# Enable checksum support
xrootd.chksum max 2 md5 adler32 crc32
ofs.ckslib * /usr/lib64/libXrdHdfs.so
```

Here is the authfile:

```
t writecmsdata /store/backfill/          a  \
               /store/data/              a  \
               /store/group/             a  \
               /store/hidata/            a  \
               /store/mc/                a  \
               /store/PhEDEx_LoadTest07/ a  \
               /store/relval/            a  \
               /store/temp/              a  \
               /store/unmerged/          a

t readcmsdata  /store/                   lr

# cmsprod and PhEDEx have full access to managed CMS data, and read for CMS
u cmsprod    writecmsdata readcmsdata
u cmsphedex  writecmsdata readcmsdata

# lcgadmin can write into /store/user/sam.
u lcgadmin readcmsdata /store/user/sam/ a /store/unmerged/SAM a

# CMS users have full access to their own directory, and read for CMS
# While xrootd allows the user to *attempt* any operation - even in other user's
# home directories - HDFS also has its internal permissions and will further
# limit things.
g /cms /store/user a /store/temp/ a readcmsdata

# LIGO can read its own files and write into the test directory.
u ligo /user/ligo lr

# DTEAM has read/write access to its own files
u dteam /user/dteam a

# All users and anonymous can read the test_access directory, but nowhere else.
u * /user/ligo/test_access/ lr
```

Here is the scitokens configuration file:

```
[Issuer CMS]
issuer = https://scitokens.org/cms
base_path = /user/uscms01/pnfs/unl.edu/data4/cms
# For CMS, there is no relationship between local usernames
# and the VO name.
map_subject = False
default_user = cmsprod

[Issuer dteam]
issuer = https://scitokens.org/dteam
base_path = /user/dteam
```

We have a fairly standard LCMAPS setup as covered by the OSG

This topic: Main > XRootDoverHTTP
Topic revision: r22 - 2019-05-14 - ErikGough