OPeNDAP Authentication For DAP Clients

Table of Contents

1. Overview

This document describes how to configure DAP clients to access data servers that require users to be authenticated. The document is divided into four sections. This Overview provides a brief introduction to the topic and some background on commonly used clients. The remaining sections provide information on three different authentication systems in use today: NASA's Earthdata Login, LDAP, and Shibboleth.

Many users access DAP servers using a browser as their primary software interface. However, there is also a growing group of users that utilize either:

- A "smart" tool. Where "smart" means that the tool understands how to interact with a DAP service and construct DAP queries for data and use that in higher level client side activities like GUI based graphs, image display, selection, navigation, etc.
- A command line tool such as "wget" or "curl" that can be used to extract data from a DAP service, but the URL construction is left to the user or other software.

In both these cases we want these client software applications to be able to manage authentication. In the case of command line style tools, being able to authenticate without a user interaction is important for automated work flows.

1.1. Some DAP Data Access Clients

The collection of software clients discussed here have some remarkable similarities in the way that they handle the client end of authentication. The clients that present a GUI to the user (Panoply, IDV, and ToolsUI) will open a dialog box and prompt the user for credentials when the server requires user authentication. The rest of these clients listed here use a file called `.netrc` to store credentials. The `.netrc` file is a simple text file and is described in detail in Our friend, the *.netrc* file.

Some DAP client software applications (those that utilize one of UNIDATA's NetCDF libraries to provide remote data access) will use the `.dodsrc` file to store important authentication related information. The `.dodsrc` file is a simple text file and is typically located in the users home directory. See *ncdump, nccopy, Matlab,* and other applications that use *NetCDF-C* for more information about the `.dodsrc` file.

We tested each of the authentication steps on the folowing applications:

- ***cURL (a.k.a. lib_curl or curl)*** The cURL library provides the HTTP functionality for a surprisingly large part the world wide web. The cURL library also has a command line application, `curl`, that can be used to issue HTTP requests and collect the responses. It can be a very powerful data access tool when used manually or in the scope of a larger shell script.
    - *Credentials*: ***~/.netrc***
    - *Uses ~/.dodsrc*: ***no***
- ***ncdump, nccopy, netCDF4 Python, Matlab, and other applications using NetCDF-C*** The `ncdump` and `nccopy` command line applications are part of the NetCDF-C library and can be utilized to access DAP resources. This means we can use `ncdump` and `nccopy` as an easy sanity check for authentication issues for command line applications that uses the netCDF-C library. Because the netCDF-C library is the software component that is performing the authentication, the configuration steps outlined here should directly translate to any application that uses netCDF-C. These steps were tested against netCDF-C version 4.9.0. If you have a different version you may wish to visit Unidata for the related documentation.

    - *Credentials*: ***~/.netrc***
    - *Uses ~/.dodsrc*: ***yes***
- ***Integrated Data Viewer (IDV)*** The Integrated Data Viewer is GUI driven data client that is based around the CDM/NetCDF data model and utilizes that NetCDF-Java (and thus the Java DAP implementation) to access remote DAP datasets. Because it has a GUI it can retrieve (and cache for later) users credentials directly from the user. Since IDV utilizes the Java-NetCDF library to access DAP resources then in theory if it works for IDV then it should work for all the other clients that use the Java-NetCDF library.
    - *Credentials*: ***GUI***
    - *Uses ~/.dodsrc*: ***no***
- ***ToolsUI*** The ToolsUI application is a simple is GUI driven data client that is based around the CDM/NetCDF data model and utilizes that NetCDF-Java (and thus the Java DAP implementation) to access remote DAP datasets. Because it has a GUI it can retrieve (and cache for later) users credentials directly from the user.
    - *Credentials*: ***GUI***
    - *Uses ~/.dodsrc*: ***no***
- ***Panoply*** The Panoply application is a sophisticated GUI driven data client that is based around the CDM/NetCDF data model and utilizes that NetCDF-Java (and thus the Java DAP implementation) to access remote DAP datasets. Because it has a GUI it can retrieve (and cache for later) users credentials directly from the user.
    - *Credentials*: ***GUI***
    - *Uses ~/.dodsrc*: ***no***
- ***PyDAP*** The PyDAP python software package provides one interface for python programs to read from OpenDAP servers (another is the netCDF4 Python module, which uses the netCDF-C library to actually access data, including data from OpenDAP servers).
    - *Credentials*: ***~/.netrc***
    - *Uses ~/.dodsrc*: ***no***

1.2. Our friend, the `.netrc` file

The `~/.netrc` file (a file named ***.netrc*** located in your home directory, aka ~) is the way in which many of the data access clients discussed here expect (i.e., are coded) to access credentials to be used in an authentication exchange.

Because the `.netrc` file name starts with a dot, it is a hidden file and will not be visible in a standard directory listing. You can use the `ls -a` command to see all files in a directory, including hidden files.

Add login credentials for various servers to the `~/.netrc` file, associating each set of credentials with the appropriate login endpoint. Here is an Earth Data Login example:

```
machine urs.earthdata.nasa.gov
    login your_edl_uid
    password your_edl_password
```

And you can add more credentials and machines (aka single sign-on endpoints) as needed:

```
machine urs.earthdata.nasa.gov
    login your_edl_uid
    password your_edl_password

machine sso.noaa.gov
    login your_noaa_uid
    password your_noaa_password

machine some.ldap.service.edu
    login your_ldap_uid
    password your_ldap_password
```

NOTE It is crucial that the access permissions be set to allow reading and writing only by the account owner. This can be done using the command: `chmod 600 ~/.netrc` If *group* or *other* are allowed any access to the `~/.netrc` file the data access client software will ignore (possibly silently) the `~/.netrc` file and authentication will fail.

2. Earthdata Login (OAuth2)

Earthdata Login is a NASA implementation of an OAuth2 Single Sign-On service. In order to access NASA held data you will need to:

1. Obtain (for free) your own set of Earthdata Login credentials.
2. *Approve* the DAP server application that is serving the data you want, as described in the following section.
3. Use this guide to configure your DAP client of choice (*curl*, browser, etc.) to utilize these credentials.

2.1. Setting Up Earthdata Login

2.1.1. Acquire Earthdata Login Credentials

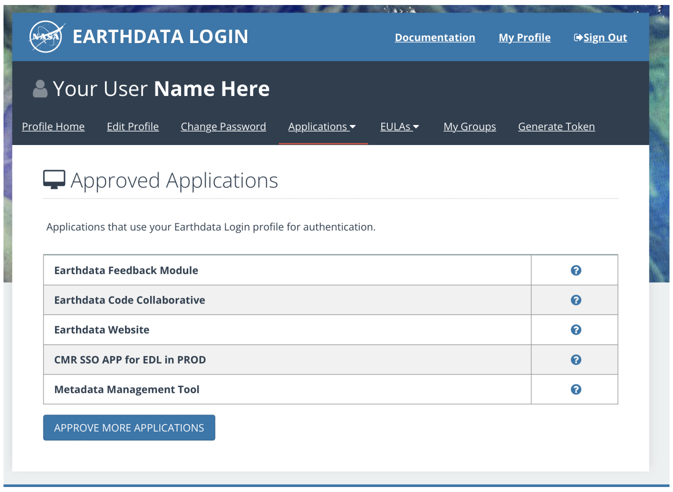
Registering with Earthdata Login (EDL) and getting a user account is free. Point you browser at the Earthdata Login New User page and do the stuff that needs the doing.

2.1.2. Approving The Hyrax OPeNDAP Service Application

Regardless of which software client you decide to employ, before you can access any new Earthdata Login authenticated server you must first add that sever to the list of **Approved Applications** in your Earthdata Login profile.

To do this you will need the Earthdata Login Application name (aka UID) under which the DAP server is registered with Earthdata Login and your Earthdata Login credentials.

- With your browser, navigate to your Earthdata Login profile page.
- Click the **Applications** tab and select **Authorized Applications** from the pull down menu.This will take you to the **Approved Applications** page.

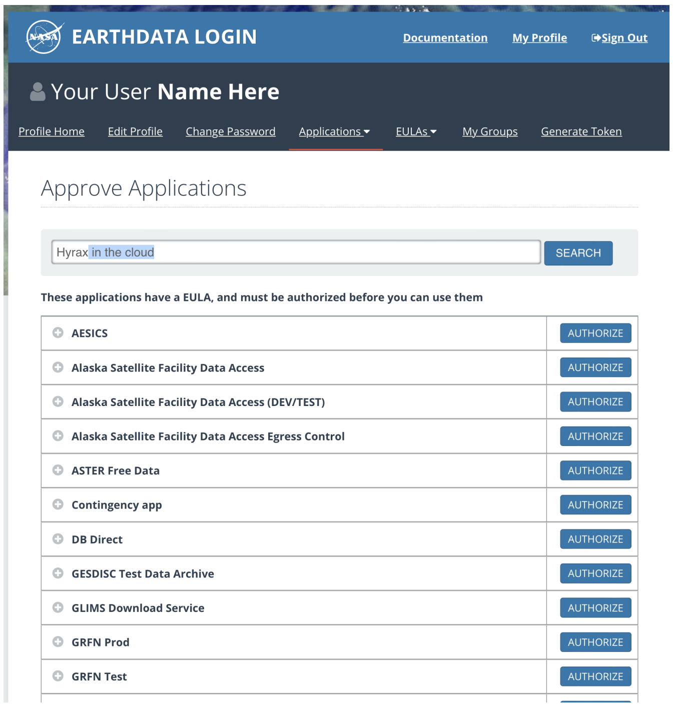


Applications that use your Earthdata Login profile for authentication.

Earthdata Feedback Module
Earthdata Code Collaborative
Earthdata Website
CMR SSO APP for EDL in PROD
Metadata Management Tool

On the **Approved Applications** page:

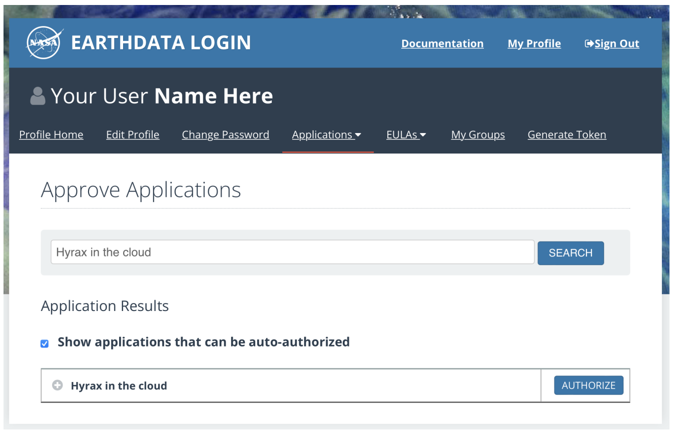
- At the bottom of the page click the **Approve More Applications** button. This will display the **Approve Applications** page.



These applications have a EULA, and must be authorized before you can use them

AESICS	AUTHORIZE
Alaska Satellite Facility Data Access	AUTHORIZE
Alaska Satellite Facility Data Access (DEV/TEST)	AUTHORIZE
Alaska Satellite Facility Data Access Egress Control	AUTHORIZE
ASTER Free Data	AUTHORIZE
Contingency app	AUTHORIZE
DB Direct	AUTHORIZE
GESDISC Test Data Archive	AUTHORIZE
GLIMS Download Service	AUTHORIZE
GRFN Prod	AUTHORIZE
GRFN Test	AUTHORIZE

In the search bar at the top of the page enter the name of the Hyrax OPeNDAP service application, ***Hyrax in the cloud*** and then click the **Search** button, this will bring you to the Earthdata Login Application Approval page:



- Click the *Authorize* button associated with the **Hyrax in the cloud_** service. You will be returned to the **My Applications** page where you should now see your new application on the list of *Approved Applications*.

NOTE

The application named **Hyrax in the cloud** used in this example is the only OPeNDAP service application running in the NASA cloud. There are many other applications deployed in NASA and to use any of them with your EDL user account you will have to authorize each application service in a similar manner to ***Hyrax in the cloud***

2.2. *cURL* (a.k.a. *lib_curl* or *curl*)

You can use command line *curl* to retrieve EDL authenticated resources using the following technique.

Create a `~/.netrc` file as described in the .netrc section above

Edit the `~/.netrc` file and associated your EDL credentials with the EDL service endpoint utilized by your target DAP server:

```
machine urs.earthdata.nasa.gov
    login your_edl_uid
    password your_edl_password
```

If the configuration is correct you should now be able to retrieve a DDS object in from the associated DAP service with the following *curl* command:

```
#!/bin/bash
touch cookie_file
curl -n -c cookie_file -b cookie_file -L --url
https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
```

What is happening here?

In this request cURL is both authenticating and maintaining a session with the remote server (this is accomplished by telling cURL to load and save cookies from the same file, *cookie_file*)

`-n`

This tells *cURL* to authenticate using the *~/.netrc* file you created/updated.

`-c cookie_file`

This tells *cURL* to stash cookies in the file *cookie_file*

`-b cookie_file`

This tells *cURL* to read cookies from the file *cookie_file*

`-L`

This option (aka *--location*) tells *cURL* to follow redirects, which is a must for any Single Sign On (SSO) authentication flow, such as OAuth2.

NOTE Do not use the `--location-trusted` option. It will cause *cURL* to spread user credentials to servers other than to which they were associated.

`--url https://opendap.earthdata.nasa.gov/ …`

The desired URL, protected by the Earthdata Login authentication flow.

In order to retrieve multiple URLs without re-authenticating you can use multiple instances of the *--url* parameter:

```
#!/bin/bash
curl -k -n -c cookie_file -b cookie_file -L \
    --url https://opendap.earthdata.nasa.gov/ \
    --url https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds \
    --url https://opendap.earthdata.nasa.gov/hyrax/data/nc/coads_climatology.nc.dds
```

Or, since *cURL* is actually pretty smart about using cookies and such you can also make multiple *curl* requests with the same cookies, and it won't have to reauthenticate with EDL once it's authenticated the first time:

```
#!/bin/bash
curl -k -n -c cookie_file -b cookie_file -L --url
https://opendap.earthdata.nasa.gov/hyrax/
curl -k -n -c cookie_file -b cookie_file -L --url
https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
curl -k -n -c cookie_file -b cookie_file -L --url
https://opendap.earthdata.nasa.gov/hyrax/data/nc/coads_climatology.nc.dds
```

2.3. *wget*

The *wget* documentation indicates that *wget* will automatically locate and utilize the *.netrc* file that we created for *curl*.

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your EDL credentials with the EDL service endpoint.

And happily it appears to work, as long as the `~/.netrc` file is in place.

Consider this *wget* command:

```
#!/bin/bash
wget  --load-cookies cookie_file --save-cookies cookie_file --keep-session-cookie
https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
```

What is happening here?

In this request *wget* is both authenticating and maintaining a session with the remote server (the latter is accomplished by telling wget to load and save cookies from the same file, *cookie_file*)

`--load-cookies cookie_file`

Load cookies from the file "cookie_file"

`--save-cookies cookie_file`

Save cookies to the file "cookie_file"

`--keep-session-cookie`

Save session cookies.

`https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds`

The URL to retrieve.

Here’s the request:

```
#!/bin/bash
touch cookie_file # Make sure the cookie file exists
wget --load-cookies cookie_file --save-cookies cookie_file --keep-session-cookie
https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
```

Here’s the output:

```
--2014-11-14 11:22:18--  https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
Connecting to opendap.earthdata.nasa.gov:443... connected.
WARNING: cannot verify opendap.earthdata.nasa.gov's certificate, issued by
`/C=US/ST=RI/L=Narragansett/O=OPeNDAP
Inc./OU=Engineering/CN=opendap.earthdata.nasa.gov/emailAddress=support@opendap.org':
  Self-signed certificate encountered.
HTTP request sent, awaiting response... 302 Found
Location: https://urs.earthdata.nasa.gov/oauth/authorize?app_type=401&client_id
=04xHKVaNdYNzCBG6KB7-Ig&response_type=code&redirect_uri
=https%3A%2F%2Fopendap.earthdata.nasa.gov%2Fopendap%2Flogin&state=aHR0cHM6Ly81NC4xNzIu
```

```
OTcuNDcvb3BlbmRhcC9kYXRhL25jL2Zub2MxLm5jLmRkcw [following]
--2014-11-14 11:22:19--  https://urs.earthdata.nasa.gov/oauth/authorize?app_type
=401&client_id=04xHKVaNdYNzCBG6KB7-Ig&response_type=code&redirect_uri
=https%3A%2F%2Fopendap.earthdata.nasa.gov%2Fopendap%2Flogin&state=aHR0cHM6Ly81NC4xNzIu
OTcuNDcvb3BlbmRhcC9kYXRhL25jL2Zub2MxLm5jLmRkcw
Resolving urs.earthdata.nasa.gov... 198.118.243.34, 2001:4d0:241a:4089::91
Connecting to urs.earthdata.nasa.gov|198.118.243.34|:443... connected.
WARNING: certificate common name `earthdata.nasa.gov' doesn't match requested host
name `urs.earthdata.nasa.gov'.
HTTP request sent, awaiting response... 401 Unauthorized
Connecting to urs.earthdata.nasa.gov|198.118.243.34|:443... connected.
WARNING: certificate common name `earthdata.nasa.gov' doesn't match requested host
name `urs.earthdata.nasa.gov'.
HTTP request sent, awaiting response... 302 Found
Location: https://opendap.earthdata.nasa.gov/hyrax/login?code
=a590cfc189783e29a7b8ab3ce1e0357618cbab3f590e7268a26e7ad1f7cf899d&state=aHR0cHM6Ly81NC
4xNzIuOTcuNDcvb3BlbmRhcC9kYXRhL25jL2Zub2MxLm5jLmRkcw [following]
--2014-11-14 11:22:20--  https://opendap.earthdata.nasa.gov/hyrax/login?code
=a590cfc189783e29a7b8ab3ce1e0357618cbab3f590e7268a26e7ad1f7cf899d&state=aHR0cHM6Ly81NC
4xNzIuOTcuNDcvb3BlbmRhcC9kYXRhL25jL2Zub2MxLm5jLmRkcw
Connecting to opendap.earthdata.nasa.gov:443... connected.
WARNING: cannot verify opendap.earthdata.nasa.gov's certificate, issued by
`/C=US/ST=RI/L=Narragansett/O=OPeNDAP
Inc./OU=Engineering/CN=opendap.earthdata.nasa.gov/emailAddress=support@opendap.org':
  Self-signed certificate encountered.
HTTP request sent, awaiting response... 302 Found
Location: https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds [following]
--2014-11-14 11:22:21--  https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc.dds
Connecting to opendap.earthdata.nasa.gov:443... connected.
WARNING: cannot verify opendap.earthdata.nasa.gov's certificate, issued by
`/C=US/ST=RI/L=Narragansett/O=OPeNDAP
Inc./OU=Engineering/CN=opendap.earthdata.nasa.gov/emailAddress=support@opendap.org':
  Self-signed certificate encountered.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: `fnoc1.nc.dds'

    [ <=> ] 197         --.-K/s   in 0s

2014-11-14 11:22:22 (7.23 MB/s) - `fnoc1.nc.dds' saved [197]

[spooky:olfs/testsuite/urs] ndp% more fnoc1.nc.dds
Dataset {
    Int16 u[time_a = 16][lat = 17][lon = 21];
    Int16 v[time_a = 16][lat = 17][lon = 21];
    Float32 lat[lat = 17];
    Float32 lon[lon = 21];
    Float32 time[time = 16];
} fnoc1.nc;
```

It appears that *wget* correctly followed the first redirect to `urs.earthdata.nasa.gov`, where the EDL server responded with "401 Unauthorized" (thanks to the app_type=401 query parameter in the redirect URL provided by the *origin* server). After getting the 401 *wget* resubmits the request with the authentication credentials and the EDL server accepts them and redirects *wget* back to the *origin* server to complete the request.

2.4. *ncdump*, *nccopy*, *Matlab*, and other applications that use *NetCDF-C*

Applications built with one of UNIDATA's NetCDF libraries may require, in addition to a *~/.netrc* file, a *.dodsrc* file to be present in the users home directory in order for the application to successfully authenticate during remote data access activities. You can learn more about the *.dodsrc* file at UNIDATA's NetCDF and DAP2 page.

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your EDL credentials with the EDL service endpoint.
- Next create (as needed) and then edit the file *~/.dodsrc* so that it tells DAP clients to use the *.netrc* file for password information:

```
HTTP.COOKIEJAR=/home/your_home_dir/cookie_file
HTTP.NETRC=/home/your_home_dir/.netrc
```

Here is a typical *.dodsrc* file.

```
# OPeNDAP client configuration file. See the OPeNDAP
# users guide for information.
USE_CACHE=0
# Cache and object size are given in megabytes (20 ==> 20Mb).
MAX_CACHE_SIZE=20
MAX_CACHED_OBJ=5
IGNORE_EXPIRES=0
CACHE_ROOT=/home/your_home_dir/.dods_cache/
DEFAULT_EXPIRES=1
ALWAYS_VALIDATE=1
# Request servers compress responses if possible?
# 1 (yes) or 0 (false).
DEFLATE=1
# Proxy configuration:
# PROXY_SERVER=<protocol>,<[username:password@]host[:port]>
# NO_PROXY_FOR=<protocol>,<host|domain>
# AIS_DATABASE=<file or="" url="">

# Earth Data Login and LDAP login information
HTTP.COOKIEJAR=/home/your_home_dir/cookie_file
HTTP.NETRC=/home/your_home_dir/.netrc
```

For other NeCDF-C built applications

Check the version of the netCDF C library that the application uses; once they have updated to 4.3.3.1 or later, authentication configuration should be the same as this `ncdump` *example. That is, both EDL and LDAP-backed HTTP/S-Basic authentication should work by reading credentials from the* `.netrc` *file given that the* `.dodsrc` *file is set to point to them.*

NOTE This was tested with the `ncdump` and `nccopy` command line applications that come bundled with the netcdf-c library. This content was developed using NetCDF-4.9.0. Previous versions may not work.

NOTE The online documentation for version netcdf-c-4.8.1 contains instructions written by UNIDATA for configuring authentication. Oddly, the online documentation most for the current netcdf-c release, 4.9.2 at the time of this writing, no longer contains an authentication/authorization discussion.

2.5. *Integrated Data Viewer (IDV)*

We downloaded the latest version of IDV (6.1u2 on 03/Apr/2023) and installed it on our local system.

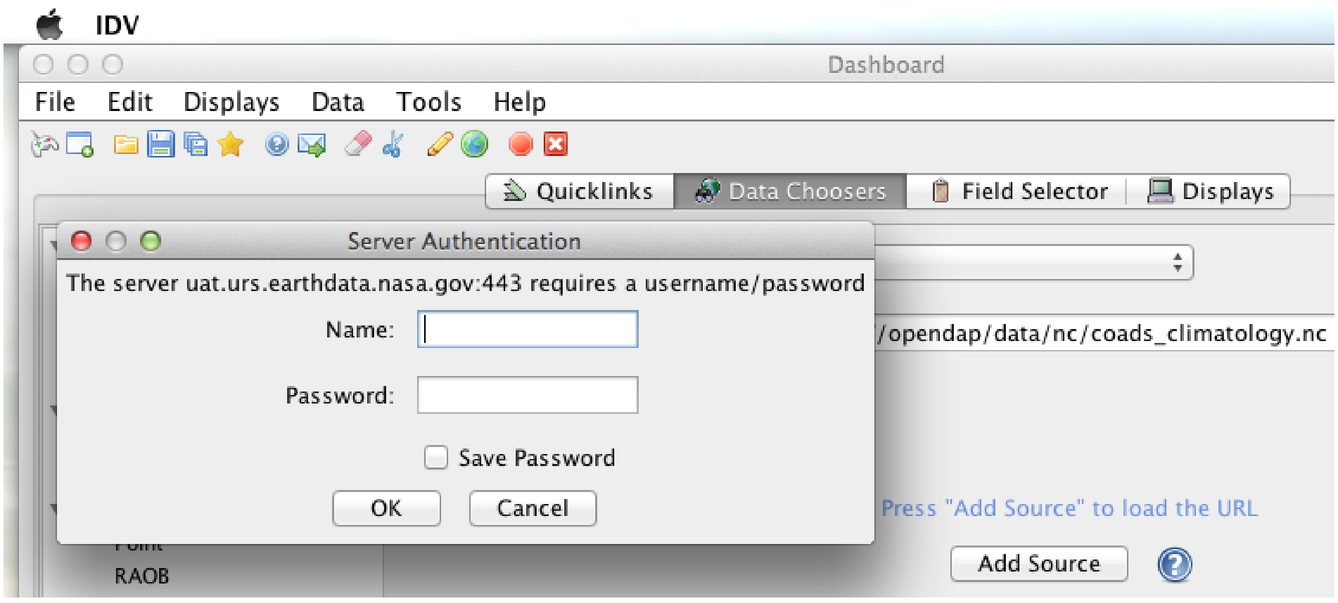
For EDL testing we utilized our AWS test service, configured to require EDL authentication for all access of Hyrax.

In IDV we attempted to choose a new dataset by starting with the "**Data**" menu: **Data** > **Choose Data** > **From A Web Server**

In the resulting pane we entered the AWS test service URL for our friend *coads_climatology.nc*:

https://opendap.earthdata.nasa.gov/hyrax/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) IDV popped up a dialog box that indicated that the *urs.earthdata.nasa.gov* server wanted our credentials:



Server Authentication

The server uat.urs.earthdata.nasa.gov:443 requires a username/password

Name:

Password:

Save Password

We entered our EDL Name and Password, clicked the save password check box, and clicked the *OK* button.

At this point IDV returned a Null Pointer Exception:

```
java.lang.NullPointerException
  at ucar.unidata.data.DataManager.createDataSource(DataManager.java:1623)
  at
ucar.unidata.idv.IntegratedDataViewer.createDataSource(IntegratedDataViewer.java:1978)
  at
ucar.unidata.idv.IntegratedDataViewer.makeDataSource(IntegratedDataViewer.java:1895)
  at
ucar.unidata.idv.IntegratedDataViewer.makeDataSource(IntegratedDataViewer.java:1829)
  at
ucar.unidata.idv.IntegratedDataViewer.handleAction(IntegratedDataViewer.java:1671)
  at ucar.unidata.idv.DefaultIdv.handleAction(DefaultIdv.java:116)
  at
ucar.unidata.idv.IntegratedDataViewer.handleAction(IntegratedDataViewer.java:1603)
  at ucar.unidata.idv.chooser.UrlChooser.loadURLInner(UrlChooser.java:267)
  at ucar.unidata.idv.chooser.UrlChooser.loadURL(UrlChooser.java:239)
  at ucar.unidata.idv.chooser.UrlChooser.doLoadInThread(UrlChooser.java:286)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
  at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
  at java.lang.reflect.Method.invoke(Method.java:498)
  at ucar.unidata.util.Misc$2.run(Misc.java:1215)
  at ucar.unidata.util.Misc$3.run(Misc.java:1243)
```

2.6. *ToolsUI*

We downloaded the latest version of ToolsUI (5.5.3 on 03/Apr/2023) and installed it on our local system. We launched ToolsUI using the command line:

```
java -Xmx1g -jar toolsUI-5.5.3.jar
```

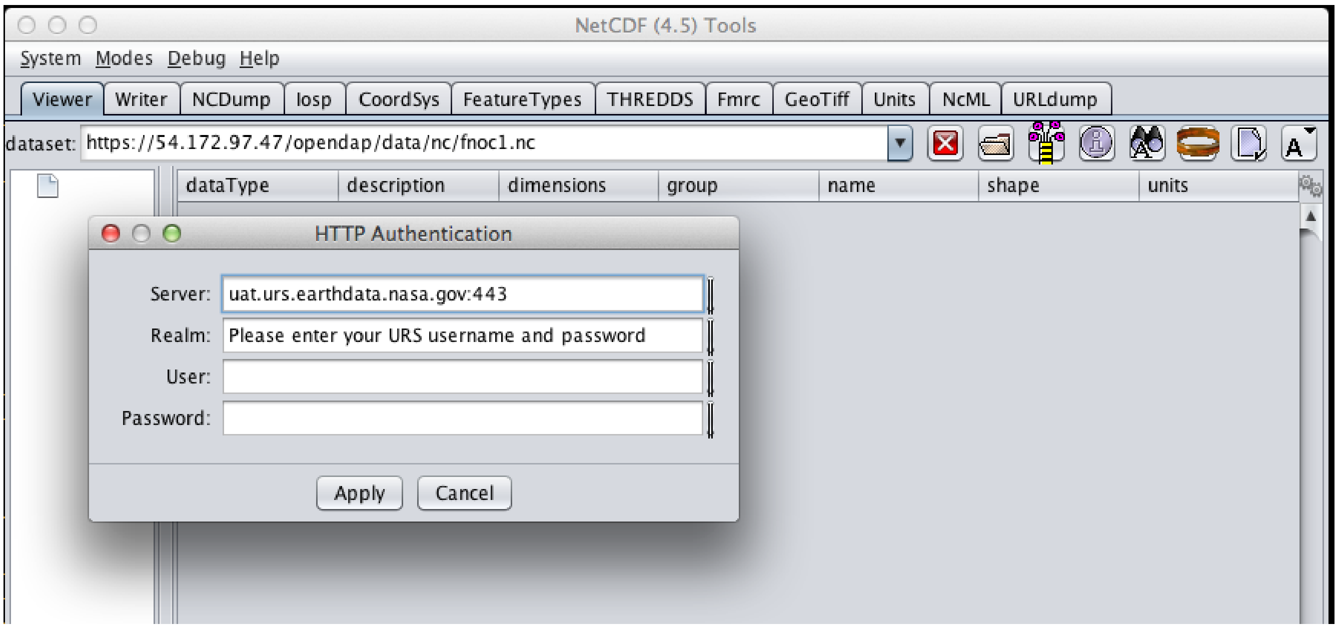
Summary: Authentication FAILED

For testing, we utilized our the NGAP Hyrax service, which requires EDL authentication for all data access.

In ToolsUI we selected the *Viewer* tab, and entered the URL for our friend *fnoc1.nc*:

- https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc

When we committed the edit (aka hit Enter) ToolsUI popped up a dialog box that indicated that the *urs.earthdata.nasa.gov* server wanted our credentials.


NetCDF (4.5) Tools
System Modes Debug Help
Viewer Writer NCDump Iosp CoordSys FeatureTypes THREDDS Fmrc GeoTiff Units NcML URLdump
dataType description dimensions group name shape units
HTTP Authentication
Apply Cancel

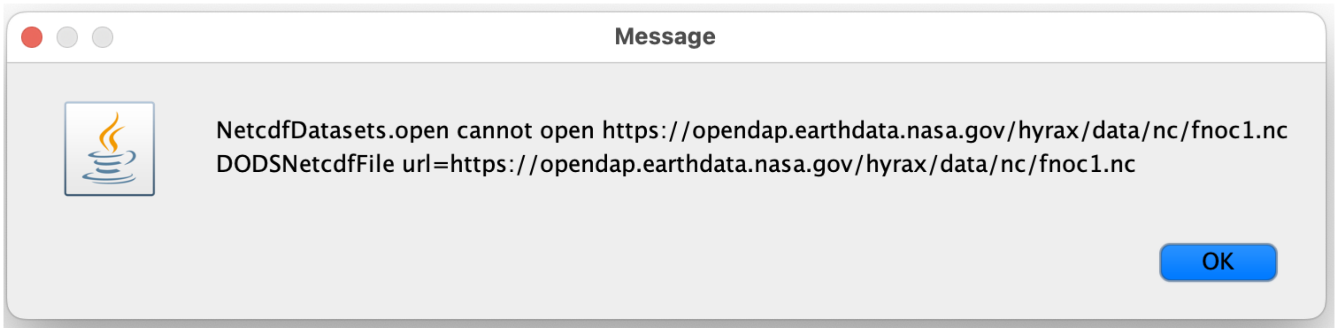
dataset: https://54.172.97.47/opendap/data/nc/fnoc1.nc

Server: uat.urs.earthdata.nasa.gov:443
Realm: Please enter your URS username and password
User:
Password:

We entered our EDL credentials and clicked the *OK* button. After a few seconds

ToolsUI returned an error:

```
java.io.IOException: https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc is not
a valid URL, return status=405
```



NetcdfDatasets.open cannot open https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc
DODSNetcdfFile url=https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc

2.7. *Panoply*

We downloaded the latest version of Panoply (5.2.5 on 04/Apr/2023) and installed it on our local system. We launched Panoply (clicking its icon in our Applications folder)

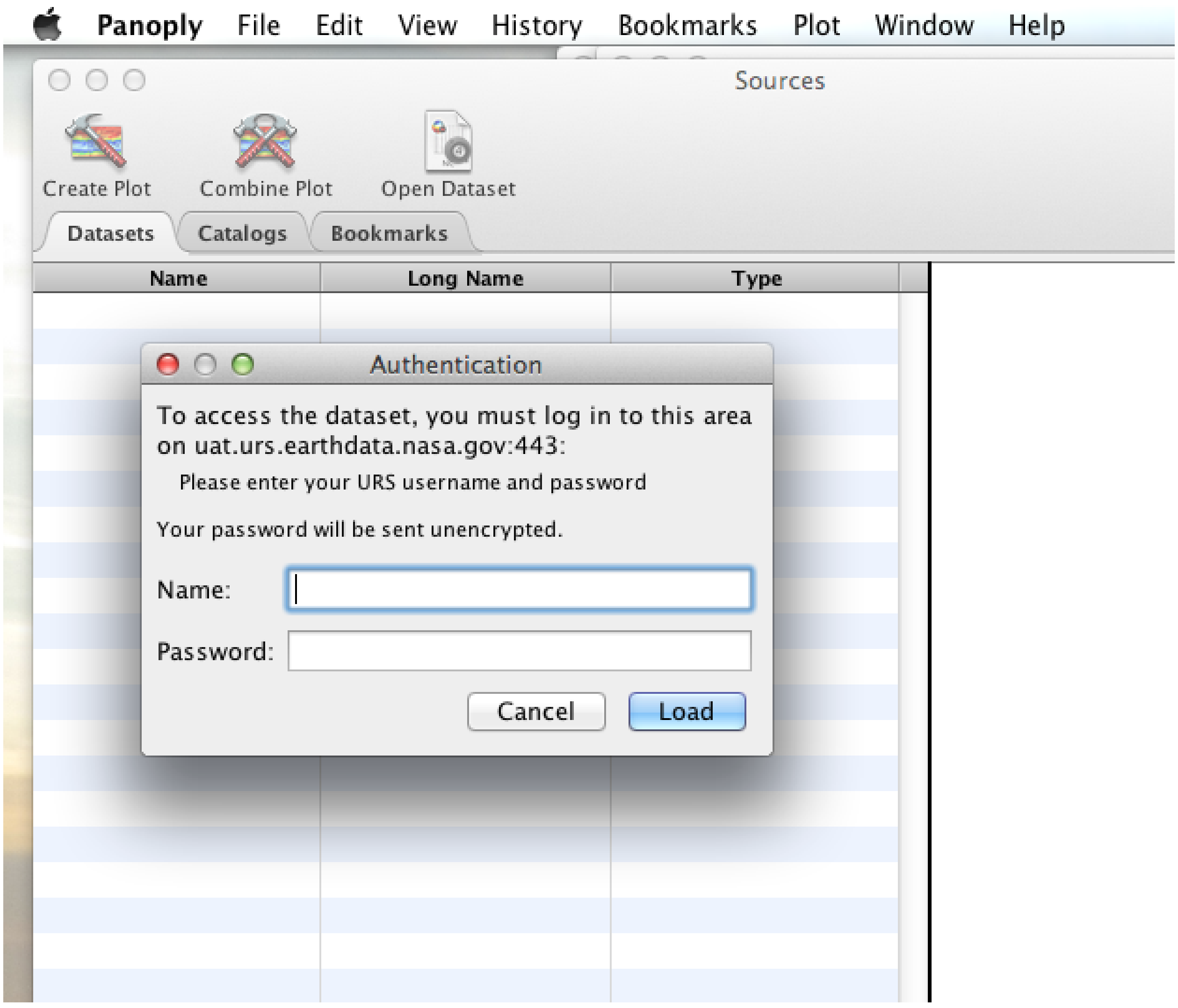
Summary: Authentication FAILED

For testing, we utilized our AWS test service, configured to require EDL authentication for all access of Hyrax.

From the *File* menu, we selected "Open Remote Dataset..." and in the pop dialog we entered the URL for our friend *coads_climatology.nc*:

https://opendap.earthdata.nasa.gov/hyrax/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) Panoply popped up a dialog box that indicated that the *urs.earthdata.nasa.gov* server wanted our credentials.



To access the dataset, you must log in to this area on uat.urs.earthdata.nasa.gov:443:

Please enter your URS username and password

Your password will be sent unencrypted.

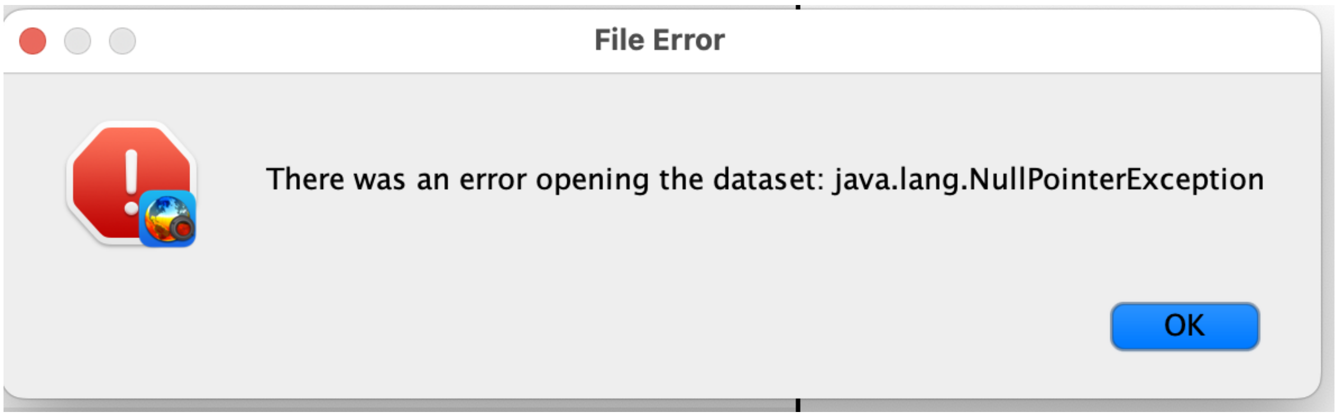
Name:

Password:

Cancel Load

We entered them, clicked the save password check box, and clicked the *OK* button.

Panoply returned a NUll Pointer exception



There was an error opening the dataset: java.lang.NullPointerException

2.8. *PyDAP*

PyDAP includes an extension mechanism so that it can interact with different kinds of authentication systems. This system is very flexible: we were able to use it to add support for both LDAP and EDL using HTTP/S Basic authentication.

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your EDL credentials with the EDL service endpoint.

Once the *.netrc* file is configured, start python, and then acquire data from remote DAP services. Here's a python script that will open a PyDAP virtual connection to an authenticated server if your `~/.netrc` is in order for EDL:

```
# PyDAP uses the request() function and automagically discovers the
# users credentials in ~/.netrc

import pydap

dataset_url="https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc"

pydap_dataset = pydap.client.open_url(dataset_url, protocol="dap4")
```

3. LDAP

The Lightweight Directory Access Protocol (LDAP) can do many things. One of those things is to Single Sign On (SSO) authentication service.

3.1. *cURL* (a.k.a. *lib_curl* or *curl*)

We were able to use command line *curl* to retrieve LDAP authenticated resources using the following technique.

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your credentials with the LDAP service endpoint.

We could then access the top level directory of an LDAP authenticated Hyrax server with the following *curl* command:

```
#!/bin/bash
curl -k -n -c cookie_file -b cookie_file  --url https://some.ldap.tester/opendap
```

What is happening here?

In this request cURL is both authenticating and maintaining a session with the remote server (this is accomplished by telling cURL to load and save cookies from the same file, *cookie_file*)

`-k`

This tells *curl* to accept self-signed certificates. This is ok for working with trusted (as in your own) "test" services but should be removed for working with production systems. Because: Security, Chain-Of-Trust, etc.

`-n`

This tells *curl* to use that *~/.netrc* file we created.

`-c cookie_file`

This tells *curl* to stash cookies in the file *cookie_file*

`-b cookie_file`

This tells *curl* to read cookies from the file *cookie_file*

`--url https://130.56.244.153/opendap`

The desired URL, protected LDAP authentication.

NOTE That the credentials are sent with every request so secure transport is a must if user accounts are to be protected.

3.2. *ncdump*, *nccopy*, *Matlab*, *Ferret*, and other applications that are built on *NetCDF-C*

To configure `nccopy` and `ncdump` (and thus just about every client application that uses netCDF C) for LDAP-back HTTP/S-Basic authentication, follow the same exact procedure as outline above for EDL, except that in the *.netrc* file, use the OpenDAP server's machine name or IP number in place of the EDL authentication site. Here's a summary, with an example:

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your credentials with the LDAP service endpoint.
- Edit (create as needed) the *~/.dodsrc* file so that it tells DAP clients to use the *~/.netrc* file for password information.

```
HTTP.COOKIEJAR=/home/your_home_dir/cookie_file
HTTP.NETRC=/home/your_home_dir/.netrc
```

3.3. *Integrated Data Viewer (IDV)*

For testing, we utilized an ANU/NCI puppet instance configured to require LDAP authentication for all access of Hyrax.

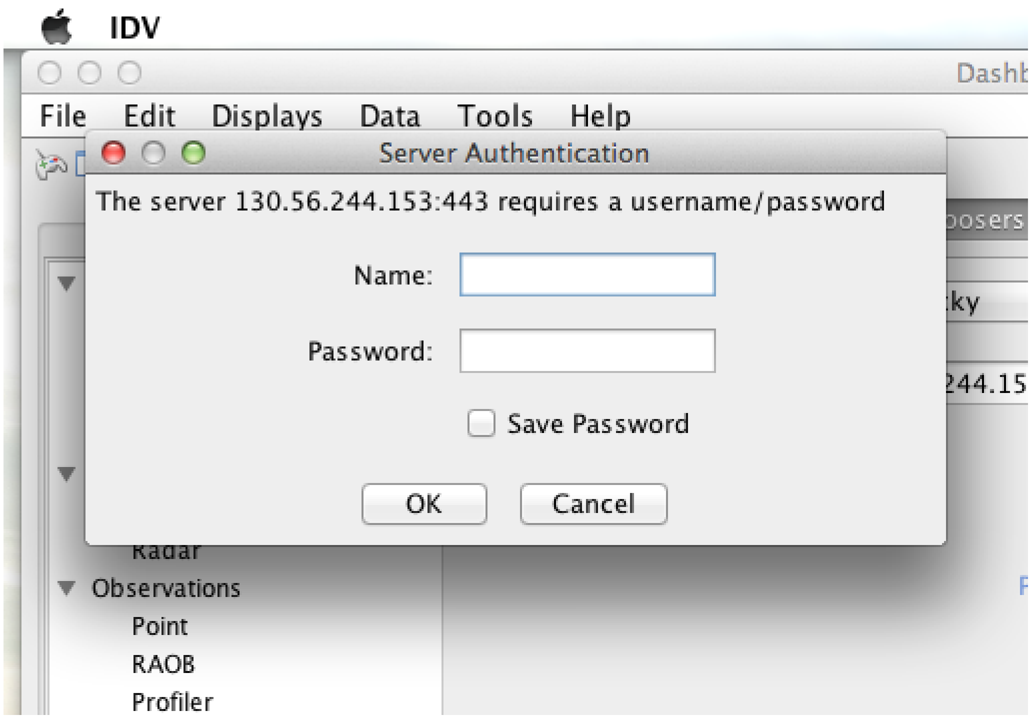
In IDV we attempted to choose a new dataset by starting with the "Data" menu: **Data** > **Choose Data** > **From A Web Server**

In the resulting pane we entered the LDAP test service URL for our friend *coads_climatology.nc*:

https://130.56.244.153/hyrax/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) IDV popped up a dialog box that indicated that the

130.56.244.153 server wanted our credentials:


The server 130.56.244.153:443 requires a username/password

Name:

Password:

Save Password

WE entered them, clicked the save password check box, and clicked the *OK* button. IDV was then able to access the requested resource.

3.4. *ToolsUI*

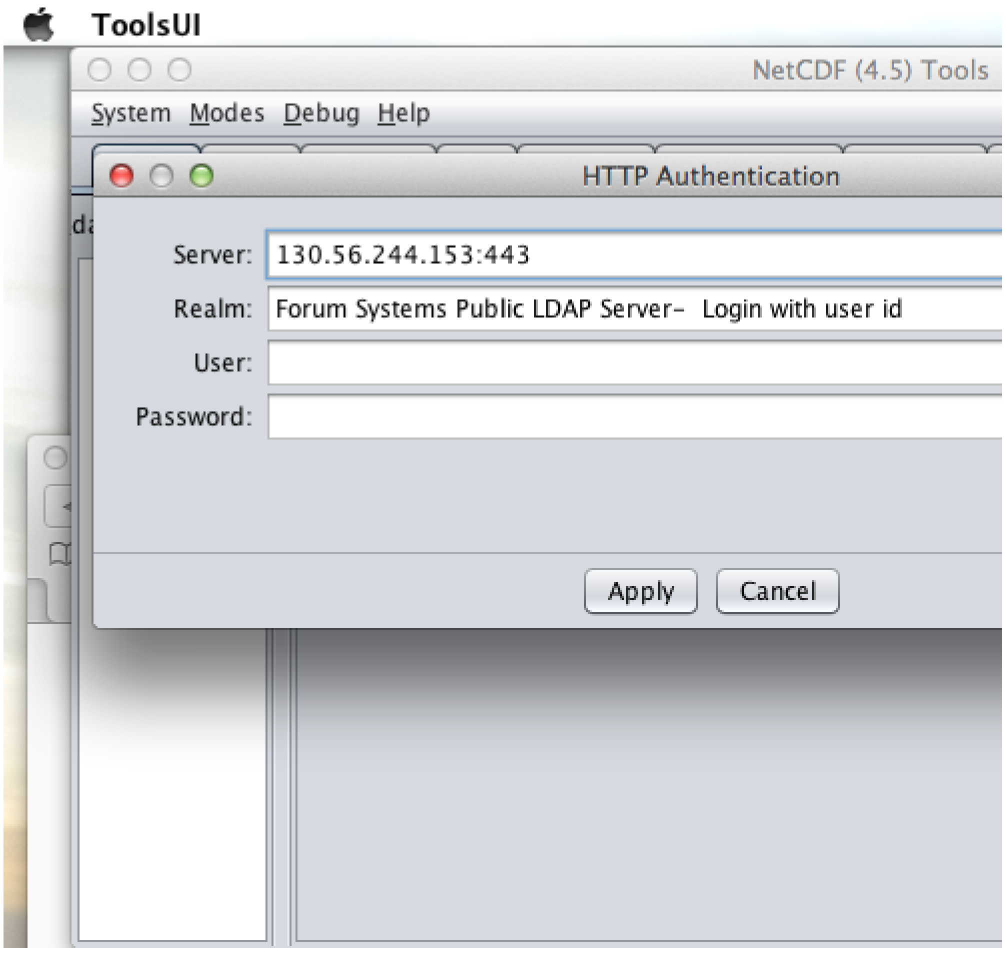
Summary: Authentication Successful

For testing, WE utilized an ANU/NCI puppet instance configured to require LDAP authentication for all access of Hyrax.

In ToolsUI selected the *Viewer* tab, and entered the LDAP test service URL for our friend *coads_climatology.nc*:

https://130.56.244.153/opendap/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) ToolsUI popped up a dialog box that indicated that the *urs.earthdata.nasa.gov* server wanted our credentials.



Server: 130.56.244.153:443

Realm: Forum Systems Public LDAP Server- Login with user id

User:

Password:

We entered them and clicked the *OK* button. ToolsUI was then able to access the requested resource.

3.5. *Panoply*

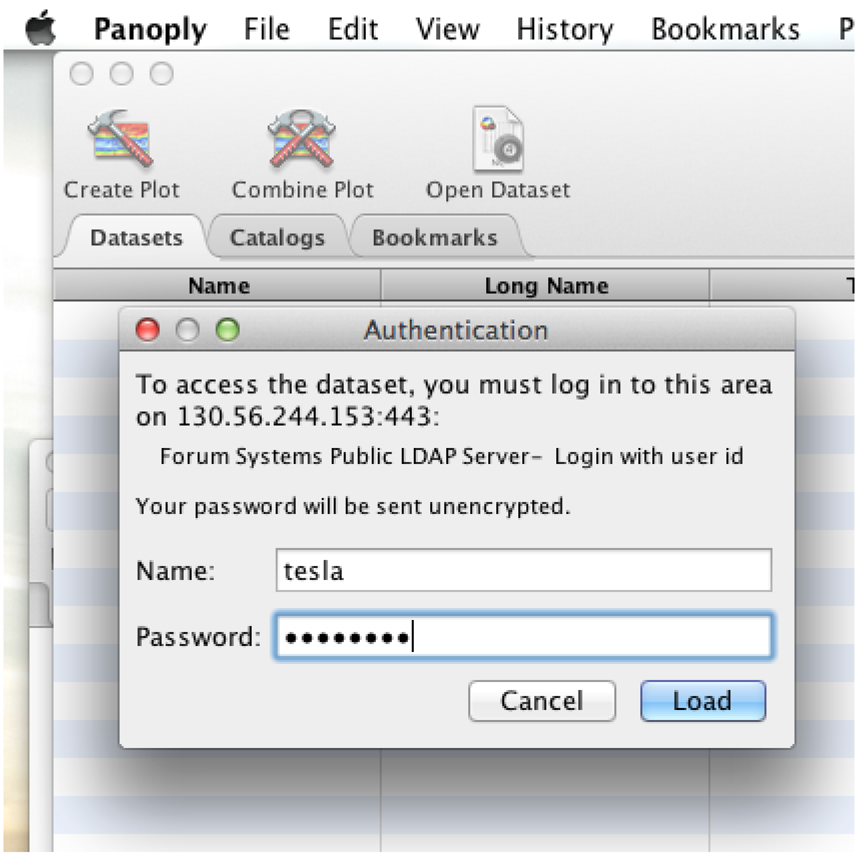
Summary: Authentication Successful

For testing, we utilized an ANU/NCI puppet instance configured to require LDAP authentication for all access of Hyrax.

From the *File* menu, we selected "Open Remote Dataset..." and in the pop dialog we entered the URL for our friend *coads_climatology.nc*:

https://130.56.244.153/opendap/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) Panoply popped up a dialog box that indicated that the *urs.earthdata.nasa.gov* server wanted our credentials.



To access the dataset, you must log in to this area on 130.56.244.153:443:
Forum Systems Public LDAP Server– Login with user id
Your password will be sent unencrypted.
Name: tesla
Password: ●●●●●●●●●

We entered them, clicked the save password check box, and clicked the *OK* button. Panoply was then able to access the requested resource.

3.6. *PyDAP*

PyDAP includes an extension mechanism so that it can interact with different kinds of authentication systems. This system is very flexible: we were able to use it to add support for both LDAP and EDL using HTTP/S Basic authentication. The same scheme could be used to add support for Shibboleth, although it would take additional development work (described in general below).

Summary

- Create a `~/.netrc` file as described in the .netrc section above
- Edit the `~/.netrc` file and associate your EDL credentials with the EDL service endpoint.

Once the *.netrc* file is configured, start python, and then acquire data from remote DAP services. Here's a python script that will open a PyDAP virtual connection to an authenticated server if your `~/.netrc` is in order for EDL:

```
# PyDAP uses the request() function and automagically discovers the
# users credentials in ~/.netrc

import pydap

dataset_url="https://opendap.earthdata.nasa.gov/hyrax/data/nc/fnoc1.nc"

pydap_dataset = pydap.client.open_url(dataset_url, protocol="dap4")
```

4. Shibboleth

Shibboleth is a collection of open source tools that provide identity management (aka authentication services) has fairly broad adoption globally. It is in use at a number of UK and Australian institutions.

4.1. *cURL* (a.k.a. *lib_curl* or *curl*)

We were not able to use command line *curl* to retrieve Shibboleth authentication resources using the *.netrc* technique described in the LDAP and EDL sections.

Analysis of the HTTP conversation between the idp.testshib.org server and *curl* shows that curl correctly follows the series of 302 redirects issued to it, first by the Apache service bound to the Hyrax server and then from the idp.testshib.org server. In every request to the idp.testshib.org server the *curl* client correctly offers the credentials via the HTTP Authorization header:

```
0000: GET /idp/Authn/UserPassword HTTP/1.1
0026: Authorization: Basic bXlzZWxmOm15c2VsZg==
0051: User-Agent: curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.2
0091: 1.4 OpenSSL/0.9.8z zlib/1.2.5
00b0: Host: idp.testshib.org
00c8: Accept: */*
00d5: Cookie: _idp_authn_lc_key=efbb6e2a9d893b47fb802ed575329ce69c101b
0115: 3ea8beb6744fab64fc406c358f; JSESSIONID=5A1731EDE00613B13803968CF
0155: AF06284
015e:
```

But the Shibboleth system doesn't respond to them. This may be a simple configuration issue on the Shibboleth end, or it could be that the Shibboleth protocol specifically forbids accepting credentials via HTTP Authorization headers.

4.2. *ncdump* and *nccopy*

At the time of this writing the *ncdump* application and the NetCDF library do not support authentication using the Shibboleth ECP profile.

4.3. *Integrated Data Viewer (IDV)*

Summary: Failed To Authenticate

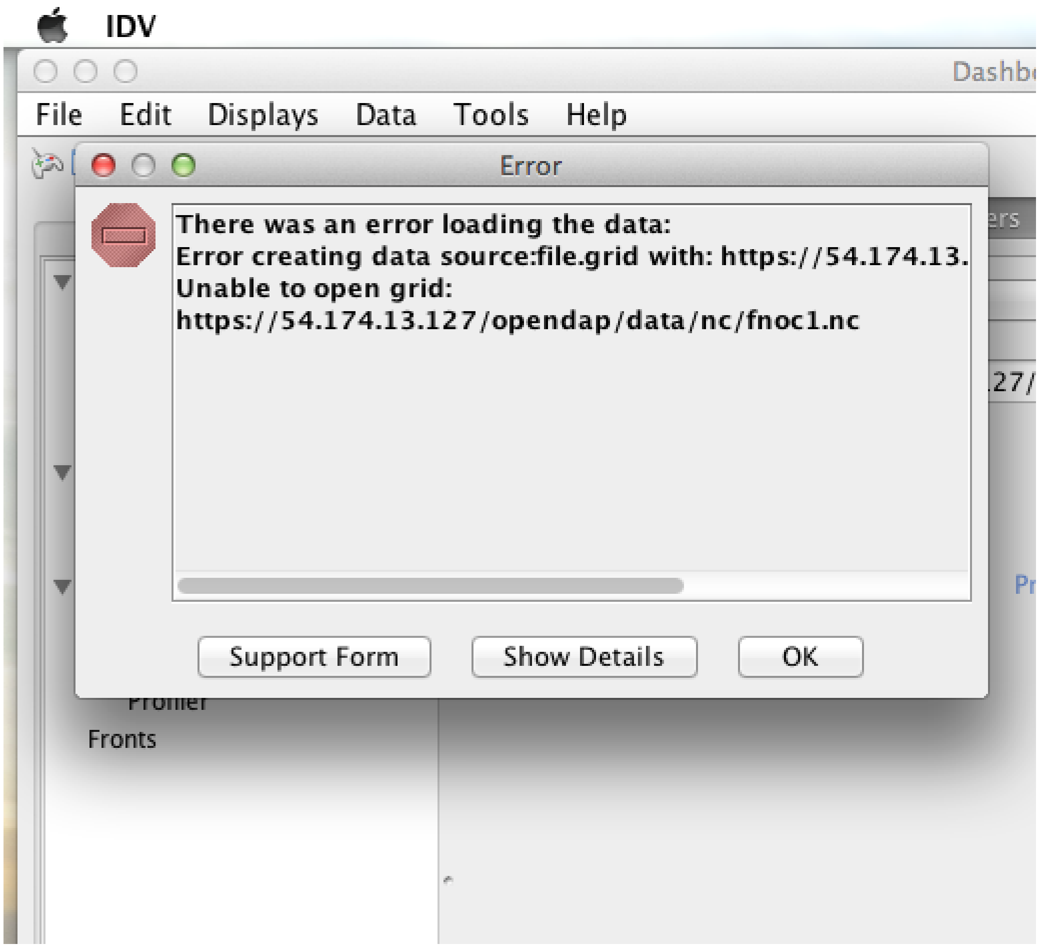
For Shibboleth testing we utilized an AWS VM, configured to require Shibboleth authentication for all access of Hyrax.

In IDV we attempted to choose a new dataset by starting with the "Data" menu: Data > Choose Data > From A Web Server

In the resulting pane we entered the AWS VM service URL for our friend *coads_climatology.nc*:

https://54.174.13.127/opendap/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) IDV popped up a dialog box that indicated that there was an error loading the data:



```
There was an error loading the data:
Error creating data source:file.grid with: https://54.174.13.
Unable to open grid:
https://54.174.13.127/opendap/data/nc/fnoc1.nc
```

4.4. *ToolsUI*

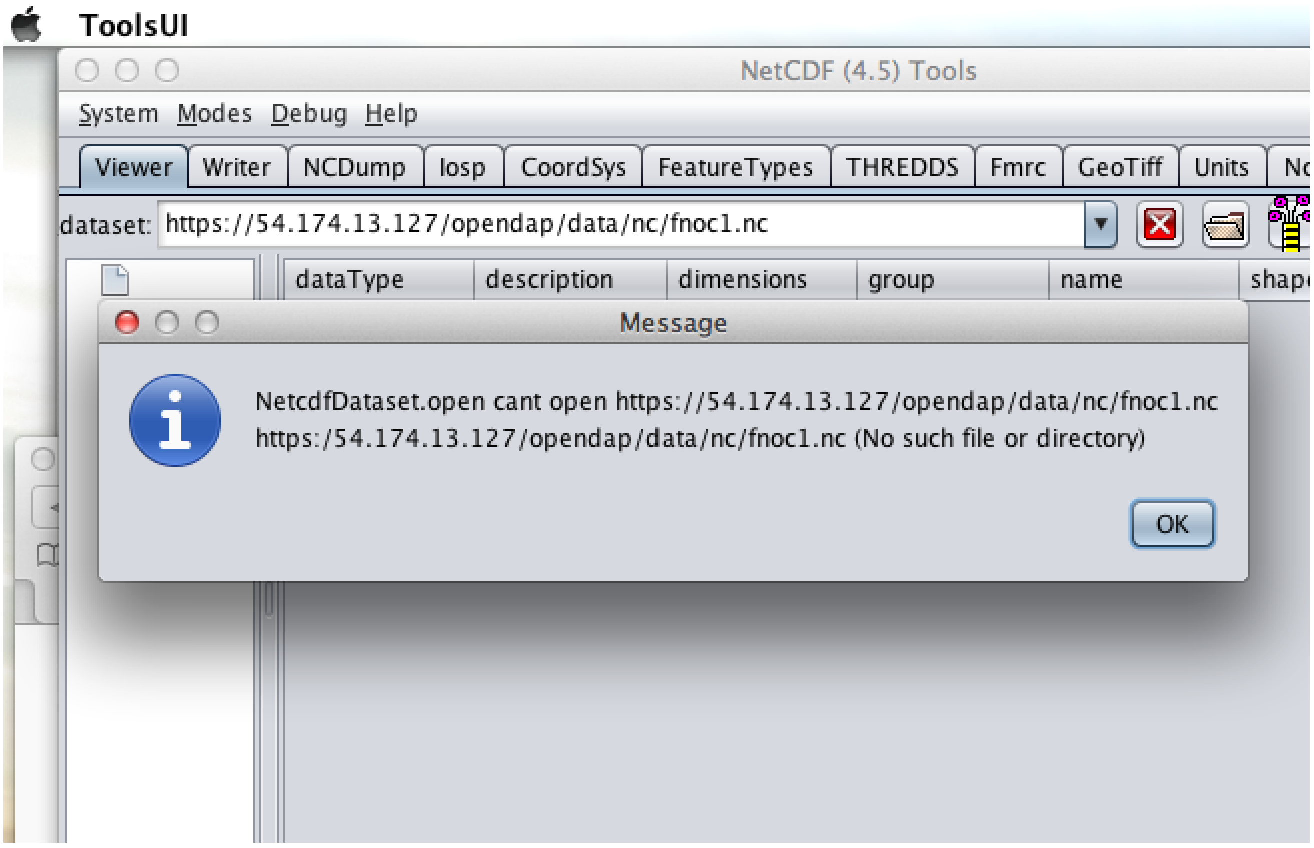
Summary: Failed To Authenticate

For Shibboleth testing we utilized an AWS VM, configured to require Shibboleth authentication for all access of Hyrax.

In ToolsUI selected the *Viewer* tab, and entered the AWS test service URL for our friend *coads_climatology.nc*:

https://54.174.13.127/opendap/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) ToolsUI popped up a dialog box that indicated that there was an error loading the data:


ToolsUI
NetCDF (4.5) Tools
System Modes Debug Help
Viewer Writer NCDump Iosp CoordSys FeatureTypes THREDDS Fmrc GeoTiff Units Nc
dataset:
dataType description dimensions group name shap
Message
OK

dataset: https://54.174.13.127/opendap/data/nc/fnoc1.nc

NetcdfDataset.open cant open https://54.174.13.127/opendap/data/nc/fnoc1.nc
https:/54.174.13.127/opendap/data/nc/fnoc1.nc (No such file or directory)

4.5. *Panoply*

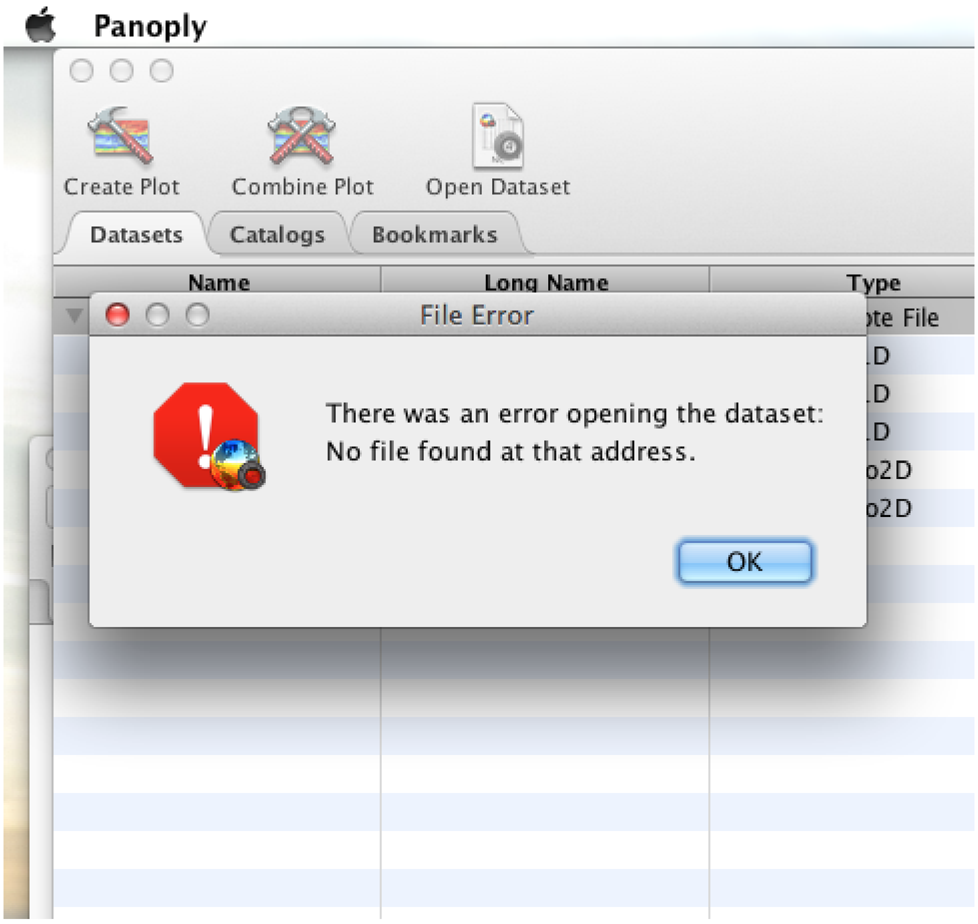
Summary: Failed To Authenticate

For Shibboleth testing we utilized an AWS VM, configured to require Shibboleth authentication for all access of Hyrax.

From the *File* menu, we selected "Open Remote Dataset..." and in the pop dialog we entered the URL for our friend *coads_climatology.nc*:

https://130.56.244.153/opendap/data/nc/coads_climatology.nc

When we committed the edit (aka hit Enter) Panoply popped up a dialog box that indicated that there was an error loading the data:



There was an error opening the dataset:
No file found at that address.

4.6. *Matlab, Ferret, Other applications that use NetCDF C*

This is certain to not work until the netCDF C library is modified to explicitly support it.

4.7. *PyDAP*

This will require a new patch function, similar to *install_basic_client()* be written. It will be a bit more complex because of the increased complexity of Shibboleth, but the operation for end-users will likely be the same.