## 1. *Build a root catalog.*

The goal of the catalog cleaner is to build a catalog hierarchy which contains data sets which meet all of the UAF criteria for quality and usability. To build such a catalog, the process starts with a single root catalodg THREDDS URL. The root catalog does not have to be on a public server. The base URL of the resulting “clean” catalog can be specified in the final step of preparing the “clean” catalog. The cleaner will descend through the entire hierarchy of the THREDDS catalog starting at the root so ideally, the root catalog should contain only data and sub-catalogs you believe are good candidates to appear in your final “clean” catalog. However, you can exclude sub-catalogs from the being treated by the catalog cleaner through configuration at any point in the process.

Once you have the catalog prepared, you can install the catalog clearer software.

## 2. *Install the Catalog Cleaner software.*

1. Expand the tar file into a directory on your machine. The resulting directory will have

   bin lib resources

   directories.

2. Set up your environment. You must set your JAVA_HOME environment variable and set a new variable called CLEANER_HOME which points to the directory that contains the bin, lib and resources directory of your catalog cleaner installation.

3. Edit the datanucleus.properties files. In the resources directory is a file called datanucleus.properties. It looks like this:

   datanucleus.ConnectionDriverName=com.mysql.jdbc.Driver
   datanucleus.query.jdoql.allowAll=true
   datanucleus.ConnectionURL=jdbc\:mysql\://127.0.0.1\:3306/database
   datanucleus.ConnectionUserName=**dbuser**
   datanucleus.ConnectionPassword=**password**
   datanucleus.identifier.case=LowerCase
   datanucleus.autoCreateSchema=true
   datanucleus.autoCreateTables=true
   datanucleus.autoCreateColumns=true
   datanucleus.validateTables=true
   datanucleus.NontransactionalRead=true

javax.jdo.option.Multithreaded=true

You need to supply a database user and password combination that has permissions to update databases in your mySQL instance. The rest of the parameters should be left unchanged.

4. Edit the /etc/my.cnf to increase the max_allowed_packet to 32M. Under the [mysqld] section add or modify the max_allowed_packet like this:

```
[mysqld]
max_allowed_packet=32M
```

Because the catalog cleaner slings around some rather large blobs of text data when processing catalogs you have to let mySQL know to expect the large chunks of data.

## 3. *Create a new database.*

Use the mysql command to create a new database to keep track of the information retrieved by the catalog cleaner. I use the convention cc_YYYYMMDD to name my database. Here is an example:

```
[rhs@dunkel ~]$ mysql -u root -h localhost -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 872
Server version: 5.0.95 Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database cc_20130913;
Query OK, 1 row affected (0.07 sec)
```

## 4. *Run the tree scan on the root catalog.*

You are now ready to start collecting data from the catalog storing in the local data store.

The first step is what we call a “tree scan”.  In this step the catalog cleaner runs through the entire catalog hierarchy and collects the XML of each catalog and records the URLs of any data sets and sub-catalogs in each catalog.

To run the catalog cleaner tree scan, from your catalog cleaner home directory run the following command:

```
./bin/treecrawler.sh -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -d cc_20130913 > & tree.log
```

where -r is the URL of the root catalog you built in step 1 and -d is the name of the database you created in step 3.

The -r option is always the specifies the root catalog of where you want to start the processing for whatever operation you are performing.  Generally, it is the URL of the raw catalog you built in step 1, but you can start a data scan a some catalog below the root in the tree by using the parent of the catalog you want to clean as the root (-r) and including the catalog you want to process in the -u option.

Because the process dump out lots of diagnostic information we recommend that you capture the output and errors from the command into a log file (combining them according to the right technique for your preferred shell script).

**Note:** If for some reason you have a sub-catalog in the hierarchy that you do not want to scan, you can exclude it from the processing done by the catalog cleaner.

In the resources directory you will find a skip.xml file.  Its contents look something like this:

```
<?xml version="1.0"?>
<skips>
   <regex value=".*las.*"/>
   <regex value=".*examples.*"/>
   <regex value=".*Clean.*"/>

    <catalog url="http://www.ngdc.noaa.gov/thredds/samplesCatalog.xml"/>
   <catalog url="http://cwcgom.aoml.noaa.gov/thredds/catalog/testBUFR/catalog.xml"/>
</skips>
```

The first set of elements (regex) define regular expressions in the value attribute.  Any catalog URL that matches the regular expression will be skipped.  The syntax is the same as for any Java regular expression.

The second type of element (catalog) is simply a list of catalogs to skip.

## 5. *Run the data crawl.*

The next part of the process is what we call a “data crawl”. This is the most time consuming and intensive part of the process. The catalog cleaner code will open each and every OPeNDAP URL it finds in the catalog hierarchy and collect information about the contents of the data source.
Because this part takes a long time to run and because sometimes it is convenient to be able to collect data from a portion of the catalog, there are two different ways to collect run a data crawl and collect the necessary data.

1. The first method is to simply launch the crawl on the entire using a command similar to this:

   ./bin/datacrawler.sh -d cc_20130913 -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml > & data.log

2. Sometimes it is convenient to just collect the data for a particular sub-catalog. You (see -f and update) do this by naming the root and the sub-catalog you want to scan on the command line. The “root” you name can actually be any catalog in the hierarchy and the sub-catalog any direct child catalog of the named root.

3. For example:

   ./bin/datacrawler.sh -d cc_20130913 -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -u http://data.nodc.noaa.gov/thredds/catalog/woa/gin_seas/catalog.xml > & data_nodc.log

   To make this second method easier, there is a utility you can run that will create a shell script for each child catalog of the root. You can then use these scripts to scan each sub-catalog in turn. In fact, this is how I always run the cleaner.

   ./bin/scripts.sh -d cc_20130913 -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml

   The resulting scripts will be in the CleanScripts directory and you should run them from the catalog cleaner root, like this:

   ./CleanScripts/data.nodc.noaa.gov0/datatop.sh

**Note**: When the data scan runs, it only looks at OPeNDAP URLs for which it has not already collected data. If you want to force the data crawler to re-open and re-collect data for all of the OPeNDAP URLs in a catalog use the -f (--force) option on the command line.
**Note:** The scripts (datatop.sh) generated by the process above start at the top of each child of the root catalog and have the -f option enabled by default.

**Note:** You can list the into a file and use that as shell script to invoke the data crawl on each sub-catalog in turn. I do this when collecting data because it's easier to see where you are in the process and to restart a particular point if something goes wrong like a network outage.

## 6. *Generate reports.*

There are three reports you can run which give increasing details about the state of the information collected and stored locally.

**Brief:** This report give the total number of leaf data sets for each of the catalogs that is referenced that the top level of the raw master catalog. For each of the catalogs, the report gives the total number of leaf data sets found in the catalog and it's children, the number that have been scanned (the process described in the previous step (5. Run the data crawl)), the number that have not been scanned and the number that have been scanned, but where no suitable variables were found in the data set. By scanned we mean, OPeNDAP URLs that have opened via netCDF API calls and had their metadata stored locally. A failed scan can happen for many reasons. You can generate a more detailed report to see what is known about why the scanning of a URL failed.

Details in a special section at the bottom of the report highlights any catalog that are missing from the data store even though they are referenced in the raw master catalog which might mean they were off .

<u>Sample Report Output for the Brief Report (use -b or --brief)</u>

```
./bin/report.sh -b -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -d cc_20130913

Report generated at 2013-09-17 07:33:03 for cc_20130913.
Root has 0 OPeNDAP datasets with:
        0 finished.
```

0 not started.
0 failed.
0 had no variables.
20 sub-catalogs.

http://cwcgom.aoml.noaa.gov/thredds/catalog.xml has:

leaf data sets = 31 with

28 scanned.
0 not scanned.
0 failed.
3 had no variables.

http://www.esrl.noaa.gov/psd/thredds/catalog/Datasets/catalog.xml has:

leaf data sets = 4022 with

4020 scanned.
0 not scanned.
0 failed.
2 had no variables.

http://ferret.pmel.noaa.gov/thredds/carbontracker.xml has:

leaf data sets = 1 with

1 scanned.
0 not scanned.
0 failed.
0 had no variables.

http://data1.gfdl.noaa.gov:8380/thredds3/catalog.xml has:

leaf data sets = 0 with

0 scanned.
0 not scanned.
0 failed.
0 had no variables.

http://osmc.noaa.gov:8180/thredds/catalog.xml has:

leaf data sets = 11 with

11 scanned.
0 not scanned.
0 failed.
0 had no variables.

http://ferret.pmel.noaa.gov/thredds/uaf.xml has:

leaf data sets = 4 with

4 scanned.
0 not scanned.
0 failed.
0 had no variables.

http://ecowatch.ncddc.noaa.gov/thredds/oceanNomads/catalog_aggs.xml has:

leaf data sets = 16 with

16 scanned.
0 not scanned.
0 failed.
0 had no variables.

http://www.ngdc.noaa.gov/thredds/catalog.xml has:

leaf data sets = 163 with

162 scanned.
0 not scanned.

0 failed.
1 had no variables.
http://data.nodc.noaa.gov/thredds/catalog/testdata/ghrsst_ncml/fullAgg/catalog.xml has:
leaf data sets = 27 with
24 scanned.
0 not scanned.
0 failed.
3 had no variables.
http://data.nodc.noaa.gov/thredds/catalog/woa/gin_seas/catalog.xml has:
leaf data sets = 102 with
102 scanned.
0 not scanned.
0 failed.
0 had no variables.
http://oceanwatch.pfeg.noaa.gov/thredds/catalog.xml has:
leaf data sets = 546 with
481 scanned.
0 not scanned.
0 failed.
65 had no variables.
http://edac-dap3.northerngulfinstitute.org/thredds/catalog/ncom_fukushima_agg/catalog.xml has:
leaf data sets = 1 with
1 scanned.
0 not scanned.
0 failed.
0 had no variables.
http://edac-dap3.northerngulfinstitute.org/thredds/catalog/ncom_sendai_agg/catalog.xml has:
leaf data sets = 1 with
1 scanned.
0 not scanned.
0 failed.
0 had no variables.
http://colossus.dl.stevens-tech.edu:8080/thredds/bight_dailyrun_fmrc.xml has:
leaf data sets = 2 with
1 scanned.
0 not scanned.
0 failed.
1 had no variables.
http://colossus.dl.stevens-tech.edu:8080/thredds/apex_dailyrun_fmrc.xml has:
leaf data sets = 4 with
1 scanned.
0 not scanned.
0 failed.
3 had no variables.
http://tds.marine.rutgers.edu/thredds/roms/espresso/catalog.xml has:
leaf data sets = 49 with
21 scanned.
0 not scanned.
0 failed.
28 had no variables.

```
        http://geoport.whoi.edu/thredds/COAWST_catalog.xml has:
                leaf data sets = 4 with
                        4 scanned.
                        0 not scanned.
                        0 failed.
                        0 had no variables.
        http://oos.soest.hawaii.edu/thredds/idd/ocn_mod.xml has:
                leaf data sets = 22 with
                        12 scanned.
                        0 not scanned.
                        0 failed.
                        10 had no variables.
        http://michigan.glin.net:8080/thredds/glcfs_nowcast_catalog.xml has:
                leaf data sets = 10 with
                        10 scanned.
                        0 not scanned.
                        0 failed.
                        0 had no variables.
        http://michigan.glin.net:8080/thredds/featurecollection.xml has:
                leaf data sets = 6 with
                        6 scanned.
                        0 not scanned.
                        0 failed.
                        0 had no variables.
Total leaf data sets = 5022 with:
        4900 scanned.
        0 not scanned.
        0 failed.
        116 had no variables.

The following catalogs are listed in the master catalog, but their XML contents is null in the data store.
http://ecowatch.ncddc.noaa.gov/thredds/catalog/ncom_us_east_before_depth_change_agg/catalog.xml
http://ecowatch.ncddc.noaa.gov/thredds/catalog/ncom_useast_agg_20091119_20130404/catalog.xml
http://ecowatch.ncddc.noaa.gov/thredds/catalog/ncom_us_east_agg/catalog.xml
http://oceanwatch.pfeg.noaa.gov/thredds/PaCOOS/GLOBEC/catalog.xml
```

**Normal:** This report gives a similar report as the --brief option however, rather than accumulating the information and reporting the results only for the direct children of the root catalog, this report gives a summary for each and every catalog in the entire catalog.

<u>Sample Report (no --brief or --varcheck option specified)</u>

```
./bin/report.sh -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -d cc_20130913
Report for http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml
Report generated at 2013-09-17 08:02:45 for cc_20130913.
Root has 0 OPeNDAP datasets with:
        0 finished.
        0 not started.
```

0 failed.
0 had no variables.
20 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/catalog.xml has:
9 OPeNDAP datasets with 9 finished 0 not started 0 failed and 0 had no variables.
13 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/IOOSTCHP.xml has:
3 OPeNDAP datasets with 0 finished 0 not started 0 failed and 3 had no variables.
1 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/ROMSMETEO.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/MCI.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/METEOSAT.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/VIBRIO.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/MODIS.xml has:
8 OPeNDAP datasets with 8 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/MODIS_CDOM.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/ALTIMETRY_GOM.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/ALTIMETRY_GOM_GRAD.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/OISST.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/CWBLENDED.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/CWBLENDEDNIGHT.xml has:
1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
0 sub-catalogs.
http://cwcgom.aoml.noaa.gov/thredds/DYNT.xml has:

1 OPeNDAP datasets with 1 finished 0 not started 0 failed and 0 had no variables.
[ … and so on for many more lines …]

**Verbose:** This report give the most detail. If catalog cleaner encountered a problem with an OPeNDAP dataset the URL and the reason for the failure will be included with the report. This report is obtained using the -v or --varcheck option:

./bin/report.sh -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -v -d cc_20130913

Examples of common errors are given below.

OPeNDAP data sets often fail because there is a problem with the conventions metadata making it impossible to recognize how the data are organized in relation to time and space. Any data source in this category will result in an error that looks like this:

http://cwcgom.aoml.noaa.gov/thredds/IOOSTCHP.xml has:
3 OPeNDAP datasets with 0 finished 0 not started 0 failed and 3 had no variables.
1 sub-catalogs.
Dataset http://cwcgom.aoml.noaa.gov/thredds/dodsC/IOOSNETCDF/RITA/PROFILES.nc was scanned, but no variables where found. Scanning threw an unexpected error. Unable to find any grids in this data set using the CF conventions.
Dataset http://cwcgom.aoml.noaa.gov/thredds/dodsC/IOOSNETCDF/EMILY/PROFILES.nc was scanned, but no variables where found. Scanning threw an unexpected error. Unable to find any grids in this data set using the CF conventions.
Dataset http://cwcgom.aoml.noaa.gov/thredds/dodsC/IOOSNETCDF/WILMA/PROFILES.nc was scanned, but no variables where found. Scanning threw an unexpected error. Unable to find any grids in this data set using the CF conventions.

If an axis is not monotonic then it does not follow the CF conventions. This most commonly occurs with the time axis, so we explicitly check to see whether the time axis is monotonic and report the problem in the verbose report as follows:

http://data.nodc.noaa.gov/thredds/catalog/testdata/ghrsst_ncml/fullAgg/catalog.xml has:
27 OPeNDAP datasets with 24 finished 0 not started 0 failed and 3 had no variables.
0 sub-catalogs.
Dataset http://data.nodc.noaa.gov/thredds/dodsC/testdata/ghrsst_ncml/fullAgg/aggregate__ghrsst_L4_NCAMERICA_JPL_MUR.ncml failed with the following errors.
Variable analysed sea surface temperature failed. Time axis is not monotonic at 6 with values 8.82576E8 at 5 and 8.797248E8 at 6
Variable estimated error standard deviation of analysed_sst failed. Time axis is not monotonic at 6 with values 8.82576E8 at 5 and 8.797248E8 at 6
Variable sea/land field composite mask failed. Time axis is not monotonic at 6 with values 8.82576E8 at 5 and 8.797248E8 at 6
Variable sea ice area fraction failed. Time axis is not monotonic at 6 with values 8.82576E8 at 5 and 8.797248E8 at 6

Dataset http://data.nodc.noaa.gov/thredds/dodsC/testdata/ghrsst_ncml/fullAgg/aggregate__ghrsst_L4_GLOB_REMSS_mw_ir_OI_v2.ncml was scanned, but no variables where found. Scanning threw an unexpected error. Unable to find any grids in this data set using the CF conventions.

Dataset http://data.nodc.noaa.gov/thredds/dodsC/testdata/ghrsst_ncml/fullAgg/aggregate__ghrsst_L3P_NAR_AVHRR_METOP_A_EUR.ncml was scanned, but no variables where found. Scanning threw an unexpected error. Unable to find any grids in this data set using the CF conventions.

## 7. *Run the cleaner*

Once you have collected the data the final step is to produce a collection of "clean" catalogs. This process does not read the remote catalogs or OPeNDAP data sources. It relies entirely on the information in the local data store that was collected in 4 and 5.

The command line options for this part of the process are the most complicated because you have to give information about the THREDDS server from which the resulting clean catalogs will be served.

1. You must supply the TDS context name. The default is /thredds and this value will be fine for most installations. However, if you are running more than one THREDDS server in the same tomcat you will need to supply the context name. For example, the UAF catalog (http://ferret.pmel.noaa.gov/**geoide**/geoIDECleanCatalog.html) uses the context "geoide". The command line option to specify the context is -c.
2. You must supply the root base URL and port number where the server will run. For my test server, I use: -s http://dunkel.pmel.noaa.gov:8930/. The UAF catalog uses -s http://ferret.pmel.noaa.gov/.

For example:

./bin/clean.sh -s http://ferret.pmel.noaa.gov/ -c geoide -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -d cc_20130913

The result of this step will be a root catalog which is an edited version of the catalog you created in the first step. The edited catalog will include all of the services expected for a UAF catalog and will have appropriate references to the "cleaned" sub-catalogs. This file is called CleanCatalog.xml

For each sub-catalog a new catalog is written under the CleanCatalogs directory which contains all of the services required for a UAF catalog. The services are either the original remote services if they were available from the original server or local version of the service that will operate on the data via OPeNDAP URL. Any data that was not

aggregated, but should have been will be aggregated in the clean catalog. However, users can expect relatively poor performance data aggregated in this way so you should work to get those data aggregated on their original server. The new catalog will also include viewer links to all of the UAF data viewers. Finally, the catalog will include many properties that can be interpreted by the LAS configuration generating program (call addXML). Therefore, it's quite easy to generate an LAS configuration for the resulting catalog even if it contains thousands of OPeNDAP URLs.

8. Install the catalog.

   If you want to serve the "clean" catalog you need to copy the resulting CleanCatalog.xml file and the entire CleanCatalogs directory hierarchy to your thredds/content directory. Then add CleanCatalog.xml to your default catalog with a <catalogRef> element.

   You can also replace your catalog.xml file at the top of your catalog hierarchy with CleanCatalog.xml

## *9. Update time axis.*

Many data sets are being updated regularly. The most frequent type of update is to append new data along the time axis (record dimension). In order to keep the metadata that has been promoted into the clean catalog (and the resulting LAS controls if you are using LAS with your clean catalog) you'll need to run the update process. We run the process nightly in a cron job for the production UAF catalog.

The update process looks among the data sets in the local data store and identifies those data sets with a time step that ends in the current year. Those are the data sets that have the greatest likelihood of having new time steps appended. The update process then reopens those data sets and collects the time axis information and storing any changes into the local data store.

The syntax for doing an update is:

```
./bin/updatecrawler.sh -r http://dunkel.pmel.noaa.gov:8920/thredds/catalog.xml -d
cc_20130913
```

## *10. Configure LAS (this feature will be available in LAS 8.1).*

The catalog cleaner adds a slew of properties to the resulting catalogs that allows the LAS configuration tools to build LAS configuration without reopening the OPeNDAP data sets. If you want to install an LAS that includes the data from your clean catalog the command is:

```
$LAS_HOME/bin/addXML.sh -m -c -v -t
http://dunkel.pmel.noaa.gov:8930/thredds/CleanCatalog.xml clean_las_config
```

The -m option triggers the use of the catalog cleaner metadata in the clean catalog.