# Advanced ways to find and retrieve data in the LTA

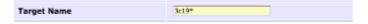
There are some useful ways to find and retrieve your data in the LTA that might not be immediately obvious. This page explains some of the more advanced options you have.

## **Queries**

You can use colons in numeric queries, to select ranges. This will for example give all
observations and pipelines that have a SAS/Observation ID in the range from 432000 to
432190:



In textual entries, wildcards can be used.

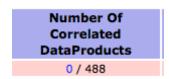


• You can put a list of SAS/Observation IDs in the query:

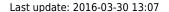


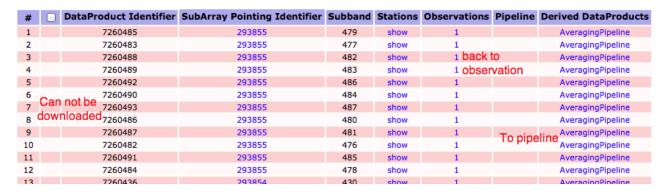
# Viewing data

When you are looking at the results of a query you might see something like this:



This means that the observation is known in the LTA, it knows what data was produced, the produced data was not archived, but further processing happened on the raw data and the results of some of those pipelines were archived. If you click on the zero, you will see something like this:





This allows you to navigate from a pipeline back to the original observation, or from the observation to any pipelines that have run on the raw data.

# **Retrieving data**

 You can retrieve data on the Observation and Pipeline level, you don't have to select all files individually.

#		Observation Id	Observing Mode	Antenna Set	Instrun Filte
1	<b>v</b>	146448	Interferometer	HBA Dual Inner	110-190
2		146447	Interferometer	HBA Dual Inner	110-190
3	✓	146446	Interferometer	HBA Dual Inner	110-190
4		146445	Interferometer	HBA Dual Inner	110-190
5	$\overline{\mathbf{v}}$	146444	Interferometer	HBA Dual Inner	110-190
6	$\checkmark$	146443	Interferometer	HBA Dual Inner	110-190
7		146442	Interferometer	HBA Dual Inner	110-190
8	$\checkmark$	146441	Interferometer	HBA Dual Inner	110-190
9	⋖	146456	Interferometer	HBA Dual Inner	110-190
10	$\checkmark$	146455	Interferometer	HBA Dual Inner	110-190
11		146454	Interferometer	HBA Dual Inner	110-190
12		146453	Interferometer	HBA Dual Inner	110-190
13		146452	Interferometer	HBA Dual Inner	110-190

• If you have a query with more than 1000 results, you can open the multiple pages each in a separate tab/window.

## Observation 1001 to 1100 (showing 100 of total 1156) -

With the small triangle next to a list, you can fold or unfold the list to get a better overview.

#### **Folded entries**

Observation 1 to 100 (showing 100 of total 1156)

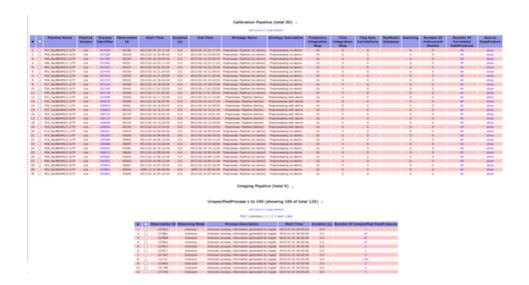
Averaging Pipeline 1 to 100 (showing 100 of total 4060)

Calibration Pipeline (total 30)

Imaging Pipeline (total 0)

UnspecifiedProcess 1 to 100 (showing 100 of total 125)

#### **Unfolded entries**



## **DBView**

There is a server that gives the option to run your own queries on the database http://lofar-dbview.target.rug.nl/

A useful query might be this one, that gives you all files for a certain Obs Id (SAS VIC tree ID).

```
SELECT fo.URI, dp."dataProductType", dp."dataProductIdentifier",
    dp."processIdentifier"
FROM AWOPER."DataProduct+" dp,
        AWOPER.FileObject fo,
        AWOPER."Process+" pr
WHERE dp."processIdentifier" = pr."processIdentifier"
    AND pr."observationId" = '123456'
    AND fo.data_object = dp."object_id"
```

Last update: 2016-03-30 13:07

```
AND dp."isValid" > 0
```

In this '123456' should be replaced with the Obs Id of an Observation/Pipeline you're looking for.

## **AstroWise Python Interface**

There is also a python interface to the LTA. With this, you can also script some advanced queries. To have this working, you first need to install the LTA client in your machine. Once you have installed the client, set up your user name and password. These are the same as for MoM. Remember that this is just a different interface to the LTA catalogue: you will need the same credentials as for the web interface.

In your home directory, add the following to the file .awe/Environment.cfg

```
database_user : <your username>
database_password : <your password>
```

then create the variable awetarget and set it to awlofar. In a bash shell, you can do so by adding the following to your .profile file:

```
export AWETARGET=awlofar
```

Finally, your hostname may cause an error, if it does not contain a full domain. In this case, check your /etc/hosts file. You should find a line that looks like this

```
127.0.0.1 localhost
```

Change that line into

```
127.0.0.1 localhost <your_host_name>
```

If you do not know your hostname, just type hostname in a shell and you will get it as an output.

Now, you can use the following script as a test. It may still give you some warnings, but if it works fine, it will print out all the observations in a certain patch of the sky archived in the LTA:

```
(Pointing.declination < 30)
print "Total Pointings %d" % len(query)
for pointing in query :
    print "Pointing found RA %f DEC %f" % (pointing.rightAscension,
pointing.declination)
    query_subarr = SubArrayPointing.pointing == pointing
    for subarr in query_subarr:
        query_obs = Observation.subArrayPointings.contains(subarr)
        for obs in query_obs :
            obs_ids.add(obs.observationId)
    result[project] = sorted(list(obs_ids))
    print result[project]</pre>
```

Once you have tested that your connection to the catalogue is working, you are ready to browse the archive and stage the data you need. Here we will list a few examples of python scripts that can be used to access the LTA. All of them will need to import some modules:

```
from datetime import datetime
from awlofar.database.Context import context
from awlofar.main.aweimports import CorrelatedDataProduct, \
    FileObject, \
    Observation
from awlofar.toolbox.LtaStager import LtaStager, LtaStagerError
```

The lines above must be added to each of the scripts below for these to work.

This simple script will allow you to find and stage all data within a single project LCX YYY.

```
do stage = True
project = 'LCX YYY'
cls = CorrelatedDataProduct
if not context.set_project(project) :
    raise Exception("You are not member of project %s" % project)
query_observations = Observation.select_all().project_only()
uris = set() # All URIS to stage
for observation in query observations :
    print("Querying ObservationID %s" % observation.observationId)
   # Instead of querying on the Observations of the DataProduct, all
DataProducts could have been gueried
    dataproduct query = cls.observations.contains(observation)
   # isValid = 1 means there should be an associated URI
   dataproduct query &= cls.isValid == 1
    for dataproduct in dataproduct query :
        # This DataProduct should have an associated URL
        fileobject = ((FileObject.data_object == dataproduct) &
(FileObject.isValid > 0)).max('creation_date')
        if fileobject :
            print("URI found %s" % fileobject.URI)
```

```
uris.add(fileobject.URI)
  else :
        print("No URI found for %s with dataProductIdentifier %d" %
  (dataproduct.__class__.__name__, dataproduct.dataProductIdentifier))

print("Total URI's found %d" % len(uris))

if do_stage :
    stager = LtaStager()
    stager.stage_uris(uris)
```

The following will find and stage subbands 301 and 302 for all targets within two different projects.

```
do stage = False
project1 = 'LCX YYY'
project2 = 'LCZ VVV'
subband1 = 301
subband2 = 302
cls = CorrelatedDataProduct
# All URIS to stage
uris = {
    project1: set(),
    project2: set(),
}
for project in (project1, project2) :
    print("Using project %s" % project)
    if not context.set project(project) :
        raise Exception("You are not member of project %s" % project)
    query observations = Observation.select all().project only()
    for observation in query observations :
        print("Querying ObservationID %s" % observation.observationId)
        dataproduct query = cls.observations.contains(observation)
        # isValid = 1 means there should be an associated URI
        dataproduct query &= cls.isValid == 1
        dataproduct_query &= ((cls.subband == subband1) | (cls.subband ==
subband2))
        # Or for stationSubband do :
        #dataproduct_query &= ((cls.stationSubband == subband1) |
(cls.stationSubband == subband2))
        for dataproduct in dataproduct guery :
            # This DataProduct should have an associated URL
            fileobject = ((FileObject.data_object == dataproduct) &
(FileObject.isValid > 0)).max('creation date')
            if fileobject :
                print("URI found %s" % fileobject.URI)
                uris[project].add(fileobject.URI)
                print("No URI found for %s with dataProductIdentifier %d" %
(dataproduct.__class__.__name__, dataproduct.dataProductIdentifier))
```

```
for project in (project1, project2) :
    print("Total URI's found for project %s: %d" % (project,
len(uris[project])))

stager = LtaStager()
for project in (project1, project2) :
    if do_stage :
        stager.stage_uris(uris[project])
```

Here, we find and stage data between freq1 and freq2 taken within one project between day1 and day2

```
do stage = True
project = 'LCX YYY'
freq1 = 172.0
freg2 = 178.0
day1 = datetime(2016,1,20) # this could include time; ie hours, minutes,
secondes
day2 = datetime(2016, 1, 31) # idem
# DataProduct class to query; CorrelatedDataProduct, SkyImageDataProduct,
cls = CorrelatedDataProduct
if not context.set project(project) :
    raise Exception("You are not member of project %s" % project)
query observations = ((Observation.startTime >= day1) &
                       (Observation.endTime < day2)).project only()
uris = set()
for observation in query observations :
    \label{lem:print} \verb|print("Querying ObservationID %s" % observation.observationId)| \\
    dataproduct query = cls.observations.contains(observation)
    # isValid = 1 means there should be an associated URI
    dataproduct_query &= cls.isValid == 1
    dataproduct query &= cls.minimumFrequency >= freq1
    dataproduct query &= cls.maximumFrequency < freq2</pre>
    for dataproduct in dataproduct query :
        # This DataProduct should have an associated URL
        fileobject = ((FileObject.data object == dataproduct) &
(FileObject.isValid > 0)).max('creation_date')
        if fileobject:
            print("URI found %s" % fileobject.URI)
            uris.add(fileobject.URI)
        else:
            print("No URI found for %s with dataProductIdentifier %d" %
(dataproduct.__class__.__name__, dataproduct.dataProductIdentifier))
print("Total URI's found %d" % len(uris))
if do stage :
    stager = LtaStager()
```

### stager.stage\_uris(uris)

From:

https://www.astron.nl/lofarwiki/ - LOFAR Wiki

Permanent link:

https://www.astron.nl/lofarwiki/doku.php?id=public:lta\_tricks&rev=1459343244

Last update: 2016-03-30 13:07

