

# 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:

Observation Id	432000:432190
Observing or Pipeline Run Date	From 0000-00-00 00:00:00 To 0000-00-00 00:00:00
Project	any
Maximum Number of Rows	

In textual entries, wildcards can be used.

Target Name	3:19*
-------------	-------

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

Observation Id	146112,147775,151778
Observing Date	From 0000-00-00 00:00:00 To 0000-00-00 00:00:00

## Viewing data

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

Number Of Correlated DataProducts
0 / 488

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:

#	<input type="checkbox"/>	DataProduct Identifier	SubArray Pointing Identifier	Subband	Stations	Observations	Pipeline	Derived DataProducts
1	<input type="checkbox"/>	7260485	293855	479	show	1		AveragingPipeline
2	<input type="checkbox"/>	7260483	293855	477	show	1		AveragingPipeline
3	<input type="checkbox"/>	7260488	293855	482	show	1	back to observation	AveragingPipeline
4	<input type="checkbox"/>	7260489	293855	483	show	1		AveragingPipeline
5	<input type="checkbox"/>	7260492	293855	486	show	1		AveragingPipeline
6	<input type="checkbox"/>	7260490	293855	484	show	1		AveragingPipeline
7	Can not be downloaded	7260493	293855	487	show	1		AveragingPipeline
8		7260486	293855	480	show	1		AveragingPipeline
9	<input type="checkbox"/>	7260487	293855	481	show	1	To pipeline	AveragingPipeline
10	<input type="checkbox"/>	7260482	293855	476	show	1		AveragingPipeline
11	<input type="checkbox"/>	7260491	293855	485	show	1		AveragingPipeline
12	<input type="checkbox"/>	7260484	293855	478	show	1		AveragingPipeline
13	<input type="checkbox"/>	7260436	293854	430	show	1		AveragingPipeline

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.

#	<input type="checkbox"/>	Observation Id	Observing Mode	Antenna Set	Instrun Filt
1	<input checked="" type="checkbox"/>	146448	Interferometer	HBA Dual Inner	110-190
2	<input type="checkbox"/>	146447	Interferometer	HBA Dual Inner	110-190
3	<input checked="" type="checkbox"/>	146446	Interferometer	HBA Dual Inner	110-190
4	<input type="checkbox"/>	146445	Interferometer	HBA Dual Inner	110-190
5	<input checked="" type="checkbox"/>	146444	Interferometer	HBA Dual Inner	110-190
6	<input checked="" type="checkbox"/>	146443	Interferometer	HBA Dual Inner	110-190
7	<input type="checkbox"/>	146442	Interferometer	HBA Dual Inner	110-190
8	<input checked="" type="checkbox"/>	146441	Interferometer	HBA Dual Inner	110-190
9	<input checked="" type="checkbox"/>	146456	Interferometer	HBA Dual Inner	110-190
10	<input checked="" type="checkbox"/>	146455	Interferometer	HBA Dual Inner	110-190
11	<input type="checkbox"/>	146454	Interferometer	HBA Dual Inner	110-190
12	<input type="checkbox"/>	146453	Interferometer	HBA Dual Inner	110-190
13	<input type="checkbox"/>	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) ▾

edit columns | stage selected

first | previous | ... | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | next | last

r Of SubArray	Start Time	Duration	Nr Stations	Nr Stations
---------------	------------	----------	-------------	-------------

- 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

Collation Pipeline (Total 30) -

48 columns | Page 100

#	ID	Pipeline Name	Pipeline Version	Process Identifier	Observation ID	Start Time	Duration (s)	End Time	Strategy Name	Strategy Description	Frequency Interval	Time Integration Step	Flag Rate Conversion	Workflow Substep	Scoring	Number Of Instrument Reads	Number Of Generated Data Products	Source Reference
1	1	Pipeline_Nominal_1_LTR	1.0.0	20170101	01001	2013-01-01 01:00:00.00	0.0	2013-01-01 01:01:00.00	Preprocessing: Pipeline ID=01001	Preprocessing with default	1s	0	0	0	0	0	0	data
2	2	Pipeline_Nominal_1_LTR	1.0.0	20170101	01010	2013-01-01 01:10:00.00	0.0	2013-01-01 01:11:00.00	Preprocessing: Pipeline ID=01010	Preprocessing with default	1s	0	0	0	0	0	0	data
3	3	Pipeline_Nominal_1_LTR	1.0.0	20170101	01020	2013-01-01 01:20:00.00	0.0	2013-01-01 01:21:00.00	Preprocessing: Pipeline ID=01020	Preprocessing with default	1s	0	0	0	0	0	0	data
4	4	Pipeline_Nominal_1_LTR	1.0.0	20170101	01030	2013-01-01 01:30:00.00	0.0	2013-01-01 01:31:00.00	Preprocessing: Pipeline ID=01030	Preprocessing with default	1s	0	0	0	0	0	0	data
5	5	Pipeline_Nominal_1_LTR	1.0.0	20170101	01040	2013-01-01 01:40:00.00	0.0	2013-01-01 01:41:00.00	Preprocessing: Pipeline ID=01040	Preprocessing with default	1s	0	0	0	0	0	0	data
6	6	Pipeline_Nominal_1_LTR	1.0.0	20170101	01050	2013-01-01 01:50:00.00	0.0	2013-01-01 01:51:00.00	Preprocessing: Pipeline ID=01050	Preprocessing with default	1s	0	0	0	0	0	0	data
7	7	Pipeline_Nominal_1_LTR	1.0.0	20170101	01060	2013-01-01 01:59:00.00	0.0	2013-01-01 02:00:00.00	Preprocessing: Pipeline ID=01060	Preprocessing with default	1s	0	0	0	0	0	0	data
8	8	Pipeline_Nominal_1_LTR	1.0.0	20170101	01070	2013-01-01 02:00:00.00	0.0	2013-01-01 02:01:00.00	Preprocessing: Pipeline ID=01070	Preprocessing with default	1s	0	0	0	0	0	0	data
9	9	Pipeline_Nominal_1_LTR	1.0.0	20170101	01080	2013-01-01 02:10:00.00	0.0	2013-01-01 02:11:00.00	Preprocessing: Pipeline ID=01080	Preprocessing with default	1s	0	0	0	0	0	0	data
10	10	Pipeline_Nominal_1_LTR	1.0.0	20170101	01090	2013-01-01 02:20:00.00	0.0	2013-01-01 02:21:00.00	Preprocessing: Pipeline ID=01090	Preprocessing with default	1s	0	0	0	0	0	0	data
11	11	Pipeline_Nominal_1_LTR	1.0.0	20170101	01100	2013-01-01 02:30:00.00	0.0	2013-01-01 02:31:00.00	Preprocessing: Pipeline ID=01100	Preprocessing with default	1s	0	0	0	0	0	0	data
12	12	Pipeline_Nominal_1_LTR	1.0.0	20170101	01110	2013-01-01 02:40:00.00	0.0	2013-01-01 02:41:00.00	Preprocessing: Pipeline ID=01110	Preprocessing with default	1s	0	0	0	0	0	0	data
13	13	Pipeline_Nominal_1_LTR	1.0.0	20170101	01120	2013-01-01 02:50:00.00	0.0	2013-01-01 02:51:00.00	Preprocessing: Pipeline ID=01120	Preprocessing with default	1s	0	0	0	0	0	0	data
14	14	Pipeline_Nominal_1_LTR	1.0.0	20170101	01130	2013-01-01 03:00:00.00	0.0	2013-01-01 03:01:00.00	Preprocessing: Pipeline ID=01130	Preprocessing with default	1s	0	0	0	0	0	0	data
15	15	Pipeline_Nominal_1_LTR	1.0.0	20170101	01140	2013-01-01 03:10:00.00	0.0	2013-01-01 03:11:00.00	Preprocessing: Pipeline ID=01140	Preprocessing with default	1s	0	0	0	0	0	0	data
16	16	Pipeline_Nominal_1_LTR	1.0.0	20170101	01150	2013-01-01 03:20:00.00	0.0	2013-01-01 03:21:00.00	Preprocessing: Pipeline ID=01150	Preprocessing with default	1s	0	0	0	0	0	0	data
17	17	Pipeline_Nominal_1_LTR	1.0.0	20170101	01160	2013-01-01 03:30:00.00	0.0	2013-01-01 03:31:00.00	Preprocessing: Pipeline ID=01160	Preprocessing with default	1s	0	0	0	0	0	0	data
18	18	Pipeline_Nominal_1_LTR	1.0.0	20170101	01170	2013-01-01 03:40:00.00	0.0	2013-01-01 03:41:00.00	Preprocessing: Pipeline ID=01170	Preprocessing with default	1s	0	0	0	0	0	0	data
19	19	Pipeline_Nominal_1_LTR	1.0.0	20170101	01180	2013-01-01 03:50:00.00	0.0	2013-01-01 03:51:00.00	Preprocessing: Pipeline ID=01180	Preprocessing with default	1s	0	0	0	0	0	0	data
20	20	Pipeline_Nominal_1_LTR	1.0.0	20170101	01190	2013-01-01 04:00:00.00	0.0	2013-01-01 04:01:00.00	Preprocessing: Pipeline ID=01190	Preprocessing with default	1s	0	0	0	0	0	0	data
21	21	Pipeline_Nominal_1_LTR	1.0.0	20170101	01200	2013-01-01 04:10:00.00	0.0	2013-01-01 04:11:00.00	Preprocessing: Pipeline ID=01200	Preprocessing with default	1s	0	0	0	0	0	0	data
22	22	Pipeline_Nominal_1_LTR	1.0.0	20170101	01210	2013-01-01 04:20:00.00	0.0	2013-01-01 04:21:00.00	Preprocessing: Pipeline ID=01210	Preprocessing with default	1s	0	0	0	0	0	0	data
23	23	Pipeline_Nominal_1_LTR	1.0.0	20170101	01220	2013-01-01 04:30:00.00	0.0	2013-01-01 04:31:00.00	Preprocessing: Pipeline ID=01220	Preprocessing with default	1s	0	0	0	0	0	0	data
24	24	Pipeline_Nominal_1_LTR	1.0.0	20170101	01230	2013-01-01 04:40:00.00	0.0	2013-01-01 04:41:00.00	Preprocessing: Pipeline ID=01230	Preprocessing with default	1s	0	0	0	0	0	0	data
25	25	Pipeline_Nominal_1_LTR	1.0.0	20170101	01240	2013-01-01 04:50:00.00	0.0	2013-01-01 04:51:00.00	Preprocessing: Pipeline ID=01240	Preprocessing with default	1s	0	0	0	0	0	0	data
26	26	Pipeline_Nominal_1_LTR	1.0.0	20170101	01250	2013-01-01 05:00:00.00	0.0	2013-01-01 05:01:00.00	Preprocessing: Pipeline ID=01250	Preprocessing with default	1s	0	0	0	0	0	0	data
27	27	Pipeline_Nominal_1_LTR	1.0.0	20170101	01260	2013-01-01 05:10:00.00	0.0	2013-01-01 05:11:00.00	Preprocessing: Pipeline ID=01260	Preprocessing with default	1s	0	0	0	0	0	0	data
28	28	Pipeline_Nominal_1_LTR	1.0.0	20170101	01270	2013-01-01 05:20:00.00	0.0	2013-01-01 05:21:00.00	Preprocessing: Pipeline ID=01270	Preprocessing with default	1s	0	0	0	0	0	0	data
29	29	Pipeline_Nominal_1_LTR	1.0.0	20170101	01280	2013-01-01 05:30:00.00	0.0	2013-01-01 05:31:00.00	Preprocessing: Pipeline ID=01280	Preprocessing with default	1s	0	0	0	0	0	0	data
30	30	Pipeline_Nominal_1_LTR	1.0.0	20170101	01290	2013-01-01 05:40:00.00	0.0	2013-01-01 05:41:00.00	Preprocessing: Pipeline ID=01290	Preprocessing with default	1s	0	0	0	0	0	0	data

Imaging Pipeline (Total 8) -

UnspecifiedProcess 5 100 (showing 100 of total 120) -

48 columns | Page 100

Arg : pipeline 1

## 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"
```

```
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 script some advanced queries. To have this working, you first need to install the [LTA client](#) in your machine. Be sure to have the latest version installed.

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.

After installing the LTA client, the file `.awe/Environment.cfg` will appear in your home directory. Add the following lines to the file

```
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, you will need to edit your `/etc/hosts` file (you will need root/superuser privileges to do so). In that 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 prints out a list of pointings, then you are ready to go. You may need to kill the script, because it will print out all the observations in a certain patch of the sky archived in the LTA.

```
# python code
from pprint import pprint
from common.database.Context import context
from awlofar.main.aweimports import Observation, Pointing, SubArrayPointing
result = {}
for project in sorted(context.get_projects()) :
    print "Project %(project)s" % vars()
    ok = context.set_project(project)
```

```
# do your query
obs_ids = set()
query = (Pointing.rightAscension > 95) & \
        (Pointing.rightAscension < 105) & \
        (Pointing.declination > 20) & \
        (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]

pprint(result)
```

If you get errors and do not manage to view the list of pointings, there may be the need to open some port on the firewall at your institution. Specifically, port 1521 should be open. In case of trouble, get in contact with Science Support.

## Examples

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 all data within a single project, for example LC2\_035. Please change the project name to the code of a project of yours. If you also want to stage the data you found, just set the do\_stage variable to True. Be careful with how many files you stage and what size they have: the same limit as for the web interface apply here.

```
do_stage = False
project = 'LC2_035'
cls = CorrelatedDataProduct
private_data = False
```

```
# To see private data of this project, you must be member of this project
if private_data and not context.set_project(project) :
    raise Exception("You are not member of project %s" % project)

query_observations = Observation.select_all().project_only(project)
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 queried
    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 script will find subbands 301 and 302 for all targets within two different projects.

Pay attention to the difference between the keys subband and stationSubband; the former is a sequential number assigned to each subband in an observation, while the latter is linked to the frequency at which the observation was performed. Example: an observation was set up covering the range 30-77.3 MHz with two simultaneous beams using 244 subbands each. In this case, subband will range from 0 to 487, while stationSubband from 153 to 396. The stationSubband information is stored in the observation, but not in the pipeline products (which instead contain the frequency). If you want to search on stationSubband, you must perform your search on observations first, then fetch the pipelines linked to those observations. If you use frequency, you can search directly on pipelines.

As a general advise, before performing a search, you need to **understand thoroughly the meaning of the keywords that you are using and where their values are stored**, otherwise you may not find the data you are looking for.

```
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_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[project].add(fileobject.URI)
            else :
                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 data between freq1 and freq2 taken within one project between day1 and day2

```

do_stage = False
project = 'LCX_YYY'
freq1 = 172.0
freq2 = 178.0
day1 = datetime(2016,1,20) # this could include time; ie hours, minutes,
seconds
day2 = datetime(2016,1,31) # idem
# DataProduct class to query; CorrelatedDataProduct, SkyImageDataProduct,

```

```
etc ...
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 :
    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
    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=1459789230](https://www.astron.nl/lofarwiki/doku.php?id=public:lta_tricks&rev=1459789230)Last update: **2016-04-04 17:00**