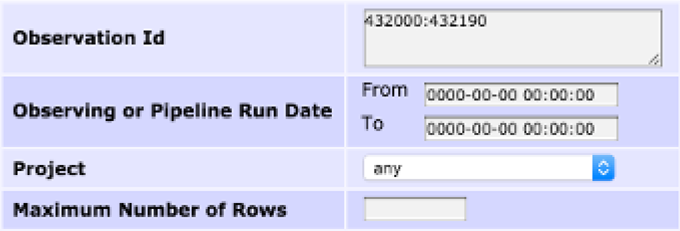
Advanced ways to find and retrieve data in the LTA

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:

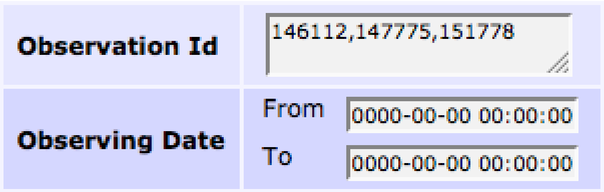


Observation Id	432000:432190
Observing or Pipeline Run Date	From 0000-00-00 00:00:00 To 0000-00-00 00:00:00
Project	any
Maximum Number of Rows	

In textual entries, wildcards can be used.

Target Name	3c19*

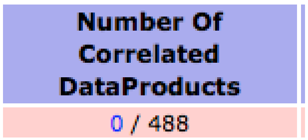
- You can put a list of SAS/Observation IDs in the query:



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

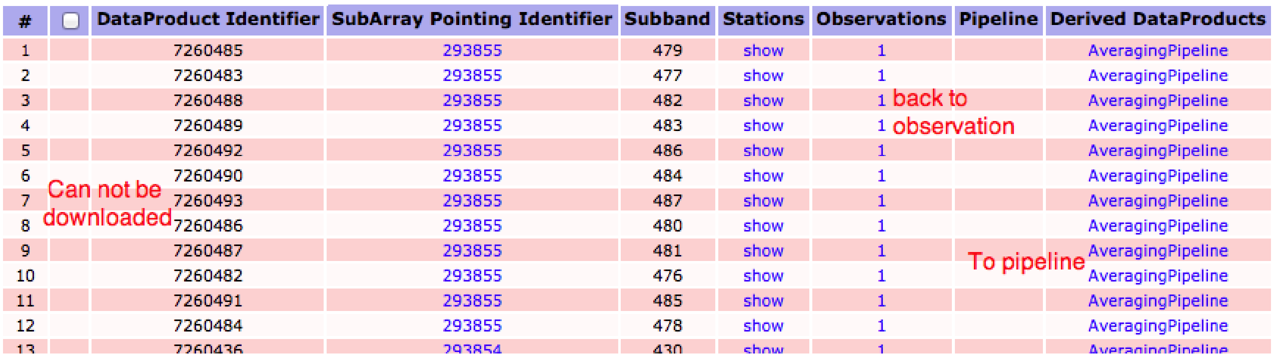
Viewing data

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:

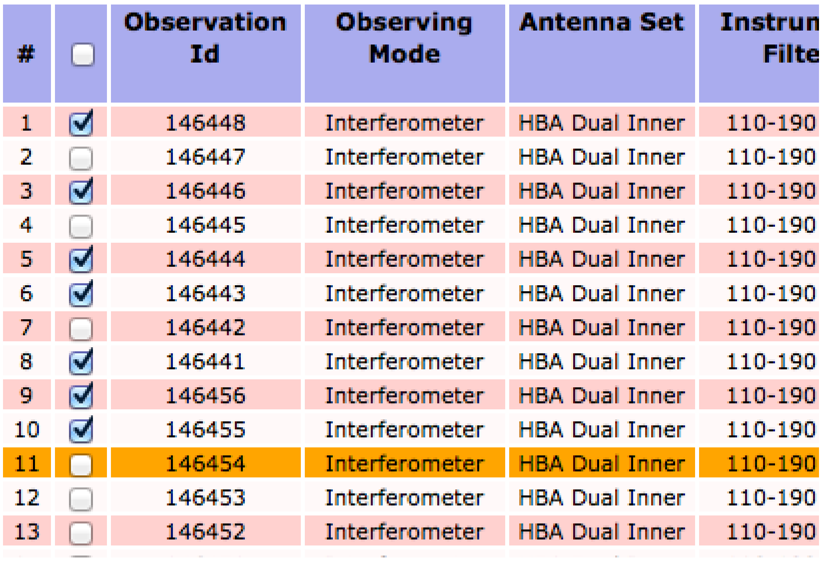


#	☐	DataProduct Identifier	SubArray Pointing Identifier	Subband	Stations	Observations	Pipeline	Derived DataProducts
1		7260485	293855	479	show	1		AveragingPipeline
2		7260483	293855	477	show	1		AveragingPipeline
3		7260488	293855	482	show	1		AveragingPipeline
4		7260489	293855	483	show	1		AveragingPipeline
5		7260492	293855	486	show	1		AveragingPipeline
6		7260490	293855	484	show	1		AveragingPipeline
7		7260493	293855	487	show	1		AveragingPipeline
8		7260486	293855	480	show	1		AveragingPipeline
9		7260487	293855	481	show	1		AveragingPipeline
10		7260482	293855	476	show	1		AveragingPipeline
11		7260491	293855	485	show	1		AveragingPipeline
12		7260484	293855	478	show	1		AveragingPipeline
13		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.

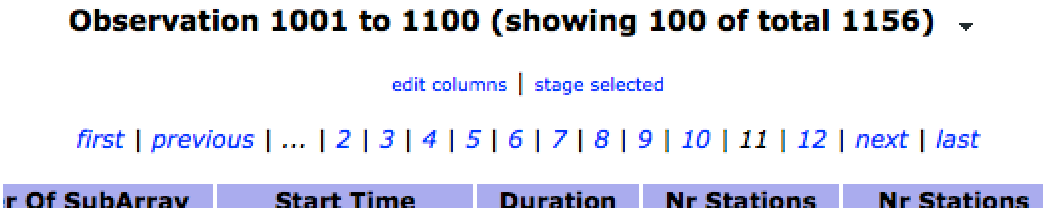
Retrieving data

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



#	☐	Observation Id	Observing Mode	Antenna Set	Instrun Filte
1	☑	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	☑	146444	Interferometer	HBA Dual Inner	110-190
6	☑	146443	Interferometer	HBA Dual Inner	110-190
7	☐	146442	Interferometer	HBA Dual Inner	110-190
8	☑	146441	Interferometer	HBA Dual Inner	110-190
9	☑	146456	Interferometer	HBA Dual Inner	110-190
10	☑	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.



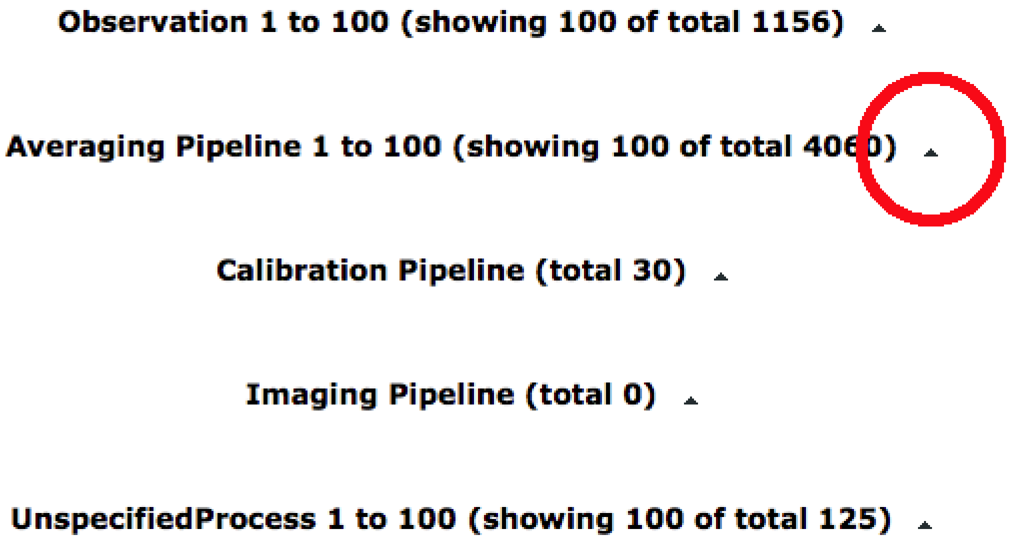
Observation 1001 to 1100 (showing 100 of total 1156) ▾

edit columns | stage selected

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

- 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)

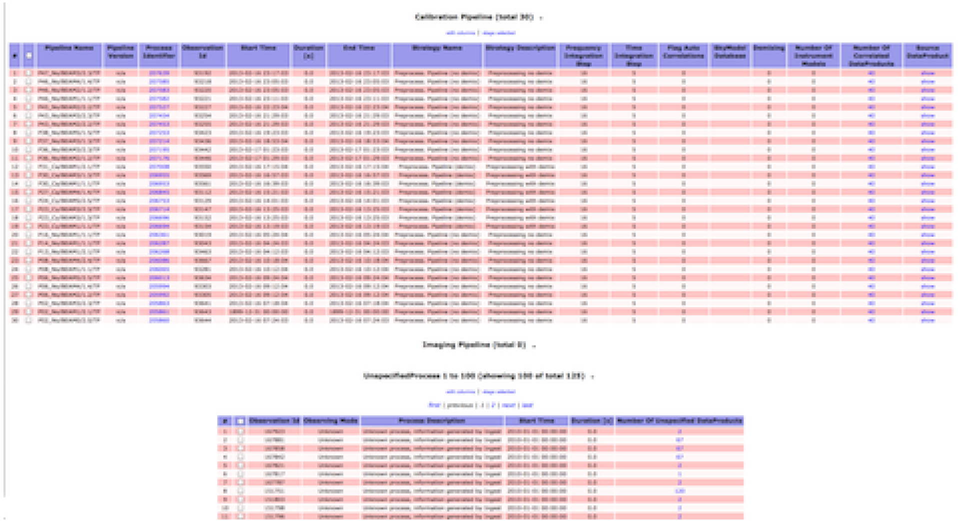
Averaging Pipeline 1 to 100 (showing 100 of total 4060)

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 https://lta-dbview.lofar.eu/

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. Pipelines also have an “observationId” == the SAS Id, even though that's a but confusing. To be able to run this query, you have to go to the link above, login as the right user, select the right project, and then put this query into the “Manual SQL”.

Example You can also modify these queries. for example if you want to also know the MD5 checksum, you can run:

```
SELECT fo.URI, fo.hash_md5, 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
```

AstroWise Python Interface

There is a Python client library for accessing the LTA. With this library, you can script your own queries. The installation description can be found here: LTA Client installation. Be sure to have the latest version installed. Note that since January 2018 this library uses python3, python2 is no longer supported.

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 (if not, then create one). Make sure the file at least contains the following lines:

```

[global]
database_user       : <your username>
database_password   : <your password>
```

The following script can be used to test your installation:

```
# Python3 code
from pprint import pprint
from awlofar.main.aweimports import Observation, Pointing, SubArrayPointing
from common.database.Context import context
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)
```

It should print out a list of pointings (note that in this example the library was installed in $HOME/tmp):

```
$ env PYTHONPATH=$HOME/tmp/lib/python3.5/site-packages python3 lta_test.py
Project ALL
Total Pointings 202
Pointing found RA 95.003499 DEC 24.838742
Pointing found RA 95.174754 DEC 28.660087
Pointing found RA 95.220000 DEC 29.140000
Pointing found RA 95.546250 DEC 23.331750
Pointing found RA 95.561458 DEC 24.584056
..etc..
```

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.

In case of errors, there may be the need to open some port on the firewall at your institution. Specifically, port 1521 should be open. Also make sure that the LTA client library can be found in your PYTHONPATH (see LTA Client installation for more details). In case of trouble, get in contact with Science Operations and 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.

Ex: get staging URI's

This 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 limits as for the web interface apply here.

```
# Should the found files be staged ?
do_stage = False
# The project to query, LC2_035 has public data
project = 'LC2_035'
# The class of data to query
cls = CorrelatedDataProduct
# Query for private data of the project, you must be member of the project
private_data = False

# To see private data of this project, you must be member of this project
if private_data :
    context.set_project(project)
    if project != context.get_current_project().name:
        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)
```

Ex: filter on subbands

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 = 'LC2_016'
project2 = 'LC2_012'
subband1 = 301
subband2 = 302
cls = CorrelatedDataProduct
# Query for private data of the project, you must be member of the project
private_data = False

# All URIS to stage
uris = {
    project1: set(),
    project2: set(),
}

for project in (project1, project2) :
    print("Using project %s" % project)
    if private_data :
        context.set_project(project)
        if project != context.get_current_project().name:
            raise Exception("You are not member of project %s" % project)
    query_observations = Observation.select_all().project_only(project)
    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()
if do_stage :
    for project in (project1, project2) :
        stager.stage_uris(uris[project])
```

Ex: filter on frequency and observation date

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

```
do_stage = False
project = 'LC2_033'
freq1 = 172.0
freq2 = 178.0
day1 = datetime(2014,8,26) # this could include time; ie hours, minutes,
secondes
day2 = datetime(2014,8,29) # idem
# DataProduct class to query; CorrelatedDataProduct, SkyImageDataProduct,
etc ...
cls = CorrelatedDataProduct
# Query for private data of the project, you must be member of the project
private_data = False

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

query_observations = (
    (Observation.startTime >= day1) &
```

```
    (Observation.endTime   <  day2) ).project_only(project)

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)
```

Ex: query public data

Querying public data in projects you are not member of. First set project ALL, then construct a query and optionally limit the query to a certain project :

```
context.set_project('ALL')
query = CorrelatedDataProduct.select_all()
query &= query.project_only('LC0_017')
print(len(query))
# 1800
```

Ex: get release dates

```
from awlofar.main.aweimports import Observation, PipelineRun, DataProduct
from common.database.Context import context

project = 'LC2_035'

# Query for private data of the project, you must be member of the project
private_data = True

```

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

# Observations
query_observations = Observation.select_all().project_only(project)
for observation in query_observations :
    print("Querying ObservationID %s, %s" % (observation.observationId,
observation.releaseDate))

# Pipelines
query_pipelines = PipelineRun.select_all().project_only(project)
for pipeline in query_pipelines :
    print("Pipeline: %s, %s, %s" % (type(pipeline).__name__,
pipeline.pipelineName, pipeline.releaseDate))

# Data products
query_products = DataProduct.select_all().project_only(project)
query_products &= DataProduct.isValid == 1
for product in query_products :
    print("Product: %s, %s, %s, %s" % (product.dataProductIdentifier,
product.dataProductIdentifierSource, product.dataProductType,
product.releaseDate))
```

Python Module for Staging

The python interaction with the LTA catalog can be complemented with the use of a specific module developed to give users more control over their staging requests. Please note that there are two versions currently available:

- Current unreleased version 2.0 (master branch) is to be used with the new LTA stager (stageit).

- Please see **“Version 2.0 usage notes”** listed below for documentation (or check the README file in the repository linked above)
- User documentation for *stageit* can be found at: https://support.astron.nl/confluence/display/SDCP/User+documentation
- Released version 1.7 is to be used with the current (soon to be old) production environment. Please check out the version found at: https://git.astron.nl/astron-sdc/lofar_stager_api/-/releases/1.7
- Please see **“Version 1.7 usage notes”** listed below for documentation

Version 2.0 usage notes

The module is made available here (master branch).

Notes:

- You need an access token to the stageit api. Please refer to the user guide linked above to sign up and login to stageit. After logging in, a token can be obtained in one of two ways:
    - Visit https://sdc.astron.nl/stageit/api/staging/get-token
    - From anywhere in the application, click on your account name in the top right to access your profile. From your profile page, click the “Request token” button to receive a token.
- The token is valid indefinitely. Requesting a token multiple times will yield the same token.
- Make sure the token is available in your ~/.stagingrc file:
    - api_token=YOUR_TOKEN_HERE
    - remove the old username and password from the `.stagingrc` file
- The script is Python2 compatible, there is a Dockerfile available for Python2 testing in `./tests/docker`
- The requests library is a required dependency. If you care about Python2 compatability, you can use at most version 2.22.0 of requests. Otherwise, you can install any version (note: you can also `pip install -r 'requirements.txt'`, which will install version 2.22.0)

Also note that some functions are not supported in the new LTA stager. Upon use, they will display an error stating that the function is deprecated. Please look at the `stager_access.py` file for more information.

Version 1.7 usage notes

The module is made available **here** and its functions are mostly self-explanatory.

Alternatively to the .awe/Environment.cfg described above, user credentials can also be provided via a file ~/.stagingrc with credentials of your Lofar account, similar to ./wgetrc:

```
user=XXX
password=YYY
```

For a description of what the user can do, we list here the functions that are available.

stage(surls)
It takes in a list of surls, queues a staging request for those urls, and outputs the ID of the request.

get_status(stageid)
It tells the user if a request is queued, in progress or finished (success). Possible statuses: “new”, “scheduled”, “in progress”, “aborted”, “failed”, “partial success”, “success”, “on hold”

abort(stageid)
It allows users to end a staging request.

get_surls_online(stageid)
It gives a list of the surls that have been staged for the relative request. The list is updated whenever a new surl comes on line.

get_srm_token(stageid)
The srm token is useful to interact directly with the SRM site through GRID/SRM tools.

reschedule(stageid)
If a request failed, it can be rescheduled.

get_progress()
No input needed. It returns the statuses of all the requests owned by the user.

Below is an example of how to use this:

```
> python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.>>>
import stager_access as sa

2016-11-24 16:39:55.865000 stager_access: Parsing user credentials from
/Users/renting/.stagingrc
2016-11-24 16:39:55.865111 stager_access: Creating proxy>>>
sa.prettyprint(sa.get_progress())

+ 12227
  - File count      ->     100
  - Files done      ->     40
  - Flagged abort      ->     false
  - Location      ->     fz-juelich
  - Percent done      ->     40
  - Status      ->     on hold
  - User id      ->     1919
```

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

Permanent link:
https://www.astron.nl/lofarwiki/doku.php?id=public:lta_tricks&rev=1682324363

Last update: **2023-04-24 08:19**