API Advanced Search

Using an Application Programming Interface (API) to query and retrieve data from MAST is efficient, and you can customize scripts that use the APIs very easily. This tutorial illustrates how to formulate queries with the astroquery.mast Python package, which is analogous to using the *Advanced Search* feature of the MAST Portal web interface. To get started, we suggest trying the Basic Search Jupyter Notebook, which covers many of the essential features of astroquery.mast.

On this page...

Prerequisites

Searching and retrieving data from MAST using astroquery requires that you have:

1. A relatively recent version of python (3.8+ recommended) installed on your machine, as well as the astroquery.mast package (v0.4.7+ recommended). These can be installed with a Conda environment.
2. You must also install astropy because the results from **astroquery** return **astropy** `Table` objects.
3. Have a valid MAST.auth token if you wish to retrieve exclusive-access data.

API Libraries

Using the astroquery.mast API is straightforward with the methods in the **`Observation`** class; specify parameter values to set search criteria. In a script or python session, begin by importing the necessary classes and methods:

```
from astroquery.mast import Observations
from astropy.table import unique, vstack, Table
```

API Queries

The workflow when using the API is analogous to that of using the Portal:

1. Use search criteria to match **Observations**
2. For each Observation of interest, find the specific data products contained within it
3. Use filters to select (or omit) specific categories of products, such as Science files, uncalibrated raw files, or guide-star files.

Query for Observations

Consult the set of all available parameters for **astroquery.mast**. Consider a search for JWST/NIRCam observations in observing program 1073, using the `F277W` filter. Use the **`.query_criteria()`** method to specify the query parameters:

```
matched_obs = Observations.query_criteria(
        obs_collection = 'JWST'
        , proposal_id = '1073'
        , instrument_name = 'NIRCAM*'
        , filters = 'F277W'
        )
```

Note the asterisk following the instrument name: that allows for queries for all instrument configurations. The result is an **astropy** `Table` object, with one row per matched observation.

If for a particular query you are not interested in observations from any mission except JWST, specify it with the parameter `obs_collection='JWST'`. This will narrow the list of possible matches considerably, and speed up your query.

Query for Data Products

Having found Observations that match your criteria, the next step is to fetch a table of data products associated with each Observation.

> ⚠ Some observations contain huge numbers of associated or linked files, sometimes in excess of 10,000 products. This is particularly true for NIRSpec/MSA, MIRI/WFSS, and NIRCam/WFSS spectroscopy, but may also be true for large mosaics of images. Each observation is likely to contain many files in common, such as guide-star files and ancillary products.
>
> **We recommend retrieving product lists for one or a few Observation(s) at a time to avoid server timeouts, and then to construct a set of unique products from the combined observations to avoid large numbers of duplicated products.**

The following retrieves a list of tables of data products for each observation, and returns combined table containing unique data products.

```
if (matched_obs > 0):
    t = [Observations.get_product_list(obs) for obs in matched_obs]
    files = unique(vstack(t), keys='productFilename')
```

If at least one observation matched the search criteria the above call returns a table of unique products, one per row, which could number in the hundreds or thousands. You may wish to filter the results by masking all but a limited number of file suffixes and excluding certain sub-strings. Below is such a function, with the explicit option of excluding guide-star products.

Filtering Filename Substrings

```
def product_filter(table, prodList, gs_omit=True):
    mask = np.full(len(table), False)
    gs_text = '_gs-'
    for r in table:
        mk = False
        fileName = r['productFilename']
        for p in prodList:
            if p in fileName:
                mk = mk | True
                if gs_text in fileName:
                    mk = not gs_omit

        mask[r.index] = mk
    return mask
```

Example API Queries

We recommended starting with the Basic Search Jupyter Notebook, as it covers many of the important features of astroquery.mast. In addition, other Notebooks in the MAST Notebooks GitHub Repository use the MAST API, so you may also find these useful.

Data Product Retrieval

The selected products may now be downloaded to your local machine.

Auth.MAST Token

Note that you will need to login with a valid Auth.MAST token to download exclusive access (EAP) data. If the token is needed but not supplied, the you will be prompted to enter one. In any case you have the option of logging in explicitly.

```
Observations.login()
```

> **Token security**
>
> Typing tokens into a terminal or a Jupyter notebook is a security risk. The Observations.login() method takes care of this automatically by using getpass for secure token entry.

Download Products

You may download all the products in the files table, or select a subset if you prefer. The following will select L-1b products (i.e. the raw, uncalibrated, *_uncal.fits) from the data product list for retrieval. Other types of products may instead be selected, either by matching product name sub-strings or selecting named products in the productSubGroupDescription.

```
manifest = Observations.download_products(
           files,
           productSubGroupDescription='UNCAL',
           curl_flag=True
           )
```

Setting the optional curl_flag parameter to **True** will instead download a **bash** script that contains **cURL** commands to fetch the files at a later time. This approach is highly recommended for large numbers of files. The name of the download script will be something like: mastDownload_**YYYYMMDDhhmmss**.sh, where the latter part of the name is a numeric timestamp. What remains is to invoke the downloaded script on your machine to retrieve the files.

For Further Reading...

- astroquery.mast