API Curl Download

When downloading large files, or a large number of them, it is often advisable to use a bash script to download via the API. These scripts use the **cURL** utility to retrieve the selected files. cURL is robust, reliable, and supports many features including authentication. It is also the preferred tool for downloading large volumes of data because of the data volume and rate limits imposed by the Portal Web interface.

On this page...

- Example Bash Script
  - The API Token
  - Retrieve Files
- Interactive Python Example
- For Further Reading...

Example Bash Script

When retrieving a selection of files from the MAST Portal, one of the download options is an auto-generated **bash** (shell) script. Note that this option will download only the script; the actual files are download when the script is run.

If you would like to customize the example script provided here to access data files in MAST, you will first need to determine the URIs for the files of interest using another API, e.g., astroquery.mast. See the 'Interactive Example' section down below to try it yourself.

You must invoke the script in a unix terminal to download the files:

```
bash MAST_2022-04-30T2140.sh
```

A look at the example script MAST_2022-04-30T2153.sh shows that, apart from some housekeeping, it does a few main things:

1. For EAP protected data, fetch the MAST API Token
2. Create a folder for the payload
3. Create a `MANIFEST.HTML` file to report status of the requested vs. retrieved data files
4. Retrieve each data file with a cURL command

Items 1 and 4 are described below in more detail.

The API Token

If the requested files include at least one with EAP protection, a MAST API token is required. As the following code snippet shows, the bash script will attempt to get the MAST token multiple ways.

Supplying the MAST Token

```
# Check for command-line argument
if [ -z "$1" ]
then
  # Check for environment variable
  if [ -z "$MAST_API_TOKEN" ]
  then
    # Prompt the user
    echo "Please enter your token here: "
    read MAST_API_TOKEN
  fi
else
  MAST_API_TOKEN=$1
fi
```

Retrieve Files

The following code snippet shows how each file is retrieved with cURL. For readability, the cURL command is re-formatted here as multi-line, with linux /MacOS escape characters.

cURL File Retrieval

```
# Announce the file
cat <<EOT
<<< Downloading
    File: mast:JWST/product/jw01409-o031_t014_nircam_clear-f212n_i2d.fits
      To: ${DOWNLOAD_FOLDER}/JWST/jw01409-o031_t014_nircam_clear-f212n/jw01409-o031_t014_nircam_clear-
f212n_i2d.fits
EOT
# Retrieve the file
curl -H "Authorization: token $MAST_API_TOKEN" \
     --globoff \
     --location-trusted \
     -f \
     --progress-bar \
     --create-dirs $CONT \
     --output ./${DOWNLOAD_FOLDER}'/JWST/jw01409-o031_t014_nircam_clear-f212n/jw00839-
o003_t001_miri_f560w_i2d.fits' \
'https://mast.stsci.edu/jwst/api/v0.1/Download/file?bundle_name=MAST_2022-04-30T2153&uri=mast:JWST/product
/jw01409-o031_t014_nircam_clear-f212n_i2d.fits'

# Announce the second file
cat <<EOT
<<< Downloading
    File: mast:JWST/product/jw01409-o031_t014_nircam_clear-f212n_cat.ecsv
    To: ${DOWNLOAD_FOLDER}/JWST/jw01409-o031_t014_nircam_clear-f212n/jw01409-o031_t014_nircam_clear-
f212n_cat.ecsv
EOT
# Retrieve the second file
curl -H "Authorization: token $MAST_API_TOKEN" \
     --globoff \
     --location-trusted \
     -f \
     --progress-bar \
     --create-dirs $CONT \
     --output ./${DOWNLOAD_FOLDER}'/JWST/jw01409-o031_t014_nircam_clear-f212n/jw01409-o031_t014_nircam_clear-
f212n_cat.ecsv' \
'https://mast.stsci.edu/jwst/api/v0.1/Download/file?bundle_name=MAST_2022-04-30T2153&uri=mast:JWST/product
/jw01409-o031_t014_nircam_clear-f212n_cat.ecsv'
```

It is worth calling out two important cURL command-line options:

- **`-H`**: pass custom header to the server. In this case, the MAST auth token
- **`--location-trusted`**: Follow re-directs, and send auth to other hosts

Interactive Python Example

In addition to requesting a cURL script from the MAST Portal, you can get one from the astroquery.mast Python API. We offer an interactive large download Jupyter Notebook that might serve as an example of how to construct such a query. Jupyter Notebooks are interactive, make it easy to follow along with code, and can be customized with ease; we highly suggest trying this tutorial in our Notebook.

For completeness, we also reproduce that code on this webpage. To begin, you must import the necessary packages. You'll need astroquery.mast to access the API, and astropy.table to handle the results, which are returned as a Table object. Then you can create a query, searching on any of the fiel ds listed on our API page. Here we'll query for JWST NIRCam observations, and specify a specific proposal ID.

Imports and Initial Query

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

matched_obs = Observations.query_criteria(
        obs_collection = 'JWST'
        , proposal_id = '1073'
        , instrument_name = 'NIRCam'
        )
```

The above code will return only matched *observations*. Each observation has a set of files associated with it; these may be guide-star images, uncalibrated exposures, or the final calibrated science product. In any case, we will need to retrieve all of these products before we can filter the results.

Requesting the products for many observations at once increases the risk of timeout errors. However, requesting products one at a time is often slow. The best balance is achieved by requesting products in groups of five.

Optimizing Product Requests

```
# Split the observations into "chunks" of size five
sz_chunk = 5
chunks = [matched_obs[i:i+sz_chunk] for i in range(0,len(matched_obs), sz_chunk)]

# Get the list of products for each chunk
t = [Observations.get_product_list(chunk) for chunk in chunks]

# Combine, and keep only the unique files
files = unique(vstack(t), keys='productFilename')
```

With our set of unique files, all that remains is to pass the table to the download products function. We also include the 'extension' filter to limit our download to .fits files. Any criteria listed on the products fields page may also be used.

Downloading Files

```
manifest = Observations.download_products(
           files
           , extension='fits'
           , curl_flag=True
                   )
```

For Further Reading...

- MAST Web Services