Getting started

This is a python interface for querying the ESO archive web service. For now, it supports the following:

- listing available instruments
- listing available surveys (phase 3)
- searching all instrument specific raw data: http://archive.eso.org/cms/eso-data/instrument-specific-query-forms.html
- searching data products (phase 3): http://archive.eso.org/wdb/wdb/adp/phase3_main/form
- downloading data by dataset identifiers: http://archive.eso.org/cms/eso-data/eso-data-direct-retrieval.html

Requirements

The following packages are required for the use of this module:

- keyring
- lxml
- requests >= 2.4.0

Authentication with ESO User Portal

Most of the datasets in the ESO Science Archive are public and can be downloaded anonymously without authenticating with the ESO User Portal (https://www.eso.org/sso/login). Data with restricted access like datasets under proprietary period can be downloaded by authorised users (for example PIs of the corresponding observing programmes and their delegates) after authentication with the ESO User Portal. This authentication is performed directly with the provided **login()** command, as illustrated in the example below. This method uses your keyring to securely store the password in your operating system. As such you should have to enter your correct password only once, and later be able to use this package for automated interaction with the ESO archive.

```
>>> from astroquery.eso import Eso
>>> eso = Eso()
>>> # First example: TEST is not a valid username, it will fail
>>> eso.login(username="TEST")
WARNING: No password was found in the keychain for the provided username. [astroquery.qu
TEST, enter your password:

INFO: Authenticating TEST on https://www.eso.org/sso ... [astroquery.eso.core]
ERROR: Authentication failed! [astroquery.eso.core]
>>> # Second example: pretend ICONDOR is a valid username
>>> eso.login(username="ICONDOR", store_password=True)
WARNING: No password was found in the keychain for the provided username. [astroquery.qu
ICONDOR, enter your password:

INFO: Authenticating ICONDOR on https://www.eso.org/sso ... [astroquery.eso.core]
INFO: Authentication successful! [astroquery.eso.core]
>>> # After the first login, your password has been stored
>>> eso.login(username="ICONDOR")
INFO: Authenticating ICONDOR on https://www.eso.org/sso ... [astroquer
INFO: Authentication successful! [astroquery.eso.core]
```

latest

```
>>> # Successful download of a public file (with or without login)
>>> eso.retrieve_data('AMBER.2006-03-14T07:40:19.830')
INFO: Downloading file 1/1 https://dataportal.eso.org/dataPortal/file/AMBER.2006-03-14T0
INFO: Successfully downloaded dataset AMBER.2006-03-14T07:40:19.830

>>> # Access denied to a restricted-access file (as anonymous user or as authenticated b
>>> eso.retrieve_data('ADP.2023-03-02T01:01:24.355')
INFO: Downloading file 1/1 https://dataportal.eso.org/dataPortal/file/ADP.2023-03-02T01:
ERROR: Access denied to https://dataportal.eso.org/dataPortal/file/ADP.2023-03-02T01:01:
```

Automatic password

As shown above, your password can be stored by the keyring module, if you pass the argument `store_password=True` to `Eso.login()`. For security reason, storing the password is turned off by default.

MAKE SURE YOU TRUST THE MACHINE WHERE YOU USE THIS FUNCTIONALITY!!!

NB: You can delete your password later with the command `keyring.delete_password('astroquery:www.eso.org', 'username')`.

Automatic login

You can further automate the authentication process by configuring a default username. The astroquery configuration file, which can be found following the procedure detailed in astropy.config, needs to be edited by adding `username = ICONDOR` in the `[eso]` section.

When configured, the username in the `login()` method call can be omitted as follows:

```
>>> from astroquery.eso import Eso
>>> eso = Eso()
>>> eso.login()
ICONDOR, enter your ESO password:
```

NB: If an automatic login is configured, other Eso methods can log you in automatically when needed.

Query the ESO archive for raw data

Identifying available instrument-specific queries

The direct retrieval of datasets is better explained with a running example, continuing from the authentication example above. The first thing to do is to identify the instrument to query. The list of available instrument-specific queries can be obtained with the **`list_instruments()`** method.

```
>>> from astroquery.eso import Eso
>>> eso = Eso()
>>> eso.list_instruments()
['fors1', 'fors2', 'sphere', 'vimos', 'omegacam', 'eris', 'hawki', 'isaac', 'naco', 'vis
 'vircam', 'apex', 'giraffe', 'uves', 'xshooter', 'espresso', 'muse', 'crires',
 'kmos', 'sinfoni', 'amber', 'gravity', 'matisse', 'midi', 'pionier', 'wlgsu']
```

In the example above, 22 instruments are available, they correspond to the instruments listed on the following web page: http://archive.eso.org/cms/eso-data/instrument-specific-query-forms.html.

latest

Inspecting available query options

Once an instrument is chosen, `midi` in our case, the query options for that instrument can be inspected by setting the `help=True` keyword of the **`query_instrument()`** method.

```
>>> eso.query_instrument('midi', help=True)
List of the column_filters parameters accepted by the midi instrument query.
The presence of a column in the result table can be controlled if prefixed with a [ ] ch
The default columns in the result table are shown as already ticked: [x].

Target Information
------------------
    target:
    resolver: simbad (SIMBAD name), ned (NED name), none (OBJECT as specified by the obs
    coord_sys: eq (Equatorial (FK5)), gal (Galactic)
    coord1:
    coord2:
    box:
    format: sexagesimal (Sexagesimal), decimal (Decimal)
[x] wdb_input_file:

Observation  and proposal parameters
------------------------------------
[ ] night:
    stime:
    starttime: 00 (00 hrs [UT]), 01 (01 hrs [UT]), 02 (02 hrs [UT]), 03 (03 hrs [UT]), 0
    etime:
    endtime: 00 (00 hrs [UT]), 01 (01 hrs [UT]), 02 (02 hrs [UT]), 03 (03 hrs [UT]), 04
[x] prog_id:
[ ] prog_type: % (Any), 0 (Normal), 1 (GTO), 2 (DDT), 3 (ToO), 4 (Large), 5 (Short), 6 (
[ ] obs_mode: % (All modes), s (Service), v (Visitor)
[ ] pi_coi:
    pi_coi_name: PI_only (as PI only), none (as PI or CoI)
[ ] prog_title:
...
```

Only the first two sections, of the parameters accepted by the `midi` instrument query, are shown in the example above: `Target Information` and `Observation and proposal parameters`.

As stated at the beginning of the help message, the parameters accepted by the query are given just before the first `:` sign (e.g. `target`, `resolver`, `stime`, `etime`...). When a parameter is prefixed by `[ ]`, the presence of the associated column in the query result can be controlled.

Note: the instrument query forms can be opened in your web browser directly using the `open_form` option of the **`query_instrument()`** method. This should also help with the identification of acceptable keywords.

Querying with constraints

It is now time to query the `midi` instrument for datasets. In the following example, observations of target `NGC 4151` between `2007-01-01` and `2008-01-01` are searched, and the query is configured to return the observation date column.

```
>>> table = eso.query_instrument('midi', column_filters={'target':'NGC 4151',
...                                                      'stime':'2007-01-01',
...                                                      'etime':'2008-01-01'},
...                              columns=['night'])
>>> print(len(table))
38
>>> print(table.columns)
<TableColumns names=('Release Date','Object','RA','DEC','Target Ra Dec','Target l b','D
```

latest

```
>>> table.pprint(max_width=100)
Release Date          Object             RA     ...       DPR.TECH       INS.MODE DIMM S
------------ ----------------------- ---------- ... -------------------- -------- ------
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
  2008-02-07                 NGC4151 182.635969 ...         IMAGE,WINDOW STARINTF     0.
         ...                     ...        ... ...                  ...      ...
  2007-02-07 SEARCH,OBJECT,DISPERSED 182.635969 ...       INTERFEROMETRY STARINTF     0.
  2007-02-07 SEARCH,OBJECT,DISPERSED 182.635969 ...       INTERFEROMETRY STARINTF     0.
  2007-02-07  TRACK,OBJECT,DISPERSED 182.635969 ...       INTERFEROMETRY STARINTF     0.
  2007-02-07  TRACK,OBJECT,DISPERSED 182.635969 ...       INTERFEROMETRY STARINTF     0.
  2007-02-07  TRACK,OBJECT,DISPERSED 182.635969 ...       INTERFEROMETRY STARINTF     0.
  2007-02-07       PHOTOMETRY,OBJECT 182.635969 ... IMAGE,WINDOW,CHOPNOD STARINTF     0.
  2007-02-07       PHOTOMETRY,OBJECT 182.635969 ... IMAGE,WINDOW,CHOPNOD STARINTF     0.
Length = 38 rows
```

And indeed, 38 datasets are found, and the `DATE OBS` column is in the result table.

Querying all instruments

The ESO database can also be queried without a specific instrument in mind. This is what the method **`query_main()`** is for. The associated query form on the ESO archive website is http://archive.eso.org/wdb/wdb/eso/eso_archive_main/form. Except for the keyword specifying the instrument the behaviour of **`query_main()`** is identical to **`query_instrument()`**.

ESO instruments without a specific query interface can be queried with **`query_main()`**, specifying the `instrument` constraint. This is the case of e.g. `harps`, `feros` or the all sky cameras APICAM and MASCOT. Here is an example to query all-sky images from APICAM with `luminance` filter.

```
>>> eso.ROW_LIMIT = -1   # Return all results
>>> table = eso.query_main(column_filters={'instrument': 'APICAM', 'filter_path': 'LUMIN
...                                        'stime':'2019-04-26', 'etime':'2019-04-27'},
>>> print(len(table))
207
>>> print(table.columns)
<TableColumns names=('OBJECT','RA','DEC','Program_ID','Instrument','Category','Type','Mo
>>> table.pprint(max_width=100)
 OBJECT      RA          DEC      Program_ID   ...   MJD-OBS    Airmass DIMM Seeing at Sta
------- ----------- ----------- ------------ ... ------------ ------- ------------------
ALL SKY 09:18:37.39 -24:32:32.7 60.A-9008(A) ... 58599.987766     1.0                  N
ALL SKY 09:21:07.68 -24:32:30.1 60.A-9008(A) ... 58599.989502     1.0                  N
ALL SKY 09:23:38.98 -24:32:27.5 60.A-9008(A) ...  58599.99125     1.0                  N
ALL SKY 09:26:10.28 -24:32:24.9 60.A-9008(A) ... 58599.992998     1.0                  N
ALL SKY 09:28:40.58 -24:32:22.4 60.A-9008(A) ... 58599.994734     1.0                  N
ALL SKY 09:31:43.93 -24:32:19.4 60.A-9008(A) ... 58599.996852     1.0                  N
ALL SKY 09:34:15.23 -24:32:17.0 60.A-9008(A) ...   58599.9986     1.0                  N
ALL SKY 09:36:47.53 -24:32:14.5 60.A-9008(A) ... 58600.000359     1.0                  N
ALL SKY 09:39:18.82 -24:32:12.2 60.A-9008(A) ... 58600.002106     1.0                  N
ALL SKY 09:41:49.11 -24:32:09.9 60.A-9008(A) ... 58600.003843     1.0
    ...         ...         ...          ... ...          ...     ...
ALL SKY 19:07:39.21 -24:39:35.1 60.A-9008(A) ... 58600.395914     1.0                  N
ALL SKY 19:10:11.68 -24:39:39.1 60.A-9008(A) ... 58600.397674     1.0                  N
```

latest

```
ALL SKY 19:12:44.15 -24:39:43.2 60.A-9008(A) ... 58600.399433     1.0                  N
ALL SKY 19:15:15.62 -24:39:47.1 60.A-9008(A) ... 58600.401181     1.0                  N
ALL SKY 19:17:46.09 -24:39:51.1 60.A-9008(A) ... 58600.402917     1.0                  N
ALL SKY 19:20:46.65 -24:39:55.8 60.A-9008(A) ...    58600.405     1.0                  N
ALL SKY 19:23:18.12 -24:39:59.7 60.A-9008(A) ... 58600.406748     1.0                  N
ALL SKY 19:25:51.60 -24:40:03.7 60.A-9008(A) ... 58600.408519     1.0                  N
ALL SKY 19:28:22.08 -24:40:07.6 60.A-9008(A) ... 58600.410255     1.0                  N
ALL SKY 19:30:52.55 -24:40:11.4 60.A-9008(A) ... 58600.411991     1.0                  N
Length = 207 rows
```

Query the ESO archive for reduced data

In addition to raw data, ESO makes available processed data. In this section, we show how to obtain these processed survey data from the archive.

Identify available surveys

The list of available surveys can be obtained with **astroquery.eso.EsoClass.list_surveys()** as follows:

```
>>> surveys = eso.list_surveys()
```

Query a specific survey with constraints

Let’s assume that we work with the `HARPS` survey, and that we are interested in target `HD203608`. The archive can be queried as follows:

```
>>> table = eso.query_surveys(surveys='HARPS', cache=False, target="HD203608")
```

The returned table has an `ARCFILE` column. It can be used to retrieve the datasets with **astroquery.eso.EsoClass.retrieve_data()** (see next section).

Obtaining extended information on data products

Only a small subset of the keywords presents in the data products can be obtained with **query_instrument()** or **query_main()**. There is however a way to get the full primary header of the FITS data products, using **get_headers()**. This method is detailed in the example below.

```
>>> table = eso.query_instrument('midi', column_filters={'target':'NGC 4151',
...                                                      'stime':'2007-01-01',
...                                                      'etime':'2008-01-01'},
...                              columns=['night'])
>>> table_headers = eso.get_headers(table['DP.ID'])
>>> table_headers.pprint()
           DP.ID             SIMPLE BITPIX ... HIERARCH ESO OCS TPL NFILE   HIERARCH ES
---------------------------- ------ ------ ... -------------------------- -------------
MIDI.2007-02-07T07:01:51.000   True     16 ...                          0
MIDI.2007-02-07T07:02:49.000   True     16 ...                          0
MIDI.2007-02-07T07:03:30.695   True     16 ...                          0
MIDI.2007-02-07T07:05:47.000   True     16 ...                          0
MIDI.2007-02-07T07:06:28.695   True     16 ...
MIDI.2007-02-07T07:09:03.000   True     16 ...
MIDI.2007-02-07T07:09:44.695   True     16 ...                          0
MIDI.2007-02-07T07:13:09.000   True     16 ...                          0
MIDI.2007-02-07T07:13:50.695   True     16 ...                          0
```

latest

```
MIDI.2007-02-07T07:15:55.000   True     16 ...                                  0
                         ...    ...    ... ...                                ...
MIDI.2007-02-07T07:52:27.992   True     16 ...                                  8 MIDI.2007-02-(
MIDI.2007-02-07T07:56:21.000   True     16 ...                                  0
MIDI.2007-02-07T07:57:35.485   True     16 ...                                  0
MIDI.2007-02-07T07:59:46.000   True     16 ...                                  0
MIDI.2007-02-07T08:01:00.486   True     16 ...                                  0
MIDI.2007-02-07T08:03:42.000   True     16 ...                                  8
MIDI.2007-02-07T08:04:56.506   True     16 ...                                  8
MIDI.2007-02-07T08:06:11.013   True     16 ...                                  8 MIDI.2007-02-(
MIDI.2007-02-07T08:08:19.000   True     16 ...                                  8 MIDI.2007-02-(
MIDI.2007-02-07T08:09:33.506   True     16 ...                                  8 MIDI.2007-02-(
Length = 38 rows
>>> len(table_headers.columns)
340
```

As shown above, for each data product ID (`DP.ID`), the full header (570 columns in our case) of the archive FITS file is collected. In the above table `table_headers` , there are as many rows as in the column `table['DP.ID']` .

Downloading datasets from the archive

Continuing from the query with constraints example, the first two datasets are selected, using their data product IDs `DP.ID` (or `ARCFILE` for surveys), and retrieved from the ESO archive.

```
>>> data_files = eso.retrieve_data(table['DP.ID'][:2])
INFO: Downloading datasets ...
INFO: Downloading 2 files ...
INFO: Downloading file 1/2 https://dataportal.eso.org/dataPortal/file/MIDI.2007-02-07T07
INFO: Successfully downloaded dataset MIDI.2007-02-07T07:01:51.000 to ...
INFO: Downloading file 2/2 https://dataportal.eso.org/dataPortal/file/MIDI.2007-02-07T07
INFO: Successfully downloaded dataset MIDI.2007-02-07T07:02:49.000 to ...
INFO: Uncompressing file /Users/szampier/.astropy/cache/astroquery/Eso/MIDI.2007-02-07T0
INFO: Uncompressing file /Users/szampier/.astropy/cache/astroquery/Eso/MIDI.2007-02-07T0
INFO: Done!
```

The file names, returned in data_files, points to the decompressed datasets (without the .Z extension) that have been locally downloaded. They are ready to be used with **fits**.

The default location (in the astropy cache) of the decompressed datasets can be adjusted by providing a `destination` keyword in the call to **retrieve_data()**.

By default, if a requested dataset is already found, it is not downloaded again from the archive. To force the retrieval of data that are present in the destination directory, use `continuation=True` in the call to **retrieve_data()**.

Troubleshooting

If you are repeatedly getting failed queries, or bad/out-of-date results, try clearing your cache:

```
>>> from astroquery.eso import Eso
>>> Eso.clear_cache()
```

If this function is unavailable, upgrade your version of astroquery. The `clear_cache` function was introduced in version 0.4.7.dev8479.

Reference/API

astroquery.eso Package

ESO service.

Classes

`EsoClass`()	
`Conf`()	Configuration parameters for `astroquery.eso`.

Created using Sphinx 8.2.3. Last built 26 Mar 2025.

Page Source Back to Top

latest