class astroquery.eso.**EsoClass** [source]

Bases: **QueryWithLogin**

Attributes Summary

AUTH_URL	
CALSELECTOR_URL	
DOWNLOAD_URL	
GUNZIP	
USERNAME	
maxrec	

Methods Summary

get_associated_files(datasets, *[, mode, ...])	Invoke Calselector service to find calibration files associated to the provided datasets.
get_headers(product_ids, *[, cache])	Get the headers associated to a list of data product IDs
list_instruments([cache])	List all the available instrument-specific queries offered by the ESO archive.
list_surveys(*[, cache])	List all the available surveys (phase 3) in the ESO archive.
login(*args[, username, store_password, ...])	Login to the ESO User Portal.
query_apex_quicklooks(*[, project_id, help, ...])	APEX data are distributed with quicklook products identified with a different name than other ESO products.
query_instrument(instrument, *[, ra, dec, ...])	Query instrument-specific raw data contained in the ESO archive.
query_main([instruments, ra, dec, radius, ...])	Query raw data from all instruments contained in the ESO archive.
query_surveys([surveys, ra, dec, radius, ...])	Query survey Phase 3 data contained in the ESO archive.
query_tap_service(query_str[, authenticated])	Query the ESO TAP service using a free ADQL string.
retrieve_data(datasets, *[, continuation, ...])	Retrieve a list of datasets form the ESO archive.

Attributes Documentation

AUTH_URL = '*https://www.eso.org/sso/oidc/token*'

CALSELECTOR_URL = '*https://archive.eso.org/calselector/v1/associations*'

DOWNLOAD_URL = '*https://dataportal.eso.org/dataPortal/file/*'

GUNZIP = '*gunzip*'

USERNAME = ''

maxrec

Methods Documentation

get_associated_files(*datasets: List[str], *, mode: str = 'raw', savexml: bool = False, destination: str | None = None*) → List[str] [source]

Invoke Calselector service to find calibration files associated to the provided datasets.

Parameters: **datasets** : *list of strings*
List of datasets for which calibration files should be retrieved.

mode : *string*
Calselector mode: 'raw' (default) for raw calibrations,
or 'processed' for processed calibrations.

savexml : *bool*
If true, save to disk the XML association tree returned by Calselector.

destination : *string*
Directory where the XML files are saved (default = astropy cache).

Returns: files
List of unique datasets associated to the input datasets.

get_headers(*product_ids, *, cache=True*) [source]

Get the headers associated to a list of data product IDs

This method returns a **Table** where the rows correspond to the provided data product IDs, and the columns are from each of the Fits headers keywords.

Note: The additional column `'DP.ID'` found in the returned table corresponds to the provided data product IDs.

Parameters: **product_ids** : *either a list of strings or a* ***Column***
List of data product IDs.

cache : *bool*
Defaults to True. If set overrides global caching behavior. See caching documentation.

Returns: **result** : ***Table***
A table where: columns are header keywords, rows are product_ids.

list_instruments(*cache=True*) → List[str] [source]

List all the available instrument-specific queries offered by the ESO archive.

Returns: **instrument_list** : *list of strings*

cache : *bool*
Deprecated - unused.

list_surveys(**, cache=True*) → List[str] [source]

List all the available surveys (phase 3) in the ESO archive.

Returns: **collection_list** : *list of strings*

cache : *bool*
Deprecated - unused.

login(**args*, *username: str = None*, *store_password: bool = False*, *reenter_password: bool = False*, ***kwargs*) → bool

Login to the ESO User Portal.

Parameters: **username** : *str, optional*

Username to the ESO Public Portal. If not given, it should be specified in the config file.

store_password : *bool, optional*

Stores the password securely in your keyring. Default is **False**.

reenter_password : *bool, optional*

Asks for the password even if it is already stored in the keyring. This is the way to overwrite an already stored passwork on the keyring. Default is **False**.

query_apex_quicklooks(***, *project_id=None*, *help: bool = False*, *column_filters: dict | None = None*, *open_form: bool = False*, *cache: bool = False*, ***kwargs*) [source]

APEX data are distributed with quicklook products identified with a different name than other ESO products. This query tool searches by project ID or any other supported keywords.

Examples

```
>>> tbl = ...
>>> files = ...
```

query_instrument(*instrument: str*, ***, *ra: float = None*, *dec: float = None*, *radius: float = None*, *columns: List | str = None*, *top: int = None*, *count_only: bool = False*, *query_str_only: bool = False*, *help: bool = False*, *authenticated: bool = False*, *column_filters: dict | None = None*, *open_form: bool = False*, *cache: bool = False*, ***kwargs*) → Table | int | str [source]

Query instrument-specific raw data contained in the ESO archive.

Parameters:

- **instrument** – Name of the instrument from which raw data is to be queried. Should be ONLY ONE of the names returned by **list_instruments()**.
- **ra** (*float*) – Cone Search Center - Right Ascension in degrees.
- **dec** (*float*) – Cone Search Center - Declination in degrees.
- **radius** (*float*) – Cone Search Radius in degrees.
- **columns** (*str or list of str*) – Name of the columns the query should return. If specified as a string, it should be a comma-separated list of column names.
- **top** (*int*) – When set to `N`, returns only the top `N` records.
- **count_only** (*bool*) – If **True**, returns only an **int**: the count of the records the query would return when set to **False**. Default is **False**.
- **query_str_only** (*bool*) – If **True**, returns only a **str**: the query string that would be issued to the TAP service. Default is **False**.
- **help** (*bool*) – If **True**, prints all the parameters accepted in `column_filters` and `columns`. Default is **False**.
- **authenticated** (*bool*) – If **True**, runs the query as an authenticated user. Authentication must be done beforehand via **login()**. Note that authenticated queries are slower. Default is **False**.
- **column_filters** (dict, **None**) – Constraints applied to the query. Default is **None**.
- **open_form** (*bool*) – **Deprecated** - unused.
- **cache** (*bool*) – **Deprecated** - unused.

Returns:
- By default, a **Table** containing records based on the specified columns and constraints. Returns **None** when the query has no results.
- When `count_only` is **True**, returns an **int** representing the record count for the specified filters.
- When `query_str_only` is **True**, returns the query string that would be issued to the TAP service given the specified arguments.

Return type: **Table**, **str**, **int**, or **None**

query_main(*instruments: List[str] | str = None, *, ra: float = None, dec: float = None, radius: float = None, columns: List | str = None, top: int = None, count_only: bool = False, query_str_only: bool = False, help: bool = False, authenticated: bool = False, column_filters: dict | None = None, open_form: bool = False, cache: bool = False, **kwargs*) → Table | int | str [source]

Query raw data from all instruments contained in the ESO archive.

Parameters:
- **instruments** (*str or list*) – Name of the instruments to filter. Should be one or more of the names returned by **list_instruments()**. If specified as a string, it should be a comma-separated list of instrument names. If not specified, returns records relative to all instruments. Default is **None**.
- **ra** (*float*) – Cone Search Center - Right Ascension in degrees.
- **dec** (*float*) – Cone Search Center - Declination in degrees.
- **radius** (*float*) – Cone Search Radius in degrees.
- **columns** (*str or list of str*) – Name of the columns the query should return. If specified as a string, it should be a comma-separated list of column names.
- **top** (*int*) – When set to `N`, returns only the top `N` records.
- **count_only** (*bool*) – If **True**, returns only an **int**: the count of the records the query would return when set to **False**. Default is **False**.
- **query_str_only** (*bool*) – If **True**, returns only a **str**: the query string that would be issued to the TAP service. Default is **False**.
- **help** (*bool*) – If **True**, prints all the parameters accepted in `column_filters` and `columns`. Default is **False**.
- **authenticated** (*bool*) – If **True**, runs the query as an authenticated user. Authentication must be done beforehand via **login()**. Note that authenticated queries are slower. Default is **False**.
- **column_filters** (dict, **None**) – Constraints applied to the query. Default is **None**.
- **open_form** (*bool*) – **Deprecated** - unused.
- **cache** (*bool*) – **Deprecated** - unused.

Returns:
- By default, a **Table** containing records based on the specified columns and constraints. Returns **None** when the query has no results.
- When `count_only` is **True**, returns an **int** representing the record count for the specified filters.
- When `query_str_only` is **True**, returns the query string that would be issued to the TAP service given the specified arguments.

Return type: **Table**, **str**, **int**, or **None**

query_surveys(*surveys: List[str] | str = None, *, ra: float = None, dec: float = None, radius: float = None, columns: List | str = None, top: int = None, count_only: bool = False, query_str_only: bool = False, help: bool = False, authenticated: bool = False, column_filters: dict | None = None, open_form: bool = False, cache: bool = False, **kwargs*) → Table | int | str [source]

Query survey Phase 3 data contained in the ESO archive.

Parameters:
- **surveys** (*str or list*) – Name of the survey(s) to query. Should be one or more of the names returned by `list_surveys()`. If specified as a string, it should be a comma-separated list of survey names. If not specified, returns records relative to all surveys. Default is `None`.
- **ra** (*float*) – Cone Search Center - Right Ascension in degrees.
- **dec** (*float*) – Cone Search Center - Declination in degrees.
- **radius** (*float*) – Cone Search Radius in degrees.
- **columns** (*str or list of str*) – Name of the columns the query should return. If specified as a string, it should be a comma-separated list of column names.
- **top** (*int*) – When set to `N`, returns only the top `N` records.
- **count_only** (*bool*) – If `True`, returns only an `int`: the count of the records the query would return when set to `False`. Default is `False`.
- **query_str_only** (*bool*) – If `True`, returns only a `str`: the query string that would be issued to the TAP service. Default is `False`.
- **help** (*bool*) – If `True`, prints all the parameters accepted in `column_filters` and `columns`. Default is `False`.
- **authenticated** (*bool*) – If `True`, runs the query as an authenticated user. Authentication must be done beforehand via `login()`. Note that authenticated queries are slower. Default is `False`.
- **column_filters** (dict, `None`) – Constraints applied to the query. Default is `None`.
- **open_form** (*bool*) – **Deprecated** - unused.
- **cache** (*bool*) – **Deprecated** - unused.

Returns:
- By default, a `Table` containing records based on the specified columns and constraints. Returns `None` when the query has no results.
- When `count_only` is `True`, returns an `int` representing the record count for the specified filters.
- When `query_str_only` is `True`, returns the query string that would be issued to the TAP service given the specified arguments.

Return type: `Table`, `str`, `int`, or `None`

query_tap_service(*query_str: str, authenticated: bool = False*) → Table | None [source]

Query the ESO TAP service using a free ADQL string.

Parameters:
- **query_str** (*str*) – The ADQL query string to be executed.
- **authenticated** (*bool*) – If `True`, the query is run as an authenticated user. Authentication must be done beforehand via `login()`. Note that authenticated queries are slower. Default is `False`.

Returns: The query results in an `Table`, or `None` if no data is found.

Return type: Optional[`Table`]

Example usage:

```
eso_instance = Eso()

eso_instance.query_tap_service("SELECT * FROM ivoa.ObsCore")
```

retrieve_data(*datasets, *, continuation=False, destination=None, with_calib=None, unzip=True, request_all_objects=None, request_id=None*) [source]

Retrieve a list of datasets form the ESO archive.

Parameters:

datasets : *list of strings or string*

List of datasets strings to retrieve from the archive.

destination: string

Directory where the files are copied. Files already found in the destination directory are skipped, unless continuation=True. Default to astropy cache.

continuation : *bool*

Force the retrieval of data that are present in the destination directory.

with_calib : *string*

Retrieve associated calibration files: None (default), 'raw' for raw calibrations, or 'processed' for processed calibrations.

unzip : *bool*

Unzip compressed files from the archive after download. `True` by default.

Returns:

files : *list of strings or string*

List of files that have been locally downloaded from the archive.

Examples

```
>>> dptbl = Eso.query_instrument('apex', pi_coi='ginsburg')
>>> dpids = [row['DP.ID'] for row in dptbl if 'Map' in row['Object']]
>>> files = Eso.retrieve_data(dpids)
```


Page Source Back to Top
Created using Sphinx 7.4.7. Last built 01 Apr 2025.