radiant_mlhub

Release 0.5.4

Radiant Earth Foundation

Nov 18, 2022

CONTENTS:

Welcome to the documentation for the Radiant MLHub Python client.

CHAPTER

ONE

GETTING STARTED

This guide will walk you through the basic usage of the `radiant_mlhub` library, including:

- Installing & configuring the library
- Discovering & fetching datasets
- Discovering & fetching collections
- Downloading dataset STAC catalog and assets

1.1 Background Info

If you have not already, browse Radiant MLHub to discover the datasets and ML models which are currently published on MLHub. Consider browsing the STAC specification to learn about SpatioTemporal Asset Catalogs (STAC). The MLHub API serves STAC Collections, Items and Assets.

1.2 Installation

1.2.1 Install with `pip`

```
$ pip install radiant_mlhub
```

1.2.2 Install with `conda`

```
$ conda install -c conda-forge radiant-mlhub
```

1.3 Configuration

If you have not done so already, you will need to register for an MLHub API key at https://mlhub.earth.

Once you have your API key, you will need to create a default profile by setting up a `.mlhub/profiles` file in your home directory. You can use the *mlhub configure* command line tool to do this:

```
$ mlhub configure
API Key: Enter your API key here...
Wrote profile to /home/user/.mlhub/profiles
```

Hint: If you do not have write access to the home directory on your machine, you can change the location of the `profiles` file using the `MLHUB_HOME` environment variables. For instance, setting `MLHUB_HOME=/tmp/some-directory/.mlhub` will cause the client to look for your profiles in a `/tmp/some-directory/.mlhub/profiles` file. You may want to permanently set this environment variable to ensure the client continues to look in the correct place for your profiles.

1.4 List Datasets

Once you have your profile configured, you can get a list of the available datasets from the Radiant MLHub API using the *Dataset.list* method. Remember that all datasets are also browseable and searchable on Radiant MLHub.

```
>>> from radiant_mlhub import Dataset
>>> datasets = Dataset.list()
>>> # print the first 5 datasets for example
>>> for dataset in datasets[0:5]:
...     print(dataset)
umd_mali_crop_type: 2019 Mali CropType Training Data
idiv_asia_crop_type: A crop type dataset for consistent land cover classification in␣
↪Central Asia
dlr_fusion_competition_germany: A Fusion Dataset for Crop Type Classification in Germany
ref_fusion_competition_south_africa: A Fusion Dataset for Crop Type Classification in␣
↪Western Cape, South Africa
bigearthnet_v1: BigEarthNet
```

1.5 Fetch a Dataset

You can also fetch a dataset by ID using the *Dataset.fetch* method. This method returns a *Dataset* instance.

```
>>> dataset = Dataset.fetch('bigearthnet_v1')
>>> print(dataset)
bigearthnet_v1: BigEarthNet V1
```

1.6 Work with Dataset Collections

Datasets have one or more collections associated with them. Collections fall into two types:

- `source_imagery`: Collections of source imagery associated with the dataset
- `labels`: Collections of labeled data associated with the dataset (these collections implement the STAC Label Extension)

To list all the collections associated with a dataset use the *collections* attribute.

```
>>> dataset.collections
[<Collection id=bigearthnet_v1_source>, <Collection id=bigearthnet_v1_labels>]
>>> type(dataset.collections[0])
radiant_mlhub.models.collection.Collection
```

You can also list the collections by type using the `collections.source_imagery` and `collections.labels` properties. This example code shows that collections are actually STAC objects.

```
>>> from pprint import pprint
>>> len(dataset.collections.source_imagery)
1
>>> source_collection = dataset.collections.source_imagery[0]
>>> pprint(source_collection.to_dict())
{'description': 'BigEarthNet v1.0',
'extent': {'spatial': {'bbox': [[-9.00023345437725,
                                 36.956956702083396,
                                 31.598439091981028,
                                 68.021682000472841]]},
           'temporal': {'interval': [['2017-06-13T10:10:31Z',
                                      '2018-05-29T11:54:01Z']]}},
'id': 'bigearthnet_v1_source',
'license': 'CDLA-Permissive-1.0',
'links': [{'href': 'https://api.radiant.earth/mlhub/v1/collections/bigearthnet_v1_source/
↪items',
           'rel': 'items',
           'type': 'application/geo+json'},
       {'href': 'https://api.radiant.earth/mlhub/v1/',
           'rel': 'parent',
           'type': 'application/json'},
       {'href': 'https://api.radiant.earth/mlhub/v1/',
           'rel': <RelType.ROOT: 'root'>,
           'title': 'Radiant MLHub API',
           'type': <MediaType.JSON: 'application/json'>},
       {'href': 'https://api.radiant.earth/mlhub/v1/collections/bigearthnet_v1_source',
           'rel': 'self',
           'type': 'application/json'}],
'providers': [{'name': 'BigEarthNet',
               'roles': ['processor', 'licensor'],
               'url': 'http://bigearth.net'}],
'sci:citation': 'G. Sumbul, M. Charfuelan, B. Demir, V. Markl, "BigEarthNet: '
               'A Large-Scale Benchmark Archive for Remote Sensing Image '
               'Understanding", IEEE International Geoscience and Remote '
               'Sensing Symposium, pp. 5901-5904, Yokohama, Japan, 2019.',
'sci:doi': '10.14279/depositonce-10149',
'stac_extensions': ['https://stac-extensions.github.io/scientific/v1.0.0/schema.json'],
'stac_version': '1.0.0',
'type': 'Collection'}
```

1.7 Download a Dataset

You can download a dataset's STAC catalog, and all of it's linked assets, using the *Dataset.download* method. Consider checking the dataset size before downloading. Here is an example dataset which is relatively small in size. The downloader can also scale up to the largest datasets.

```
>>> dataset = Dataset.fetch('nasa_marine_debris')
>>> print(dataset)
```

(continues on next page)

(continued from previous page)

```
nasa_marine_debris: Marine Debris Dataset for Object Detection in Planetscope Imagery
>>> print(dataset.stac_catalog_size)  # OK the STAC catalog archive is only ~260KB
263582
>>> print(dataset.estimated_dataset_size)  # OK the total dataset assets are ~77MB
77207762
>>> dataset.download()
nasa_marine_debris: fetch stac catalog: 258KB [00:00, 412.53KB/s]
INFO:radiant_mlhub.client.catalog_downloader:unarchive nasa_marine_debris.tar.gz ...
unarchive nasa_marine_debris.tar.gz: 100%|| 2830/2830 [00:00<00:00, 5772.09it/s]
INFO:radiant_mlhub.client.catalog_downloader:create stac asset list (please wait) ...
INFO:radiant_mlhub.client.catalog_downloader:2825 unique assets in stac catalog.
download assets: 100%|| 2825/2825 [03:27<00:00, 13.62it/s]
INFO:radiant_mlhub.client.catalog_downloader:assets saved to nasa_marine_debris
```

The `Dataset.download` method saves the STAC catalog and assets into your current working directory (by default).

The downloader has the ability to download in parallel with many cores, resume interrupted downloads, as well as options for filtering the assets to a more manageable size (highly recommended, depending on your application).

- *Filter by Collection and Asset Keys*
- *Filter by Temporal Range*
- *Filter by Bounding Box*
- *Filter by GeoJSON Area of Interest*

Hint: The *Datasets* guide has more downloading examples and the `Dataset.download` API reference is available as well.

Hint: The *Collections* guide has examples of downloading collection archives. Collection archives are not available for all collections, so consider using the Dataset downloader instead.

1.8 Discovering ML Models

ML Models are discoverable through the Python client as well. See the *ML Models* guide for more information.

CHAPTER

TWO

AUTHENTICATION

The Radiant MLHub API uses API keys to authenticate users. These keys must be passed as a `key` query parameter in any request made to the API. Anyone can register for an API key by going to https://mlhub.earth and creating an account. Once you have logged into your account, go to the Settings & API keys page at https://mlhub.earth/profile to create an API key.

2.1 Using API Keys

The best way to add your API key to requests is to create a *Session* instance using the *get_session()* helper function and making requests using this instance:

```
>>> from radiant_mlhub import get_session
>>> session = get_session()
>>> r = session.get(...)
```

You can associate an API key with a session in a number of ways:

- programmatically via an instantiation argument
- using environment variables
- using a named profile

The *Session* resolves an API key by trying each of the following (in this order):

1) Use an `api_key` argument provided during instantiation

```
>>> session = get_session(api_key='myapikey')
```

2) Use an `MLHUB_API_KEY` environment variable

```
>>> import os
>>> os.environ['MLHUB_API_KEY'] = 'myapikey'
>>> session = get_session()
```

3) Use a `profile` argument provided during instantiation (see *Using Profiles* section for details)

```
>>> session = get_session(profile='my-profile')
```

4) Use an `MLHUB_PROFILE` environment variable (see *Using Profiles* section for details)

```
>>> os.environ['MLHUB_PROFILE'] = 'my-profile'
>>> session = get_session()
```

5) Use the `default` profile (see *Using Profiles* section for details)

```
>>> session = get_session()
```

If none of the above strategies results in a valid API key, then an `APIKeyNotFound` exception is raised.

The *`radiant_mlhub.session.Session`* instance inherits from `requests.Session` and adds a few conveniences to a typical session:

- Adds the API key as a `key` query parameter
- Adds an `Accept: application/json` header
- Adds a `User-Agent` header that contains the package name and version, plus basic system information like the the OS name
- Prepends the MLHub root URL (`https://api.radiant.earth/mlhub/v1/`) to any request paths without a domain
- Raises a *`radiant_mlhub.exceptions.AuthenticationError`* for `401 (UNAUTHORIZED)` responses

2.2 Using Profiles

Profiles in `radiant_mlhub` are inspired by the Named Profiles used by `boto3` and `awscli`. These named profiles provide a way to store API keys (and potentially other configuration) on your local system so that you do not need to explicitly set environment variables or pass in arguments every time you create a session.

All profile configuration must be stored in a `.mlhub/profiles` file in your home directory. The `profiles` file uses the INI file structure supported by Python's `configparser` module as described here.

Hint: If you do not have write access to the home directory on your machine, you can change the location of the `profiles` file using the `MLHUB_HOME` environment variables. For instance, setting `MLHUB_HOME=/tmp/some-directory/.mlhub` will cause the client to look for your profiles in a `/tmp/some-directory/.mlhub/profiles` file. You may want to permanently set this environment variable to ensure the client continues to look in the correct place for your profiles.

The easiest way to configure a profile is using the `mlhub configure` CLI tool documented in the *CLI Tools section*:

```
$ mlhub configure
API Key: <Enter your API key when prompted>
Wrote profile to /home/user/.mlhub/profiles
```

Given the following `profiles` file...

```
[default]
api_key = default_api_key

[project1]
api_key = some_other_api_key

[project2]
api_key = yet_another_api_key
```

These would be the API keys used by sessions created using the various methods described in *Using API Keys*:

```
# As long as we haven't set the MLHUB_API_KEY or MLHUB_PROFILE environment variables
#  this will pull from the default profile
>>> session = get_session()
>>> session.params['key']
'default_api_key'

# Setting the MLHUB_PROFILE environment variable overrides the default profile
>>> os.environ['MLHUB_PROFILE'] = 'project1'
>>> session = get_session()
>>> session.params['key']
'some_other_api_key'

# Passing the profile argument directly overrides the MLHUB_PROFILE environment variable
>>> session = get_session(profile='profile2')
>>> session.params['key']
'yet_another_api_key'

# Setting the MLHUB_API_KEY environment variable overrides any profile-related arguments
>>> os.environ['MLHUB_API_KEY'] = 'environment_direct'
>>> session = get_session()
>>> session.params['key']
'environment_direct'

# Passing the api_key argument overrides all other strategies or finding the key
>>> session = get_session(api_key='argument_direct')
>>> session.params['key']
'argument_direct'
```

2.3 Making API Requests

Once you have your `profiles` file in place, you can create a session that will be used to make authenticated requests to the API:

```
>>> from radiant_mlhub import get_session
>>> session = get_session()
```

You can use this session to make authenticated calls to the API. For example, to list all collections:

```
>>> r = session.get('/collections')  # Leading slash is optional
>>> collections = r.json()['collections']
>>> print(len(collections))
47
```

2.3.1 Relative v. Absolute URLs

Any URLs that do not include a scheme (`http://`, `https://`) are assumed to be relative to the Radiant MLHub root URL. For instance, the following code would make a request to `https://api.radiant.earth/mlhub/v1/some-endpoint`:

```
>>> session.get('some-endpoint')
```

but the following code would make a request to `https://example.org`:

```
>>> session.get('https://example.org')
```

It is not recommended to make calls to APIs other than the Radiant MLHub API using these sessions.

CHAPTER

THREE

DATASETS

A **dataset** represents a group of one or more related STAC Collections. They group together any source imagery collections with the associated label collections to provide a convenient mechanism for accessing all of these data together. For instance, the `bigearthnet_v1_source` collection contains the source imagery for the BigEarthNet training dataset and, likewise, the `bigearthnet_v1_labels` collection contains the annotations for that same dataset. These two collections are grouped together into the `bigearthnet_v1` dataset.

Radiant MLHub provides an overview of the datasets available through the Radiant MLHub API along with dataset metadata and a listing of the associated collections.

To list and fetch datasets, the *Dataset* class is the recommended approach, but there are also low-level client methods from *radiant_mlhub.client*. Both methods are described below.

Hint: The Radiant MLHub web application provides an overview of all the datasets and collections available through the Radiant MLHub API.

Note: The objects returned by the Radiant MLHub API Dataset endpoints are not STAC-compliant objects and therefore the *Dataset* class described below is not a PySTAC object.

3.1 Discovering Datasets

You can discover datasets using the *Dataset.list* method. This method returns a list of *Dataset* instances.

```
>>> from radiant_mlhub import Dataset
>>> datasets = Dataset.list()
>>> for dataset in datasets[0:5]:  # print first 5 datasets, for example
>>>     print(dataset)
umd_mali_crop_type: 2019 Mali CropType Training Data
idiv_asia_crop_type: A crop type dataset for consistent land cover classification in␣
↪Central Asia
dlr_fusion_competition_germany: A Fusion Dataset for Crop Type Classification in Germany
ref_fusion_competition_south_africa: A Fusion Dataset for Crop Type Classification in␣
↪Western Cape, South Africa
bigearthnet_v1: BigEarthNet
```

The `list()` method also accepts `tags` and `text` arguments that can be used to filter datasets by their tags or a free text search, respectively. The `tags` argument may be either a single string or a list of strings. Only datasets that contain all of provided tags will be returned and these tags must be an *exact* match. The *text* argument may, similarly, be either a string or a list of strings. These will be used to search all of the text-based metadata fields for a dataset (e.g.

description, title, citation, etc.). Each argument is treated as a phrase by the text search engine and only datasets with matches for all of the provided phrases will be returned. So, for instance, `text=["maize", "rice"]` will return all datasets with either `"maize"` or `"rice"` somewhere in their text metadata, while `text=["maize rice"]` will not match any datasets. The search `text="land cover"` will return all datasets with the *phrase* `"land cover"` in their text metadata.

3.1.1 Low-level client

The Radiant MLHub `/datasets` endpoint returns a list of objects describing the available datasets and their associated collections. You can use the low-level *`list_datasets()`* function to work with these responses as native Python data types (`list` and `dict`).

```
>>> from radiant_mlhub.client import list_datasets
>>> from pprint import pprint
>>> datasets = list_datasets()
>>> first_dataset = datasets[0]
>>> pprint(first_dataset)
{'id': 'umd_mali_crop_type',
'title': '2019 Mali CropType Training Data',
...
```

3.2 Fetching Dataset Metadata

You can fetch a dataset from the Radiant MLHub API based on the dataset ID using the *`Dataset.fetch`* method. This method returns a *`Dataset`* instance. Fetching returns the metadata but does not download assets.

```
>>> dataset = Dataset.fetch_by_id('bigearthnet_v1')
>>> print(dataset.id)
bigearthnet_v1: BigEarthNet
```

If you would rather fetch the dataset using its DOI you can do so as well:

```
dataset = Dataset.fetch_by_doi("10.6084/m9.figshare.12047478.v2")
```

You can also use the more general *`Dataset.fetch`* method to get a dataset using either ID or DOI.

```
from radiant_mlhub.client import get_dataset
# These will all return the same dataset
dataset = Dataset.fetch("ref_african_crops_kenya_02")
dataset = Dataset.fetch("10.6084/m9.figshare.12047478.v2")
```

3.2.1 Low-level client

The Radiant MLHub `/datasets/{dataset_id}` endpoint returns an object representing a single dataset. You can use the low-level *`get_dataset()`* function to work with this response as a `dict`.

```
>>> from radiant_mlhub.client import get_dataset_by_id
>>> dataset = get_dataset_by_id('bigearthnet_v1')
>>> pprint(dataset)
{'collections': [{'id': 'bigearthnet_v1_source', 'types': ['source_imagery']},
```

(continues on next page)

(continued from previous page)

```
            {'id': 'bigearthnet_v1_labels', 'types': ['labels']}],
 'id': 'bigearthnet_v1',
 'title': 'BigEarthNet V1'}
```

3.3 Dataset Collections

If you are using the *Dataset* class, you can list the Collections associated with the dataset using the *Dataset.collections* property. This method returns a modified `list` that has 2 additional attributes: `source_imagery` and `labels`. You can use these attributes to list only the collections of a the associated type. All elements of these lists are instances of *Collection*. See the *Collections* documentation for details on how to work with these instances.

```
>>> len(first_dataset.collections)
2
>>> len(first_dataset.collections.source_imagery)
1
>>> first_dataset.collections.source_imagery[0].id
'umd_mali_crop_type_source'
>>> len(first_dataset.collections.labels)
1
>>> first_dataset.collections.labels[0].id
'umd_mali_crop_type_source'
```

Warning: There are rare cases of collections that contain both `source_imagery` and `labels` items (e.g. the SpaceNet collections). In these cases, the collection will be listed in both the `dataset.collections.labels` and `dataset.collections.source_imagery` lists, but *will only appear once in the main ``dataset.collections`` list*. This may cause what appears to be a mismatch in list lengths:

```
>>> len(dataset.collections.source_imagery) + len(dataset.collections.labels) ==␣
↪len(dataset.collections)
False
```

Note: Both the class methods and the low-level client functions accept keyword arguments that are passed directly to *get_session()* to create a session. See the *Authentication* documentation for details on how to use these arguments or configure the client to read your API key automatically.

3.4 Downloading Datasets

The dataset downloader offers download of STAC catalog archives, linked dataset assets, as well as partial downloads with filtering options.

- **Robustness**
  - Asset download resuming.
  - Retry and backoff for http error conditions.
  - Error reporting for unrecoverable download errors.

- **Performance**
  - Scales to millions of assets.
  - Multithreaded workers: parallel downloads.
- **Convenience**
  - STAC collection_id and item asset key filter
  - Temporal filter
  - Bounding box filter
  - GeoJSON intersection filter

3.4.1 Download All Assets

The most basic usage is to fetch a dataset, and then call it's download method. The output directory is the current working directory (by default).

```
>>> from radiant_mlhub import Dataset
>>> nasa_marine_debris = Dataset.fetch_by_id('nasa_marine_debris')
>>> print(nasa_marine_debris)
nasa_marine_debris: Marine Debris Dataset for Object Detection in Planetscope Imagery
>>> nasa_marine_debris.download()
nasa_marine_debris: fetch stac catalog: 258KB [00:00, 412.53KB/s]
INFO:radiant_mlhub.client.catalog_downloader:unarchive nasa_marine_debris.tar.gz ...
unarchive nasa_marine_debris.tar.gz: 100%|| 2830/2830 [00:00<00:00, 5772.09it/s]
INFO:radiant_mlhub.client.catalog_downloader:create stac asset list (please wait) ...
INFO:radiant_mlhub.client.catalog_downloader:2825 unique assets in stac catalog.
download assets: 100%|| 2825/2825 [03:27<00:00, 13.62it/s]
INFO:radiant_mlhub.client.catalog_downloader:assets saved to nasa_marine_debris
```

3.4.2 Download STAC Catalog Archive Only

If you want to inspect the STAC catalog or write your own download client for the assets just pass the `catalog_only` option to the download method:

```
>>> sen12floods.download(catalog_only=True)
sen12floods: fetch stac catalog: 2060KB [00:00, 127903.52KB/s]
INFO:radiant_mlhub.client.catalog_downloader:unarchive sen12floods.tar.gz...
unarchive sen12floods.tar.gz: 100%|| 22278/22278 [00:01<00:00, 14284.65it/s]
INFO:radiant_mlhub.client.catalog_downloader:catalog saved to /home/user/sen12floods
```

3.4.3 Logging

The Python logging module can be used to control the verbosity of the downloader. The default log level is INFO.

- Turn on WARNING level to see fewer log messages.
- Set DEBUG level to see more messages. This includes verbose HTTP-level log messages.

```
>>> import logging
>>> logging.basicConfig(level=logging.DEBUG)
>>> nasa_marine_debris.download()
...
DEBUG:radiant_mlhub.client.catalog_downloader:(thread id: 123145809592320) https://
↪radiantearth.blob.core.windows.net/mlhub/nasa-marine-debris/labels/20170326_153234_
↪0e26_17069-29758-16.npy -> .../nasa_marine_debris/nasa_marine_debris_labels/nasa_
↪marine_debris_labels_20170326_153234_0e26_17069-29758-16/pixel_bounds.npy
...
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): radiantearth.blob.core.
↪windows.net:443
DEBUG:urllib3.connectionpool:https://radiantearth.blob.core.windows.net:443 "HEAD /mlhub/
↪nasa-marine-debris/labels/20181031_095925_103b_32713-31765-16.npy HTTP/1.1" 200 0
...
(omitted many log messages here)
```

3.4.4 Output Directory

The output directory is by default, the current working directory. The `output_dir` parameter takes a `str` or `pathlib.Path`. It will be created if it does not exist.

```
# output_dir as string
nasa_marine_debris.download(output_dir='/tmp')

# output_dir as Path object
from pathlib import Path
nasa_marine_debris.download(output_dir=Path.home() / 'my_projects' / 'ml_datasets')
```

3.4.5 Large Dataset Performance

Let's try a bit larger dataset (tens of thousands of assets). After downloading the complete dataset, we'll explore all of the options for filtering assets. Filtering lets you limit the items and assets to those you are interested in prior to downloading.

This download example was run on a compute-optimized 16-core virtual machine in the MS Azure West-Europe region. You would likely experience slower download performance on your machine, depending on number of cores and network bandwidth.

```
>>> sen12floods = Dataset.fetch_by_id('sen12floods')
>>> %%time
>>> sen12floods.download()
sen12floods: fetch stac catalog: 2060KB [00:00, 127699.36KB/s]
INFO:radiant_mlhub.client.catalog_downloader:unarchive sen12floods.tar.gz...
unarchive sen12floods.tar.gz: 100%|| 22278/22278 [00:01<00:00, 14239.53it/s]
```

(continues on next page)

(continued from previous page)

```
INFO:radiant_mlhub.client.catalog_downloader:create stac asset list...
INFO:radiant_mlhub.client.catalog_downloader:39063 unique assets in stac catalog.
download assets: 100%|| 39063/39063 [06:26<00:00, 101.06it/s]
INFO:radiant_mlhub.client.catalog_downloader:assets saved to /home/user/sen12floods

CPU times: user 11min 44s, sys: 2min 15s, total: 14min
Wall time: 6min 40s
```

15GB of assets were downloaded into the `sen12floods/` directory. You may not necessarily want to download all of the assets in a dataset. In the following sections, all the filtering options are explained.

Hint: Download filters may be freely combined, except `bbox` and `intersects` which are independent options.

3.4.6 Checking Dataset Size

Consider checking the dataset size before downloading.

```
>>> dataset = Dataset.fetch('nasa_marine_debris')
>>> print(dataset)
nasa_marine_debris: Marine Debris Dataset for Object Detection in Planetscope Imagery
>>> print(dataset.stac_catalog_size)  # OK the STAC catalog archive is only ~260KB
263582
>>> print(dataset.estimated_dataset_size)  # OK the total dataset assets are ~77MB
77207762
```

3.4.7 Filter by Collection and Asset Keys

To download only the specified STAC collection ids and STAC item asset keys, create a dictionary in this format and pass it to the `collection_filter` parameter:

`{ collection_id1: [ asset_key1, asset_key2, ...], collection_id2: [asset_key1, asset_key2, ...] , ... }`

For example, using the `sen12floods` dataset, if we only wanted to download four bands of the source imagery:

```
my_filter = dict(
    sen12floods_s2_source=['B02', 'B03', 'B04', 'B08'],   # Red, Green, Blue, NIR
    sen12floods_s2_labels=['labels', 'documentation'],
)
sen12floods.download(collection_filter=my_filter)
```

3.4.8 Filter by Temporal Range

To download only STAC assets within a temporal range, use `datetime` parameter to specify a datetime range (tuple of `datetime` objects), or a single `datetime`

object).

```
from dateutil.parser import parse
my_start_date=parse("2019-04-01T00:00:00+0000")
my_end_date=parse("2019-04-07T00:00:00+0000")
sen12floods.download(datetime=(my_start_date, my_end_date))
```

3.4.9 Filter by Bounding Box

To download only STAC assets with a spatial bounding box, use the `bbox` parameter to specify a bounding box in lat/lng (CRS EPSG:4326). This performs a spatial intersection test with each STAC item's bounding box.

```
my_bbox = [-13.278254, 8.447033,
           -13.231551, 8.493532]
sen12floods.download(bbox=my_bbox)
```

Hint: The `bbox` filter may not be used with the `intersects` filter (use one or the other).

3.4.10 Filter by GeoJSON Area of Interest

To download only STAC assets within an area of interest, use the `intersects` parameter. This performs a spatial intersection test with each STAC item's bounding box.

```
import json
my_geojson = json.loads(
    """
    {
        "type": "Feature",
        "geometry": {
            "type": "Polygon",
            "coordinates": [
                [
                    [
                        -13.278048,
                        8.493532
                    ],
                    [
                        -13.278254,
                        8.447241
                    ],
                    [
                        -13.231762,
                        8.447033
                    ],
                    [
```

(continues on next page)

(continued from previous page)

```
                        -13.231551,
                        8.493323
                    ],
                    [
                        -13.278048,
                        8.493532
                    ]
                ]
            ]
        }
    }
    """
)
sen12floods.download(intersects=my_geojson)
```

Hint: The `intersects` filter may not be used with the `bbox` filter (use one or the other).

3.4.11 Error reporting

Any unrecoverable download errors will be logged to `{output_dir}/{dataset_id}/err_report.csv` and a Python exception will be raised.

3.4.12 Appendix: Default Filesystem Layout of Downloads

STAC archive file:

```
{output_dir}/{dataset_id}.tar.gz
```

Unarchived STAC catalog:

```
{output_dir}/{dataset_id}/catalog.json
```

Collection, Item and Asset layout:

```
{output_dir}/{dataset_id}/{collection_id}/{item_id}/{asset_key}.{ext}
```

Common Assets, ex: documentation.pdf are saved into a _common directory instead of duplicating them for many items:

```
{output_dir}/{dataset_id}/_common/{asset_key}.{ext}
```

Asset Database:

```
{output_dir}/{dataset_id}/mlhub_stac_assets.db
```

Error Report:

```
{output_dir}/{dataset_id}/err_report.csv
```

Hint: The `mlhub_stac_assets.db` file is an artifact which may be safely deleted to free up disk space.

CHAPTER

FOUR

COLLECTIONS

A **collection** represents either a group of related labels or a group of related source imagery for a given time period and geographic area. All collections in the Radiant MLHub API are valid STAC Collections. For instance, the `umd_mali_crop_type_source` collection catalogs the source imagery associated with the 2019 Mali CropType dataset, while the `umd_mali_crop_type_labels` collection catalogs the land cover labels associated with this imagery. These collections are considered part of a single `umd_mali_crop_type` **dataset** (see the *Datasets* documentation for details on working with datasets).

Hint: The Radiant MLHub web application provides an overview of all the datasets and collections available through the Radiant MLHub API.

Note: Collections are grouped into Datasets. See also the *Datasets* guide for more information about finding and downloading Datasets.

To list and fetch collections, the *Collection* class is the recommended approach, but there are also low-level client methods from *radiant_mlhub.client*. Both methods are described below.

4.1 Discovering Collections

You can discover collections using the *Collection.list* method. This method returns a list of *Collection* instances.

```
>>> from radiant_mlhub import Collection
>>> collections = Collection.list()
>>> first_collection = collections[0]
>>> print(first_collection)
ref_landcovernet_sa_v1_source_landsat_8: LandCoverNet South America Landsat 8 Source␣
↪Imagery
```

4.1.1 Low-level client

The Radiant MLHub `/collections` endpoint returns a list of objects describing the available collections. You can use the low-level *`list_collections()`* function to work with these responses as native Python data types (`list` and `dict`). This function returns a list of JSON-like dictionaries representing STAC Collections.

```
>>> from radiant_mlhub.client import list_collections
>>> from pprint import pprint
>>> collections = list_collections()
>>> first_collection = collections[0]
>>> pprint(first_collection)
{'description': 'LandCoverNet South America Landsat 8 Source Imagery',
 'id': 'ref_landcovernet_sa_v1_source_landsat_8',
 ...
```

4.2 Fetching Collection Metadata

You can fetch a collection from the Radiant MLHub API based on the collection ID using the *`Collection.fetch`* method. This is the recommended way of fetching a collection. This method returns a *`Collection`* instance. Fetching returns the metadata but does not download assets.

```
>>> collection = Collection.fetch('ref_african_crops_kenya_01_labels')
>>> print(collection)
ref_african_crops_kenya_01_labels: African Crops Kenya
```

For more information on a collection, you can browse to the MLHub page for the related dataset, for example:

```
>>> print(collection.registry_url)
https://registry.mlhub.earth/10.34911/rdnt.u41j87
```

Browse to https://registry.mlhub.earth/10.34911/rdnt.u41j87

4.2.1 Low-level client

The Radiant MLHub `/collections/{id}` endpoint returns an object representing a single collection's metadata. You can use the low-level *`get_collection()`* function to work with this response as a `dict`.

```
>>> from radiant_mlhub.client import get_collection
>>> collection = get_collection('ref_african_crops_kenya_01_labels')
>>> pprint(collection)
{'description': 'African Crops Kenya',
'extent': {'spatial': {'bbox': [[34.18191992149459,
                                 0.4724181558451209,
                                 34.3714943155646,
                                 0.7144217206851109]]},
           'temporal': {'interval': [['2018-04-10T00:00:00Z',
                                      '2020-03-13T00:00:00Z']]}},
'id': 'ref_african_crops_kenya_01_labels',
...
```

4.3 Downloading a Collection

Note: Not all collections have downloadable archives (depending on size). Consider instead using the dataset downloader functionality. The *Datasets* guide has more examples and the *Dataset.download* API reference is available as well.

You can download a collection archive using the *Collection.download* method. This is the recommended way of downloading a collection archive.

Hint: To check the existence, and size of the download archive without actually downloading it, you can use the `Collection.archive_size` property, which returns a size in bytes.

```
>>> collection = Collection.fetch('sn1_AOI_1_RIO')
>>> collection.archive_size
3504256089
>>> archive_path = collection.download('~/Downloads')
28%|       | 985.0/3496.9 [00:35<00:51, 48.31M/s]
>>> archive_path
PosixPath('/Users/someuser/Downloads/sn1_AOI_1_RIO.tar.gz')
```

If a file of the same name already exists, these methods will check whether the downloaded file is complete by comparing its size against the size of the remote file. If they are the same size, the download is skipped, otherwise the download will be resumed from the point where it stopped. You can control this behavior using the `if_exists` argument. Setting this to `"skip"` will skip the download for existing files *without* checking for completeness (a bit faster since it doesn't require a network request), and setting this to `"overwrite"` will overwrite any existing file.

Collection archives are gzipped tarballs. You can read more about the structure of these archives in this Medium post.

4.3.1 Low-level client

The Radiant MLHub `/archive/{archive_id}` endpoint allows you to download an archive of all assets associated with a given collection. You can use the low-level *download_collection_archive()* function to download the archive to your local file system.

```
>>> from radiant_mlhub.client import download_collection_archive
>>> archive_path = download_collection_archive('sn1_AOI_1_RIO')
28%|       | 985.0/3496.9 [00:35<00:51, 48.31M/s]
>>> archive_path
PosixPath('/path/to/current/directory/sn1_AOI_1_RIO.tar.gz')
```

CHAPTER

FIVE

ML MODELS

A **Model** represents a STAC Item implementing the ML Model extension. The goal of the ML Model Extension is to provide a way of cataloging machine learning models that operate on earth observation (EO) data described as a STAC catalog.

To discover and fetch models you can either use the *MLModel* class or the low-level client methods from *radiant_mlhub.client*. Using the *MLModel* class is the recommended approach, but both methods are described below.

Hint: The Radiant MLHub web application provides an overview of all the ML models available through the Radiant MLHub API.

5.1 Discovering Models

You can discover models using the *MLModel.list* method. This method returns a list of *MLModel* instances.

```
>>> from radiant_mlhub import MLModel
>>> models = MLModel.list()
>>> first_model = models[0]
>>> for model in models[0:2]:  # print first two models, for example
>>>     print(model)
model-crop-detection-torchgeo-v1: A Spatio-Temporal Deep Learning-Based Crop␣
↪Classification Model for Satellite Imagery
model-cyclone-wind-estimation-torchgeo-v1: Tropical Cyclone Wind Estimation Model
```

You can fetch a model by ID using *MLModel.fetch* method.

```
>>> model = MLModel.fetch('model-cyclone-wind-estimation-torchgeo-v1')
>>> model.assets
.. {'inferencing-compose': <Asset href=https://raw.githubusercontent.com/RadiantMLHub/
↪cyclone-model-torchgeo/main/inferencing.yml>,
.. 'inferencing-checkpoint': <Asset href=https://zenodo.org/record/5773331/files/last.
↪ckpt?download=1>}
>>> len(first_model.links)
8
>>> # print only the ml-model and mlhub related links
>>> from pprint import pprint
>>> pprint([ link for link in first_model.links if 'ml-model:' in link.rel or 'mlhub:'␣
↪in link.rel])
```

(continues on next page)

(continued from previous page)

```
.. [<Link rel=ml-model:inferencing-image target=docker://docker.io/radiantearth/crop-
↪detection-dl:1>,
.. <Link rel=ml-model:train-data target=https://api.radiant.earth/mlhub/v1/collections/
↪ref_african_crops_kenya_02_source>,
.. <Link rel=ml-model:train-data target=https://api.radiant.earth/mlhub/v1/collections/
↪ref_african_crops_kenya_02_labels>,
.. <Link rel=mlhub:training-dataset target=https://mlhub.earth/data/ref_african_crops_
↪kenya_02>]
>>> # you can access rest of properties as a dict
>>> first_model.properties.keys()
.. dict_keys(['title', 'license', 'sci:doi', 'datetime', 'providers', 'description',
↪'end_datetime', 'sci:citation', 'ml-model:type', 'start_datetime', 'sci:publications',
↪'ml-model:training-os', 'ml-model:architecture', 'ml-model:prediction_type', 'ml-
↪model:learning_approach', 'ml-model:training-processor-type'])
```

5.1.1 Low-level Client

The Radiant MLHub `/models` endpoint returns a list of objects describing the available models and their properties. You can use the low-level *`list_models()`* function to work with these responses as native Python data types (`list` and `dict`).

```
>>> from radiant_mlhub.client import list_models
>>> models = list_models()
>>> first_model = models[0]
>>> first_model.keys()
dict_keys(['id', 'bbox', 'type', 'links', 'assets', 'geometry', 'collection', 'properties
↪', 'stac_version', 'stac_extensions'])
>>> first_model['id']
'model-cv4a-crop-detection-v1'
>>> first_model['properties'].keys()
dict_keys(['title', 'license', 'sci:doi', 'datetime', 'providers', 'description', 'end_
↪datetime', 'sci:citation', 'ml-model:type', 'start_datetime', 'sci:publications', 'ml-
↪model:training-os', 'ml-model:architecture', 'ml-model:prediction_type', 'ml-
↪model:learning_approach', 'ml-model:training-processor-type'])
```

5.2 Fetching Model Metadata

The Radiant MLHub `/models/{model_id}` endpoint returns an object representing a single model. You can use the low-level *`get_model_by_id()`* function to work with this response as a `dict`.

```
>>> from radiant_mlhub.client import get_model_by_id
>>> model = get_model_by_id('model-cyclone-wind-estimation-torchgeo-v1')
>>> model.keys()
dict_keys(['id', 'bbox', 'type', 'links', 'assets', 'geometry', 'collection', 'properties
↪', 'stac_version', 'stac_extensions'])
```

You can also fetch a model from the Radiant MLHub API based on the model ID using the *`MLModel.fetch`* method. This is the recommended way of fetching a model. This method returns a *`MLModel`* instance.

```
>>> from radiant_mlhub import MLModel
>>> model = MLModel.fetch('model-cyclone-wind-estimation-torchgeo-v1')
>>> model.id
'model-cyclone-wind-estimation-torchgeo-v1'
>>> len(model.assets)
2
>>> len(model.links)
8
```

CHAPTER

SIX

RADIANT_MLHUB PACKAGE

6.1 Subpackages

6.1.1 radiant_mlhub.client package

Submodules

radiant_mlhub.client.catalog_downloader module

class radiant_mlhub.client.catalog_downloader.**AssetRecord**(**args*, ***kwargs*)

Bases: dict

A stac_assets db record.

asset_key: Optional[str]

asset_save_path: Optional[str]

asset_url: str

bbox_json: Optional[str]

collection_id: Optional[str]

common_asset: bool

end_datetime: Optional[datetime.datetime]

filtered: bool

geometry_json: Optional[str]

item_id: Optional[str]

rowid: Optional[int]

single_datetime: Optional[datetime.datetime]

start_datetime: Optional[datetime.datetime]

radiant_mlhub.client.catalog_downloader.**COMMON_ASSET_NAMES** = ['documentation', 'readme', 'test_split', 'train_split', 'validation_split']

Common assets will be put into *_common* and only downloaded once.

class radiant_mlhub.client.catalog_downloader.**CatalogDownloader**(*config:* radiant_mlhub.client.catalog_downloader.CatalogDown

Bases: object

asset_dir: pathlib.Path

catalog_file: pathlib.Path

config: *radiant_mlhub.client.catalog_downloader.CatalogDownloaderConfig*

db_conn: sqlite3.Connection

db_cur: sqlite3.Cursor

err_report: _io.TextIOWrapper

err_report_path: pathlib.Path

err_writer: Any

work_dir: pathlib.Path

class radiant_mlhub.client.catalog_downloader.**CatalogDownloaderConfig**(*, *api_key: str = None*, *bbox: Optional[Union[Tuple[float], List[float]]] = None*, *catalog_only: bool = False*, *collection_filter: Dict[str, List[str]] = None*, *dataset_id: str*, *if_exists:* radiant_mlhub.if_exists.DownloadIfExistsOpts *= DownloadIfExistsOpts.resume*, *intersects: Dict[str, Any] = None*, *output_dir: pathlib.Path*, *asset_output_dir: pathlib.Path = None*, *profile: str = None*, *mlhub_api_session:* radiant_mlhub.session.Session, *temporal_query: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]] = None*)

Bases: pydantic.main.BaseModel

Configuration model & validator for CatalogDownloader.

class Config

Bases: object

arbitrary_types_allowed = True

api_key: Optional[str]

asset_output_dir: Optional[pathlib.Path]

bbox: Optional[Union[Tuple[float], List[float]]]

catalog_only: bool

collection_filter: Optional[Dict[str, List[str]]]

dataset_id: str

if_exists: *radiant_mlhub.if_exists.DownloadIfExistsOpts*

intersects: Optional[Dict[str, Any]]

mlhub_api_session: *radiant_mlhub.session.Session*

Requests session for mlhub api calls.

output_dir: pathlib.Path

profile: Optional[str]

temporal_query: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]]

radiant_mlhub.client.collections module

radiant_mlhub.client.collections.**get_collection**(*collection_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /collections/{p1}` endpoint.

See the MLHub API docs for details.

Parameters

- **collection_id** (*str*) – The ID of the collection to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns collection

Return type dict

Raises

- ***EntityDoesNotExist*** – If a 404 response code is returned by the API
- ***MLHubException*** – If any other response code is returned

radiant_mlhub.client.collections.**get_collection_item**(*collection_id: str*, *item_id: str*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /collections/{p1}/items/{p2}` endpoint.

Parameters

- **collection_id** (*str*) – The ID of the Collection to which the Item belongs.

- **item_id** (*str*) – The ID of the Item.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns item

Return type dict

radiant_mlhub.client.collections.**list_collection_items**(*collection_id: str*, ***, *page_size: Optional[int] = None*, *extensions: Optional[List[str]] = None*, *limit: int = 10*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Iterator[Dict[str, Any]]

Yields JSON-like dictionaries representing STAC Item objects returned by the Radiant MLHub `GET /collections/{collection_id}/items` endpoint.

Note: Because some collections may contain hundreds of thousands of items, this function limits the total number of responses to `10` by default. You can change this value by increasing the value of the `limit` keyword argument, or setting it to `None` to list all items. **Be aware that trying to list all items in a large collection may take a very long time.**

Parameters

- **collection_id** (*str*) – The ID of the collection from which to fetch items
- **page_size** (*int*) – The number of items to return in each page. If set to `None`, then this parameter will not be passed to the API and the default API value will be used (currently `30`).
- **extensions** (*list*) – If provided, then only items that support all of the extensions listed will be returned.
- **limit** (*int*) – The maximum *total* number of items to yield. Defaults to `10`.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Yields **item** (*dict*) – JSON-like dictionary representing a STAC Item associated with the given collection.

radiant_mlhub.client.collections.**list_collections**(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing STAC Collection objects returned by the Radiant MLHub `GET /collections` endpoint.

See the MLHub API docs for details.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collections** – List of JSON-like dictionaries representing STAC Collection objects.

Return type List[dict]

radiant_mlhub.client.datasets module

radiant_mlhub.client.datasets.**download_collection_archive**(*archive_id: str*, *output_dir: Optional[Union[pathlib.Path, str]] = None*, ***, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pathlib.Path

Downloads the archive with the given ID to an output location (current working directory by default).

The `if_exists` argument determines how to handle an existing archive file in the output directory. The default behavior (defined by `if_exists="resume"`) is to resume the download by requesting a byte range starting at the size of the existing file. If the existing file is the same size as the file to be downloaded (as determined by the `Content-Length` header), then the download is skipped. You can automatically skip download using `if_exists="skip"` (this may be faster if you know the download was not interrupted, since no network request is made to get the archive size). You can also overwrite the existing file using `if_exists="overwrite"`.

Parameters

- **archive_id** (*str*) – The ID of the archive to download. This is the same as the Collection ID.
- **output_dir** (*Path*) – Path to which the archive will be downloaded. Defaults to the current working directory.
- **if_exists** (*str, optional*) – How to handle an existing archive at the same location. If `"skip"`, the download will be skipped. If `"overwrite"`, the existing file will be overwritten and the entire file will be re-downloaded. If `"resume"` (the default), the existing file size will be compared to the size of the download (using the `Content-Length` header). If the existing file is smaller, then only the remaining portion will be downloaded. Otherwise, the download will be skipped.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **output_path** – The full path to the downloaded archive file.

Return type Path

Raises **ValueError** – If `if_exists` is not one of `"skip"`, `"overwrite"`, or `"resume"`.

radiant_mlhub.client.datasets.**get_catalog_info**(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Gets info for the given archive from the `/catalog/{dataset_id}/info` endpoint as a JSON-like dictionary.

The JSON object returned by the API has the following properties:

- `dataset`: ID of the dataset that this archive's Collection belongs to.
- `stac_catalog_size`: size of the dataset_id.tar.gz STAC archive (in bytes)
- `estimated_dataset_size`: size in bytes of entire dataset (estimated)

Parameters

- **dataset_id** (*str*) – The ID of the dataset
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **archive_info** – JSON-like dictionary representing the API response.

Return type dict

radiant_mlhub.client.datasets.**get_collection_archive_info**(*archive_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Gets info for the given archive from the `/archive/{archive_id}/info` endpoint as a JSON-like dictionary.

The JSON object returned by the API has the following properties:

- `collection`: The ID of the Collection that this archive is associated with.
- `dataset`: The ID of the dataset that this archive's Collection belongs to.
- `size`: The size of the archive (in bytes)
- `types`: The types associated with this archive's Collection. Will be one of `"source_imagery"` or `"label"`.

Parameters

- **archive_id** (*str*) – The ID of the archive. This is the same as the Collection ID.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **archive_info** – JSON-like dictionary representing the API response.

Return type dict

radiant_mlhub.client.datasets.**get_dataset**(*dataset_id_or_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing a dataset by first trying to look up the dataset by ID, then falling back to finding the dataset by DOI.

See the MLHub API docs for details.

Parameters

- **dataset_id_or_doi** (*str*) – The ID of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **dataset**

Return type dict

radiant_mlhub.client.datasets.**get_dataset_by_doi**(*dataset_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /datasets/doi/{dataset_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **dataset_doi** (*str*) – The DOI of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type dict

radiant_mlhub.client.datasets.**get_dataset_by_id**(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /datasets/{dataset_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **dataset_id** (*str*) – The ID of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type dict

radiant_mlhub.client.datasets.**list_datasets**(***, *tags: Optional[Union[str, Iterable[str]]] = None*, *text: Optional[Union[str, Iterable[str]]] = None*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing dataset objects returned by the Radiant MLHub `GET /datasets` endpoint.

See the MLHub API docs for details.

Parameters

- **tags** (A tag or list of tags to filter datasets by. If not `None`, only datasets) – containing all provided tags will be returned.
- **text** (A text phrase or list of text phrases to filter datasets by. If not `None`,) – only datasets containing all phrases will be returned.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns datasets

Return type List[dict]

radiant_mlhub.client.datetime_utils module

radiant_mlhub.client.datetime_utils.**one_to_one_check**(*d1: datetime.datetime*, *d2: datetime.datetime*) → bool

Compare two dates for equality.

radiant_mlhub.client.datetime_utils.**one_to_range_check**(*d1: datetime.datetime*, *d2: Tuple[datetime.datetime, datetime.datetime]*) → bool

Check for overlap: single datetime with date range.

radiant_mlhub.client.datetime_utils.**range_to_range_check**(*d1: Tuple[datetime.datetime, datetime.datetime]*, *d2: Tuple[datetime.datetime, datetime.datetime]*) → bool

Check for overlap: two date ranges.

radiant_mlhub.client.ml_models module

radiant_mlhub.client.ml_models.**get_model_by_id**(*model_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /models/{model_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **model_id** (*str*) – The ID of the ML Model to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns model

Return type dict

radiant_mlhub.client.ml_models.**list_models**(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing ML Model objects returned by the Radiant MLHub `GET /models` endpoint.

See the MLHub API docs for details.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns models

Return type List[dict]

radiant_mlhub.client.resumable_downloader module

class radiant_mlhub.client.resumable_downloader.**ResumableDownloader**(*url: str*, *out_file: pathlib.Path*, *desc: Optional[str] = None*, *session: Optional[requests.sessions.Session] = None*, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.overwrite*, *disable_progress_bar: bool = True*, *chunk_size: int = 1024*, *chunk_unit: str = 'KB'*)

Bases: object

Resumable downloader, for a single file.

- Similar to datasets._download_collection_archive_chunked(), but this is not parallelized.
- Supports DownloadIfExistsOpts.
- Displays progress bar (optional).

chunk_size: int

chunk_unit: str

desc: Optional[str]

disable_progress_bar: bool

if_exists: *radiant_mlhub.if_exists.DownloadIfExistsOpts*

out_file: pathlib.Path

run() → None

session: requests.sessions.Session

url: str

Module contents

Low-level functions for making requests to MLHub API and Blob Storage endpoints.

class radiant_mlhub.client.**CatalogDownloader**(*config: radiant_mlhub.client.catalog_downloader.CatalogDownloaderConfig*)

Bases: object

asset_dir: pathlib.Path

catalog_file: pathlib.Path

config: *radiant_mlhub.client.catalog_downloader.CatalogDownloaderConfig*

db_conn: sqlite3.Connection

db_cur: sqlite3.Cursor

err_report: _io.TextIOWrapper

err_report_path: pathlib.Path

err_writer: Any

work_dir: pathlib.Path

class radiant_mlhub.client.CatalogDownloaderConfig(*, *api_key: str = None*, *bbox: Optional[Union[Tuple[float], List[float]]] = None*, *catalog_only: bool = False*, *collection_filter: Dict[str, List[str]] = None*, *dataset_id: str*, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.resume*, *intersects: Dict[str, Any] = None*, *output_dir: pathlib.Path*, *asset_output_dir: pathlib.Path = None*, *profile: str = None*, *mlhub_api_session: radiant_mlhub.session.Session*, *temporal_query: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]] = None*)

Bases: pydantic.main.BaseModel

Configuration model & validator for CatalogDownloader.

class Config

Bases: object

arbitrary_types_allowed = True

api_key: Optional[str]

asset_output_dir: Optional[pathlib.Path]

bbox: Optional[Union[Tuple[float], List[float]]]

catalog_only: bool

collection_filter: Optional[Dict[str, List[str]]]

dataset_id: str

if_exists: *radiant_mlhub.if_exists.DownloadIfExistsOpts*

intersects: Optional[Dict[str, Any]]

mlhub_api_session: *radiant_mlhub.session.Session*

Requests session for mlhub api calls.

output_dir: pathlib.Path

profile: Optional[str]

temporal_query: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]]

radiant_mlhub.client.**download_collection_archive**(*archive_id: str*, *output_dir: Optional[Union[pathlib.Path, str]] = None*, ***, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pathlib.Path

Downloads the archive with the given ID to an output location (current working directory by default).

The `if_exists` argument determines how to handle an existing archive file in the output directory. The default behavior (defined by `if_exists="resume"`) is to resume the download by requesting a byte range starting at the size of the existing file. If the existing file is the same size as the file to be downloaded (as determined by the `Content-Length` header), then the download is skipped. You can automatically skip download using `if_exists="skip"` (this may be faster if you know the download was not interrupted, since no network request is made to get the archive size). You can also overwrite the existing file using `if_exists="overwrite"`.

Parameters

- **archive_id** (*str*) – The ID of the archive to download. This is the same as the Collection ID.
- **output_dir** (*Path*) – Path to which the archive will be downloaded. Defaults to the current working directory.
- **if_exists** (*str, optional*) – How to handle an existing archive at the same location. If `"skip"`, the download will be skipped. If `"overwrite"`, the existing file will be overwritten and the entire file will be re-downloaded. If `"resume"` (the default), the existing file size will be compared to the size of the download (using the `Content-Length` header). If the existing file is smaller, then only the remaining portion will be downloaded. Otherwise, the download will be skipped.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **output_path** – The full path to the downloaded archive file.

Return type Path

Raises **ValueError** – If `if_exists` is not one of `"skip"`, `"overwrite"`, or `"resume"`.

radiant_mlhub.client.**get_catalog_info**(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Gets info for the given archive from the `/catalog/{dataset_id}/info` endpoint as a JSON-like dictionary.

The JSON object returned by the API has the following properties:

- `dataset`: ID of the dataset that this archive's Collection belongs to.
- `stac_catalog_size`: size of the dataset_id.tar.gz STAC archive (in bytes)
- `estimated_dataset_size`: size in bytes of entire dataset (estimated)

Parameters

- **dataset_id** (*str*) – The ID of the dataset
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **archive_info** – JSON-like dictionary representing the API response.

Return type dict

radiant_mlhub.client.**get_collection**(*collection_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /collections/{p1}` endpoint.

See the MLHub API docs for details.

Parameters

- **collection_id** (*str*) – The ID of the collection to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collection**

Return type dict

Raises

- ***EntityDoesNotExist*** – If a 404 response code is returned by the API
- ***MLHubException*** – If any other response code is returned

radiant_mlhub.client.**get_collection_archive_info**(*archive_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Gets info for the given archive from the `/archive/{archive_id}/info` endpoint as a JSON-like dictionary.

The JSON object returned by the API has the following properties:

- `collection`: The ID of the Collection that this archive is associated with.
- `dataset`: The ID of the dataset that this archive's Collection belongs to.
- `size`: The size of the archive (in bytes)
- `types`: The types associated with this archive's Collection. Will be one of `"source_imagery"` or `"label"`.

Parameters

- **archive_id** (*str*) – The ID of the archive. This is the same as the Collection ID.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **archive_info** – JSON-like dictionary representing the API response.

Return type dict

radiant_mlhub.client.**get_collection_item**(*collection_id: str*, *item_id: str*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /collections/{p1}/items/{p2}` endpoint.

Parameters

- **collection_id** (*str*) – The ID of the Collection to which the Item belongs.
- **item_id** (*str*) – The ID of the Item.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns item

Return type dict

radiant_mlhub.client.**get_dataset**(*dataset_id_or_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing a dataset by first trying to look up the dataset by ID, then falling back to finding the dataset by DOI.

See the MLHub API docs for details.

Parameters

- **dataset_id_or_doi** (*str*) – The ID of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type dict

radiant_mlhub.client.**get_dataset_by_doi**(*dataset_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /datasets/doi/{dataset_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **dataset_doi** (*str*) – The DOI of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type dict

radiant_mlhub.client.**get_dataset_by_id**(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /datasets/{dataset_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **dataset_id** (*str*) – The ID of the dataset to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable

- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type dict

radiant_mlhub.client.**get_model_by_id**(*model_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Dict[str, Any]

Returns a JSON-like dictionary representing the response from the Radiant MLHub `GET /models/{model_id}` endpoint.

See the MLHub API docs for details.

Parameters

- **model_id** (*str*) – The ID of the ML Model to fetch
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns model

Return type dict

radiant_mlhub.client.**list_collection_items**(*collection_id: str*, ***, *page_size: Optional[int] = None*, *extensions: Optional[List[str]] = None*, *limit: int = 10*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Iterator[Dict[str, Any]]

Yields JSON-like dictionaries representing STAC Item objects returned by the Radiant MLHub `GET /collections/{collection_id}/items` endpoint.

Note: Because some collections may contain hundreds of thousands of items, this function limits the total number of responses to `10` by default. You can change this value by increasing the value of the `limit` keyword argument, or setting it to `None` to list all items. **Be aware that trying to list all items in a large collection may take a very long time.**

Parameters

- **collection_id** (*str*) – The ID of the collection from which to fetch items
- **page_size** (*int*) – The number of items to return in each page. If set to `None`, then this parameter will not be passed to the API and the default API value will be used (currently `30`).
- **extensions** (*list*) – If provided, then only items that support all of the extensions listed will be returned.
- **limit** (*int*) – The maximum *total* number of items to yield. Defaults to `10`.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Yields **item** (*dict*) – JSON-like dictionary representing a STAC Item associated with the given collection.

radiant_mlhub.client.**list_collections**(*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing STAC Collection objects returned by the Radiant MLHub `GET /collections` endpoint.

See the MLHub API docs for details.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collections** – List of JSON-like dictionaries representing STAC Collection objects.

Return type List[dict]

radiant_mlhub.client.**list_datasets**(*, *tags: Optional[Union[str, Iterable[str]]] = None*, *text: Optional[Union[str, Iterable[str]]] = None*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing dataset objects returned by the Radiant MLHub `GET /datasets` endpoint.

See the MLHub API docs for details.

Parameters

- **tags** (A tag or list of tags to filter datasets by. If not `None`, only datasets) – containing all provided tags will be returned.
- **text** (A text phrase or list of text phrases to filter datasets by. If not `None`,) – only datasets containing all phrases will be returned.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **datasets**

Return type List[dict]

radiant_mlhub.client.**list_models**(*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[Dict[str, Any]]

Gets a list of JSON-like dictionaries representing ML Model objects returned by the Radiant MLHub `GET /models` endpoint.

See the MLHub API docs for details.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **models**

Return type List[dict]

6.1.2 radiant_mlhub.models package

Submodules

radiant_mlhub.models.collection module

Extensions of the PySTAC classes that provide convenience methods for interacting with the Radiant MLHub API.

class radiant_mlhub.models.collection.**Collection**(*id: str*, *description: str*, *extent: pystac.collection.Extent*, *title: Optional[str] = None*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, *catalog_type: Optional[pystac.catalog.CatalogType] = None*, *license: str = 'proprietary'*, *keywords: Optional[List[str]] = None*, *providers: Optional[List[pystac.provider.Provider]] = None*, *summaries: Optional[pystac.summaries.Summaries] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: pystac.collection.Collection

Class inheriting from pystac.Collection that adds some convenience methods for listing and fetching from the Radiant MLHub API.

property **archive_size**: Optional[int]

The size of the tarball archive for this collection in bytes (or None if the archive does not exist).

download(*output_dir: Union[str, pathlib.Path]*, ***, *if_exists:* radiant_mlhub.if_exists.DownloadIfExistsOpts = *DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pathlib.Path

Downloads the archive for this collection to an output location (current working directory by default). If the parent directories for output_path do not exist, they will be created.

The if_exists argument determines how to handle an existing archive file in the output directory. See the documentation for the download_archive() function for details. The default behavior is to resume downloading if the existing file is incomplete and skip the download if it is complete.

Note: Some collections may be very large and take a significant amount of time to download, depending on your connection speed.

Parameters

- **output_dir** (*Path*) – Path to a local directory to which the file will be downloaded. File name will be generated automatically based on the download URL.
- **if_exists** (*str, optional*) – How to handle an existing archive at the same location. If "skip", the download will be skipped. If "overwrite", the existing file will be overwritten and the entire file will be re-downloaded. If "resume" (the default), the existing file size will be compared to the size of the download (using the Content-Length header). If the existing file is smaller, then only the remaining portion will be downloaded. Otherwise, the download will be skipped.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable

- **profile** (*str*) – A profile to use when making this request.

Returns **output_path** – The path to the downloaded archive file.

Return type pathlib.Path

Raises **FileExistsError** – If file at `output_path` already exists and both `exist_okay` and `overwrite` are `False`.

classmethod **fetch**(*collection_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Creates a *Collection* instance by fetching the collection with the given ID from the Radiant MLHub API.

Parameters

- **collection_id** (*str*) – The ID of the collection to fetch (e.g. `bigearthnet_v1_source`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collection**

Return type *Collection*

fetch_item(*item_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pystac.item.Item

classmethod **from_dict**(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Patches the `pystac.Collection.from_dict()` method so that it returns the calling class instead of always returning a `pystac.Collection` instance.

get_items(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Iterator[pystac.item.Item]

Note: The `get_items` method is not implemented for Radiant MLHub *Collection* instances for performance reasons. Please use the `Dataset.download()` method to download Dataset assets.

Raises **NotImplementedError** –

classmethod **list**(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Collection*]

Returns a list of *Collection* instances for all collections hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collections**

Return type List[*Collection*]

property registry_url: Optional[str]

The URL of the registry page for this Collection. The URL is based on the DOI identifier for the collection. If the Collection does not have a "sci:doi" property then registry_url will be None.

radiant_mlhub.models.dataset module

Extensions of the PySTAC classes that provide convenience methods for interacting with the Radiant MLHub API.

class radiant_mlhub.models.dataset.CollectionType(*value*)

Bases: enum.Enum

Valid values for the type of a collection associated with a Radiant MLHub dataset.

LABELS = 'labels'

SOURCE = 'source_imagery'

class radiant_mlhub.models.dataset.Dataset(*id: str*, *collections: List[Dict[str, Any]]*, *title: Optional[str] = None*, *registry: Optional[str] = None*, *doi: Optional[str] = None*, *citation: Optional[str] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, ***_: Any*)

Bases: object

Class that brings together multiple Radiant MLHub "collections" that are all considered part of a single "dataset". For instance, the bigearthnet_v1 dataset is composed of both a source imagery collection (bigearthnet_v1_source) and a labels collection (bigearthnet_v1_labels).

id

The dataset ID.

Type str

title

The title of the dataset (or None if dataset has no title).

Type str or None

registry_url

The URL to the registry page for this dataset, or None if no registry page exists.

Type str or None

doi

The DOI identifier for this dataset, or None if there is no DOI for this dataset.

Type str or None

citation

The citation information for this dataset, or None if there is no citation information.

Type str or None

property collections: radiant_mlhub.models.dataset._CollectionList

List of collections associated with this dataset. The list that is returned has 2 additional attributes (source_imagery and labels) that represent the list of collections corresponding the each type.

Note: This is a cached property, so updating `self.collection_descriptions` after calling `self.collections` the first time will have no effect on the results. See `functools.cached_property()` for details on clearing the cached value.

Examples

```
>>> from radiant_mlhub import Dataset
>>> dataset = Dataset.fetch('bigearthnet_v1')
>>> len(dataset.collections)
2
>>> len(dataset.collections.source_imagery)
1
>>> len(dataset.collections.labels)
1
```

To loop through all collections

```
>>> for collection in dataset.collections:
...     # Do something here
```

To loop through only the source imagery collections:

```
>>> for collection in dataset.collections.source_imagery:
...     # Do something here
```

To loop through only the label collections:

```
>>> for collection in dataset.collections.labels:
...     # Do something here
```

download(*output_dir: Union[pathlib.Path, str] = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/radiant-mlhub/checkouts/v0.5.4/docs/source')*, ***, *asset_output_dir: Optional[Union[pathlib.Path, str]] = None*, *catalog_only: bool = False*, *if_exists:* radiant_mlhub.if_exists.DownloadIfExistsOpts *= DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, *bbox: Optional[List[float]] = None*, *intersects: Optional[Dict[str, Any]] = None*, *datetime: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]] = None*, *collection_filter: Optional[Dict[str, List[str]]] = None*) → None

Downloads dataset's STAC catalog and all linked assets. The download may be customized and controlled by providing bbox, intersects, datetime, and filter options.

Parameters

- **output_dir** (*str or pathlib.Path*) – The directory into which the STAC catalog will be written. If no asset_output_dir is specified, the assets will also be saved to the output_dir. Defaults to current working directory.
- **asset_output_dir** (*Otional[str, pathlib.Path]*) – The directory into which the archives will be written. If not defined by the user, the assets are saved to their respective asset level STAC catalog directories in the output_dir, which is in the current working directory by default.
- **catalog_only** (*bool*) – If True, the STAC catalog will be downloaded and unarchived, but no assets wll be downloaded. Defaults to False.

- **if_exists** (*Optional[str]*) – Allowed values: *skip*, *overwrite*, or *resume* (default).
- **bbox** (*Optional[List[float]]*) – List representing a bounding box of coordinates, for spatial intersection filter. Must be in CRS EPSG:4326.
- **intersects** (*Optional[GeoJSON]*) – GeoJSON object for spatial intersects filter. Must be a parsed GeoJSON dict with a *geometry* property.
- **datetime** (*Optional[datetime, Tuple[datetime, datetime]]*) – Single datetime or datetime range for temporal filter.
- **collection_filter** (*Optional[Dict[str, list]]*) – Mapping of collection_id and asset keys to include (exclusively).

  examples:

  – **download will only include this collection:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=[])*

  – **download will only include this collection and only these asset keys:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=["B02", "B03", "B04"])*
- **api_key** (*Optional[str]*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable.
- **profile** (*Optional[str]*) – Authentication Profile to use when making this request.

Raises

- **IOError** – If `output_dir` exists and is not a directory. If unrecoverable download errors occurred.
- **ValueError** – If provided filters are incompatible, for example bbox and intersects.
- **RuntimeError** – If filters result in zero assets to download.

Any unrecoverable download errors will be logged to *{output_dir}/{dataset_id}/err_report.csv*.

property estimated_dataset_size: Optional[int]

Size in bytes of entire dataset (bytes)

classmethod fetch(*dataset_id_or_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by first trying to fetching the dataset based on ID, then falling back to fetching by DOI.

Parameters

- **dataset_id_or_doi** (*str*) – The ID or DOI of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_doi(*dataset_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by fetching the dataset with the given DOI from the Radiant MLHub API.

Parameters

- **dataset_doi** (*str*) – The DOI of the dataset to fetch (e.g. `10.6084/m9.figshare.12047478.v2`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_id(*dataset_id: str*, *, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a `Dataset` instance by fetching the dataset with the given ID from the Radiant MLHub API.

Parameters

- **dataset_id** (*str*) – The ID of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod list(*, *tags: Optional[Union[str, Iterable[str]]] = None*, *text: Optional[Union[str, Iterable[str]]] = None*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Dataset*]

Returns a list of `Dataset` instances for each datasets hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **tags** (A list of tags to filter datasets by. If not `None`, only datasets containing all) – provided tags will be returned.
- **text** (A list of text phrases to filter datasets by. If not `None`, only datasets) – containing all phrases will be returned.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Yields **dataset** (*Dataset*)

property stac_catalog_size: Optional[int]

Size of the dataset_id.tar.gz STAC archive (bytes)

radiant_mlhub.models.ml_model module

Extensions of the PySTAC classes that provide convenience methods for interacting with the Radiant MLHub API.

class radiant_mlhub.models.ml_model.**MLModel**(*id: str*, *geometry: Optional[Dict[str, Any]]*, *bbox: Optional[List[float]]*, *datetime: Optional[datetime.datetime]*, *properties: Dict[str, Any]*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *collection: Optional[Union[str, pystac.collection.Collection]] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: pystac.item.Item

assets: Dict[str, Asset]

Dictionary of Asset objects, each with a unique key.

bbox: Optional[List[float]]

Bounding Box of the asset represented by this item using either 2D or 3D geometries. The length of the array is 2*n where n is the number of dimensions. Could also be None in the case of a null geometry.

collection: Optional[*Collection*]

Collection to which this Item belongs, if any.

collection_id: Optional[str]

The Collection ID that this item belongs to, if any.

datetime: Optional[Datetime]

Datetime associated with this item. If None, then start_datetime and end_datetime in common_metadata will supply the datetime range of the Item.

extra_fields: Dict[str, Any]

Extra fields that are part of the top-level JSON fields the Item.

classmethod fetch(*model_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Fetches a *MLModel* instance by id.

Parameters

- **model_id** (*str*) – The ID of the ML Model to fetch (e.g. model-cyclone-wind-estimation-torchgeo-v1).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns model

Return type *MLModel*

classmethod from_dict(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Patches the pystac.Item.from_dict() method so that it returns the calling class instead of always returning a pystac.Item instance.

geometry: Optional[Dict[str, Any]]

Defines the full footprint of the asset represented by this item, formatted according to RFC 7946, section 3.1 (GeoJSON).

id: str

Provider identifier. Unique within the STAC.

links: List[Link]

A list of Link objects representing all links associated with this Item.

classmethod list(*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*radiant_mlhub.models.ml_model.MLModel*]

Returns a list of *MLModel* instances for all models hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns models

Return type List[*MLModel*]

properties: Dict[str, Any]

A dictionary of additional metadata for the Item.

session_kwargs: Dict[str, Any] = {}

Class inheriting from pystac.Item that adds some convenience methods for listing and fetching from the Radiant MLHub API.

stac_extensions: List[str]

List of extensions the Item implements.

Module contents

Extensions of the PySTAC classes that provide convenience methods for interacting with the Radiant MLHub API.

class radiant_mlhub.models.**Collection**(*id: str*, *description: str*, *extent: pystac.collection.Extent*, *title: Optional[str] = None*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, *catalog_type: Optional[pystac.catalog.CatalogType] = None*, *license: str = 'proprietary'*, *keywords: Optional[List[str]] = None*, *providers: Optional[List[pystac.provider.Provider]] = None*, *summaries: Optional[pystac.summaries.Summaries] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: pystac.collection.Collection

Class inheriting from pystac.Collection that adds some convenience methods for listing and fetching from the Radiant MLHub API.

property archive_size: Optional[int]

The size of the tarball archive for this collection in bytes (or None if the archive does not exist).

download(*output_dir: Union[str, pathlib.Path]*, ***, *if_exists:* radiant_mlhub.if_exists.DownloadIfExistsOpts = *DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pathlib.Path

Downloads the archive for this collection to an output location (current working directory by default). If the parent directories for `output_path` do not exist, they will be created.

The `if_exists` argument determines how to handle an existing archive file in the output directory. See the documentation for the `download_archive()` function for details. The default behavior is to resume downloading if the existing file is incomplete and skip the download if it is complete.

Note: Some collections may be very large and take a significant amount of time to download, depending on your connection speed.

Parameters

- **output_dir** (*Path*) – Path to a local directory to which the file will be downloaded. File name will be generated automatically based on the download URL.
- **if_exists** (*str, optional*) – How to handle an existing archive at the same location. If `"skip"`, the download will be skipped. If `"overwrite"`, the existing file will be overwritten and the entire file will be re-downloaded. If `"resume"` (the default), the existing file size will be compared to the size of the download (using the `Content-Length` header). If the existing file is smaller, then only the remaining portion will be downloaded. Otherwise, the download will be skipped.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **output_path** – The path to the downloaded archive file.

Return type pathlib.Path

Raises **FileExistsError** – If file at `output_path` already exists and both `exist_okay` and `overwrite` are `False`.

classmethod fetch(*collection_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Creates a *Collection* instance by fetching the collection with the given ID from the Radiant MLHub API.

Parameters

- **collection_id** (*str*) – The ID of the collection to fetch (e.g. `bigearthnet_v1_source`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collection**

Return type *Collection*

fetch_item(*item_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pystac.item.Item

classmethod from_dict(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Patches the `pystac.Collection.from_dict()` method so that it returns the calling class instead of always returning a `pystac.Collection` instance.

get_items(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Iterator[pystac.item.Item]

Note: The `get_items` method is not implemented for Radiant MLHub *`Collection`* instances for performance reasons. Please use the *`Dataset.download()`* method to download Dataset assets.

Raises **NotImplementedError** –

classmethod list(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Collection*]

Returns a list of *`Collection`* instances for all collections hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **collections**

Return type List[*Collection*]

property registry_url: Optional[str]

The URL of the registry page for this Collection. The URL is based on the DOI identifier for the collection. If the Collection does not have a `"sci:doi"` property then `registry_url` will be `None`.

class radiant_mlhub.models.**Dataset**(*id: str*, *collections: List[Dict[str, Any]]*, *title: Optional[str] = None*, *registry: Optional[str] = None*, *doi: Optional[str] = None*, *citation: Optional[str] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, ***_: Any*)

Bases: `object`

Class that brings together multiple Radiant MLHub "collections" that are all considered part of a single "dataset". For instance, the `bigearthnet_v1` dataset is composed of both a source imagery collection (`bigearthnet_v1_source`) and a labels collection (`bigearthnet_v1_labels`).

id

The dataset ID.

Type str

title

The title of the dataset (or `None` if dataset has no title).

Type str or None

registry_url

The URL to the registry page for this dataset, or `None` if no registry page exists.

Type str or None

doi

The DOI identifier for this dataset, or `None` if there is no DOI for this dataset.

Type str or None

citation

The citation information for this dataset, or `None` if there is no citation information.

Type str or None

property collections: radiant_mlhub.models.dataset._CollectionList

List of collections associated with this dataset. The list that is returned has 2 additional attributes (`source_imagery` and `labels`) that represent the list of collections corresponding the each type.

Note: This is a cached property, so updating `self.collection_descriptions` after calling `self.collections` the first time will have no effect on the results. See `functools.cached_property()` for details on clearing the cached value.

Examples

```
>>> from radiant_mlhub import Dataset
>>> dataset = Dataset.fetch('bigearthnet_v1')
>>> len(dataset.collections)
2
>>> len(dataset.collections.source_imagery)
1
>>> len(dataset.collections.labels)
1
```

To loop through all collections

```
>>> for collection in dataset.collections:
...     # Do something here
```

To loop through only the source imagery collections:

```
>>> for collection in dataset.collections.source_imagery:
...     # Do something here
```

To loop through only the label collections:

```
>>> for collection in dataset.collections.labels:
...     # Do something here
```

download(*output_dir: Union[pathlib.Path, str] = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/radiant-mlhub/checkouts/v0.5.4/docs/source')*, ***, *asset_output_dir: Optional[Union[pathlib.Path, str]] = None*, *catalog_only: bool = False*, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, *bbox: Optional[List[float]] = None*, *intersects: Optional[Dict[str, Any]] = None*, *datetime: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]] = None*, *collection_filter: Optional[Dict[str, List[str]]] = None*) → None

Downloads dataset's STAC catalog and all linked assets. The download may be customized and controlled by providing bbox, intersects, datetime, and filter options.

Parameters

- **output_dir** (*str or pathlib.Path*) – The directory into which the STAC catalog will be written. If no asset_output_dir is specified, the assets will also be saved to the output_dir. Defaults to current working directory.
- **asset_output_dir** (*Otional[str, pathlib.Path]*) – The directory into which the archives will be written. If not defined by the user, the assets are saved to their respective asset level STAC catalog directories in the output_dir, which is in the current working directory by default.
- **catalog_only** (*bool*) – If True, the STAC catalog will be downloaded and unarchived, but no assets wll be downloaded. Defaults to False.
- **if_exists** (*Optional[str]*) – Allowed values: *skip*, *overwrite*, or *resume* (default).
- **bbox** (*Optional[List[float]]*) – List representing a bounding box of coordinates, for spatial intersection filter. Must be in CRS EPSG:4326.
- **intersects** (*Optional[GeoJSON]*) – GeoJSON object for spatial intersects filter. Must be a parsed GeoJSON dict with a *geometry* property.
- **datetime** (*Optional[datetime, Tuple[datetime, datetime]]*) – Single datetime or datetime range for temporal filter.
- **collection_filter** (*Optional[Dict[str, list]]*) – Mapping of collection_id and asset keys to include (exclusively).

  examples:

  - **download will only include this collection:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=[])*
  - **download will only include this collection and only these asset keys:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=["B02", "B03", "B04"])*
- **api_key** (*Optional[str]*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable.
- **profile** (*Optional[str]*) – Authentication Profile to use when making this request.

Raises

- **IOError** – If `output_dir` exists and is not a directory. If unrecoverable download errors occurred.
- **ValueError** – If provided filters are incompatible, for example bbox and intersects.
- **RuntimeError** – If filters result in zero assets to download.

Any unrecoverable download errors will be logged to *{output_dir}/{dataset_id}/err_report.csv*.

property estimated_dataset_size: Optional[int]

Size in bytes of entire dataset (bytes)

classmethod fetch(*dataset_id_or_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by first trying to fetching the dataset based on ID, then falling back to fetching by DOI.

Parameters

- **dataset_id_or_doi** (*str*) – The ID or DOI of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_doi(*dataset_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a `Dataset` instance by fetching the dataset with the given DOI from the Radiant MLHub API.

Parameters

- **dataset_doi** (*str*) – The DOI of the dataset to fetch (e.g. `10.6084/m9.figshare.12047478.v2`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_id(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a `Dataset` instance by fetching the dataset with the given ID from the Radiant MLHub API.

Parameters

- **dataset_id** (*str*) – The ID of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod list(***, *tags: Optional[Union[str, Iterable[str]]] = None*, *text: Optional[Union[str, Iterable[str]]] = None*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Dataset*]

Returns a list of `Dataset` instances for each datasets hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **tags** (A list of tags to filter datasets by. If not `None`, only datasets containing all) – provided tags will be returned.
- **text** (A list of text phrases to filter datasets by. If not `None`, only datasets) – containing all phrases will be returned.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable

- **profile** (*str*) – A profile to use when making this request.

Yields **dataset** (*Dataset*)

property stac_catalog_size: Optional[int]

Size of the dataset_id.tar.gz STAC archive (bytes)

class radiant_mlhub.models.**MLModel**(*id: str*, *geometry: Optional[Dict[str, Any]]*, *bbox: Optional[List[float]]*, *datetime: Optional[datetime.datetime]*, *properties: Dict[str, Any]*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *collection: Optional[Union[str, pystac.collection.Collection]] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: pystac.item.Item

assets: Dict[str, Asset]

Dictionary of Asset objects, each with a unique key.

bbox: Optional[List[float]]

Bounding Box of the asset represented by this item using either 2D or 3D geometries. The length of the array is 2*n where n is the number of dimensions. Could also be None in the case of a null geometry.

collection: Optional[*Collection*]

Collection to which this Item belongs, if any.

collection_id: Optional[str]

The Collection ID that this item belongs to, if any.

datetime: Optional[Datetime]

Datetime associated with this item. If None, then start_datetime and end_datetime in common_metadata will supply the datetime range of the Item.

extra_fields: Dict[str, Any]

Extra fields that are part of the top-level JSON fields the Item.

classmethod fetch(*model_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Fetches a *MLModel* instance by id.

Parameters

- **model_id** (*str*) – The ID of the ML Model to fetch (e.g. model-cyclone-wind-estimation-torchgeo-v1).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **model**

Return type *MLModel*

classmethod from_dict(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Patches the pystac.Item.from_dict() method so that it returns the calling class instead of always returning a pystac.Item instance.

geometry: Optional[Dict[str, Any]]

Defines the full footprint of the asset represented by this item, formatted according to RFC 7946, section 3.1 (GeoJSON).

id: str

Provider identifier. Unique within the STAC.

links: List[Link]

A list of Link objects representing all links associated with this Item.

classmethod list(*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*radiant_mlhub.models.ml_model.MLModel*]

Returns a list of *MLModel* instances for all models hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns models

Return type List[*MLModel*]

properties: Dict[str, Any]

A dictionary of additional metadata for the Item.

session_kwargs: Dict[str, Any] = {}

Class inheriting from pystac.Item that adds some convenience methods for listing and fetching from the Radiant MLHub API.

stac_extensions: List[str]

List of extensions the Item implements.

6.2 Submodules

6.3 radiant_mlhub.exceptions module

exception radiant_mlhub.exceptions.**APIKeyNotFound**

Bases: *radiant_mlhub.exceptions.MLHubException*

Raised when an API key cannot be found using any of the strategies described in the *Authentication* docs.

exception radiant_mlhub.exceptions.**AuthenticationError**

Bases: *radiant_mlhub.exceptions.MLHubException*

Raised when the Radiant MLHub API cannot authenticate the request, either because the API key is invalid or expired, or because no API key was provided in the request.

exception radiant_mlhub.exceptions.**EntityDoesNotExist**

Bases: *radiant_mlhub.exceptions.MLHubException*

Raised when attempting to fetch a collection that does not exist in the Radiant MLHub API.

exception radiant_mlhub.exceptions.**MLHubException**

Bases: Exception

Base exception class for all Radiant MLHub exceptions

6.4 radiant_mlhub.if_exists module

class radiant_mlhub.if_exists.**DownloadIfExistsOpts**(*value*)

Bases: str, enum.Enum

Allowed values for *download*'s *if_exists* option.

overwrite = 'overwrite'

resume = 'resume'

skip = 'skip'

6.5 radiant_mlhub.retry_config module

radiant_mlhub.retry_config.**config**() → Optional[urllib3.util.retry.Retry]

Common configuration for http backoff/retry strategy.

*{backoff factor} * (2 ** ({number of total retries} - 1))*

*0.2 * (2 ** (10 - 1)) = 102.4 seconds*

6.6 radiant_mlhub.session module

Methods and classes to simplify constructing and authenticating requests to the MLHub API.

It is generally recommended that you use the *get_session()* function to create sessions, since this will properly handle resolution of the API key from function arguments, environment variables, and profiles as described in *Authentication*. See the *get_session()* docs for usage examples.

class radiant_mlhub.session.**Session**(*, *api_key: Optional[str]*)

Bases: requests.sessions.Session

Custom class inheriting from requests.Session with some additional conveniences:

- Adds the API key as a `key` query parameter
- Adds an `Accept: application/json` header
- Adds a `User-Agent` header that contains the package name and version, plus basic system information like the OS name
- Prepends the MLHub root URL (`https://api.radiant.earth/mlhub/v1/`) to any request paths without a domain
- Raises a *radiant_mlhub.exceptions.AuthenticationError* for `401 (UNAUTHORIZED)` responses
- Calls requests.Response.raise_for_status() after all requests to raise exceptions for any status codes above 400.

API_KEY_ENV_VARIABLE = 'MLHUB_API_KEY'

DEFAULT_ROOT_URL = 'https://api.radiant.earth/mlhub/v1/'

MLHUB_HOME_ENV_VARIABLE = 'MLHUB_HOME'

PROFILE_ENV_VARIABLE = 'MLHUB_PROFILE'

ROOT_URL_ENV_VARIABLE = 'MLHUB_ROOT_URL'

classmethod from_config(*profile: Optional[str] = None*) → *radiant_mlhub.session.Session*

Create a session object by reading an API key from the given profile in the `profiles` file. By default, the client will look for the `profiles` file in a `.mlhub` directory in the user's home directory (as determined by `Path.home`). However, if an `MLHUB_HOME` environment variable is present, the client will look in that directory instead.

Parameters **profile** (*str, optional*) – The name of a profile configured in the `profiles` file.

Returns **session**

Return type *Session*

Raises ***APIKeyNotFound*** – If the given config file does not exist, the given profile cannot be found, or there is no `api_key` property in the given profile section.

classmethod from_env() → *radiant_mlhub.session.Session*

Create a session object from an API key from the environment variable.

Returns **session**

Return type *Session*

Raises ***APIKeyNotFound*** – If the API key cannot be found in the environment

paginate(*url: str*, ***kwargs: Any*) → Iterator[Dict[str, Any]]

Makes a GET request to the given `url` and paginates through all results by looking for a link in each response with a `rel` type of `"next"`. Any additional keyword arguments are passed directly to `requests.Session.get()`.

Parameters **url** (*str*) – The URL to which the initial request will be made. Note that this may either be a full URL or a path relative to the `ROOT_URL` as described in *`Session.request()`*.

Yields **page** (*dict*) – An individual response as a dictionary.

request(*method: str*, *url: str*, ***kwargs: Any*) → requests.models.Response

Overwrites the default `requests.Session.request()` method to prepend the MLHub root URL if the given `url` does not include a scheme. This will raise an *`AuthenticationError`* if a 401 response is returned by the server, and a `HTTPError` if any other status code of 400 or above is returned.

Parameters

- **method** (*str*) – The request method to use. Passed directly to the `method` argument of `requests.Session.request()`
- **url** (*str*) – Either a full URL or a path relative to the `ROOT_URL`. For example, to make a request to the Radiant MLHub API `/collections` endpoint, you could use `session.get('collections')`.
- ****kwargs** – All other keyword arguments are passed directly to `requests.Session.request()` (see that documentation for an explanation of these keyword arguments).

Raises

- ***AuthenticationError*** – If the response status code is 401
- **HTTPError** – For all other response status codes at or above 400

radiant_mlhub.session.**get_session**(*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.session.Session*

Gets a *Session* object that uses the given `api_key` for all requests. Resolves an API key by trying each of the following (in this order):

1. Use the *api_key* argument provided (Optional).
2. Use an *MLHUB_API_KEY* environment variable.
3. Use the profile argument provided (Optional).
4. Use the *MLHUB_PROFILE* environment variable.
5. Use the default profile

If none of the above strategies results in a valid API key, then an APIKeyNotFound exception is raised. See Using Profiles section for details.

Parameters

- **api_key** (*str, optional*) – The API key to use for all requests from the session. See description above for how the API key is resolved if not provided as an argument.
- **profile** (*str, optional*) – The name of a profile configured in the `.mlhub/profiles` file. This will be passed directly to *from_config()*.

Returns session

Return type *Session*

Raises ***APIKeyNotFound*** – If no API key can be resolved.

Examples

```
>>> from radiant_mlhub import get_session
# Get the API from the "default" profile
>>> session = get_session()
# Get the session from the "project1" profile
# Alternatively, you could set the MLHUB_PROFILE environment variable to "project1"
>>> session = get_session(profile='project1')
# Pass an API key directly to the session
# Alternatively, you could set the MLHUB_API_KEY environment variable to "some-api-
↪key"
>>> session = get_session(api_key='some-api-key')
```

6.7 Module contents

class radiant_mlhub.**Collection**(*id: str*, *description: str*, *extent: pystac.collection.Extent*, *title: Optional[str] = None*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, *catalog_type: Optional[pystac.catalog.CatalogType] = None*, *license: str = 'proprietary'*, *keywords: Optional[List[str]] = None*, *providers: Optional[List[pystac.provider.Provider]] = None*, *summaries: Optional[pystac.summaries.Summaries] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: pystac.collection.Collection

Class inheriting from pystac.Collection that adds some convenience methods for listing and fetching from the Radiant MLHub API.

property archive_size: Optional[int]

The size of the tarball archive for this collection in bytes (or None if the archive does not exist).

download(*output_dir: Union[str, pathlib.Path]*, ***, *if_exists:* radiant_mlhub.if_exists.DownloadIfExistsOpts = *DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pathlib.Path

Downloads the archive for this collection to an output location (current working directory by default). If the parent directories for output_path do not exist, they will be created.

The if_exists argument determines how to handle an existing archive file in the output directory. See the documentation for the download_archive() function for details. The default behavior is to resume downloading if the existing file is incomplete and skip the download if it is complete.

Note: Some collections may be very large and take a significant amount of time to download, depending on your connection speed.

Parameters

- **output_dir** (*Path*) – Path to a local directory to which the file will be downloaded. File name will be generated automatically based on the download URL.
- **if_exists** (*str, optional*) – How to handle an existing archive at the same location. If "skip", the download will be skipped. If "overwrite", the existing file will be overwritten and the entire file will be re-downloaded. If "resume" (the default), the existing file size will be compared to the size of the download (using the Content-Length header). If the existing file is smaller, then only the remaining portion will be downloaded. Otherwise, the download will be skipped.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **output_path** – The path to the downloaded archive file.

Return type pathlib.Path

Raises **FileExistsError** – If file at output_path already exists and both exist_okay and overwrite are False.

classmethod fetch(*collection_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Creates a *Collection* instance by fetching the collection with the given ID from the Radiant MLHub API.

Parameters

- **collection_id** (*str*) – The ID of the collection to fetch (e.g. `bigearthnet_v1_source`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns collection

Return type *Collection*

fetch_item(*item_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → pystac.item.Item

classmethod from_dict(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Collection*

Patches the `pystac.Collection.from_dict()` method so that it returns the calling class instead of always returning a `pystac.Collection` instance.

get_items(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → Iterator[pystac.item.Item]

Note: The `get_items` method is not implemented for Radiant MLHub *Collection* instances for performance reasons. Please use the *Dataset.download()* method to download Dataset assets.

Raises **NotImplementedError** –

classmethod list(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Collection*]

Returns a list of *Collection* instances for all collections hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns collections

Return type List[*Collection*]

property registry_url: Optional[str]

The URL of the registry page for this Collection. The URL is based on the DOI identifier for the collection. If the Collection does not have a `"sci:doi"` property then `registry_url` will be `None`.

class radiant_mlhub.**Dataset**(*id: str*, *collections: List[Dict[str, Any]]*, *title: Optional[str] = None*, *registry: Optional[str] = None*, *doi: Optional[str] = None*, *citation: Optional[str] = None*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, ***_: Any*)

Bases: object

Class that brings together multiple Radiant MLHub "collections" that are all considered part of a single "dataset". For instance, the bigearthnet_v1 dataset is composed of both a source imagery collection (bigearthnet_v1_source) and a labels collection (bigearthnet_v1_labels).

id

The dataset ID.

Type str

title

The title of the dataset (or None if dataset has no title).

Type str or None

registry_url

The URL to the registry page for this dataset, or None if no registry page exists.

Type str or None

doi

The DOI identifier for this dataset, or None if there is no DOI for this dataset.

Type str or None

citation

The citation information for this dataset, or None if there is no citation information.

Type str or None

property collections: radiant_mlhub.models.dataset._CollectionList

List of collections associated with this dataset. The list that is returned has 2 additional attributes (source_imagery and labels) that represent the list of collections corresponding the each type.

Note: This is a cached property, so updating self.collection_descriptions after calling self.collections the first time will have no effect on the results. See functools.cached_property() for details on clearing the cached value.

Examples

```
>>> from radiant_mlhub import Dataset
>>> dataset = Dataset.fetch('bigearthnet_v1')
>>> len(dataset.collections)
2
>>> len(dataset.collections.source_imagery)
1
>>> len(dataset.collections.labels)
1
```

To loop through all collections

```
>>> for collection in dataset.collections:
...     # Do something here
```

To loop through only the source imagery collections:

```
>>> for collection in dataset.collections.source_imagery:
...     # Do something here
```

To loop through only the label collections:

```
>>> for collection in dataset.collections.labels:
...     # Do something here
```

download(*output_dir: Union[pathlib.Path, str] = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/radiant-mlhub/checkouts/v0.5.4/docs/source')*, ***, *asset_output_dir: Optional[Union[pathlib.Path, str]] = None*, *catalog_only: bool = False*, *if_exists: radiant_mlhub.if_exists.DownloadIfExistsOpts = DownloadIfExistsOpts.resume*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*, *bbox: Optional[List[float]] = None*, *intersects: Optional[Dict[str, Any]] = None*, *datetime: Optional[Union[datetime.datetime, Tuple[datetime.datetime, datetime.datetime]]] = None*, *collection_filter: Optional[Dict[str, List[str]]] = None*) → None

Downloads dataset's STAC catalog and all linked assets. The download may be customized and controlled by providing bbox, intersects, datetime, and filter options.

Parameters

- **output_dir** (*str or pathlib.Path*) – The directory into which the STAC catalog will be written. If no asset_output_dir is specified, the assets will also be saved to the output_dir. Defaults to current working directory.
- **asset_output_dir** (*Otional[str, pathlib.Path]*) – The directory into which the archives will be written. If not defined by the user, the assets are saved to their respective asset level STAC catalog directories in the output_dir, which is in the current working directory by default.
- **catalog_only** (*bool*) – If True, the STAC catalog will be downloaded and unarchived, but no assets wll be downloaded. Defaults to False.
- **if_exists** (*Optional[str]*) – Allowed values: *skip*, *overwrite*, or *resume* (default).
- **bbox** (*Optional[List[float]]*) – List representing a bounding box of coordinates, for spatial intersection filter. Must be in CRS EPSG:4326.
- **intersects** (*Optional[GeoJSON]*) – GeoJSON object for spatial intersects filter. Must be a parsed GeoJSON dict with a *geometry* property.
- **datetime** (*Optional[datetime, Tuple[datetime, datetime]]*) – Single datetime or datetime range for temporal filter.
- **collection_filter** (*Optional[Dict[str, list]]*) – Mapping of collection_id and asset keys to include (exclusively).

  examples:

  - **download will only include this collection:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=[])*
  - **download will only include this collection and only these asset keys:** *dict(ref_landcovernet_sa_v1_source_sentinel_2=["B02", "B03", "B04"])*
- **api_key** (*Optional[str]*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable.
- **profile** (*Optional[str]*) – Authentication Profile to use when making this request.

Raises

- **IOError** – If `output_dir` exists and is not a directory. If unrecoverable download errors occurred.
- **ValueError** – If provided filters are incompatible, for example bbox and intersects.
- **RuntimeError** – If filters result in zero assets to download.

Any unrecoverable download errors will be logged to *{output_dir}/{dataset_id}/err_report.csv*.

property estimated_dataset_size: Optional[int]

Size in bytes of entire dataset (bytes)

classmethod fetch(*dataset_id_or_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by first trying to fetching the dataset based on ID, then falling back to fetching by DOI.

Parameters

- **dataset_id_or_doi** (*str*) – The ID or DOI of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_doi(*dataset_doi: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by fetching the dataset with the given DOI from the Radiant MLHub API.

Parameters

- **dataset_doi** (*str*) – The DOI of the dataset to fetch (e.g. `10.6084/m9.figshare.12047478.v2`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod fetch_by_id(*dataset_id: str*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *Dataset*

Creates a *Dataset* instance by fetching the dataset with the given ID from the Radiant MLHub API.

Parameters

- **dataset_id** (*str*) – The ID of the dataset to fetch (e.g. `bigearthnet_v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns dataset

Return type *Dataset*

classmethod list(*, *tags: Optional[Union[str, Iterable[str]]] = None*, *text: Optional[Union[str, Iterable[str]]] = None*, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*Dataset*]

Returns a list of `Dataset` instances for each datasets hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **tags** (A list of tags to filter datasets by. If not `None`, only datasets containing all) – provided tags will be returned.
- **text** (A list of text phrases to filter datasets by. If not `None`, only datasets) – containing all phrases will be returned.
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Yields **dataset** (*Dataset*)

property stac_catalog_size: Optional[int]

Size of the dataset_id.tar.gz STAC archive (bytes)

class radiant_mlhub.**DownloadIfExistsOpts**(*value*)

Bases: `str`, `enum.Enum`

Allowed values for *download*'s *if_exists* option.

overwrite = 'overwrite'

resume = 'resume'

skip = 'skip'

class radiant_mlhub.**MLModel**(*id: str*, *geometry: Optional[Dict[str, Any]]*, *bbox: Optional[List[float]]*, *datetime: Optional[datetime.datetime]*, *properties: Dict[str, Any]*, *stac_extensions: Optional[List[str]] = None*, *href: Optional[str] = None*, *collection: Optional[Union[str, pystac.collection.Collection]] = None*, *extra_fields: Optional[Dict[str, Any]] = None*, *, *api_key: Optional[str] = None*, *profile: Optional[str] = None*)

Bases: `pystac.item.Item`

classmethod fetch(*model_id: str*, *, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Fetches a `MLModel` instance by id.

Parameters

- **model_id** (*str*) – The ID of the ML Model to fetch (e.g. `model-cyclone-wind-estimation-torchgeo-v1`).
- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns **model**

Return type *MLModel*

classmethod from_dict(*d: Dict[str, Any]*, *href: Optional[str] = None*, *root: Optional[pystac.catalog.Catalog] = None*, *migrate: bool = False*, *preserve_dict: bool = True*, ***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.models.ml_model.MLModel*

Patches the `pystac.Item.from_dict()` method so that it returns the calling class instead of always returning a `pystac.Item` instance.

classmethod list(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → List[*radiant_mlhub.models.ml_model.MLModel*]

Returns a list of *MLModel* instances for all models hosted by MLHub.

See the *Authentication* documentation for details on how authentication is handled for this request.

Parameters

- **api_key** (*str*) – An API key to use for this request. This will override an API key set in a profile on using an environment variable
- **profile** (*str*) – A profile to use when making this request.

Returns models

Return type List[*MLModel*]

session_kwargs: Dict[str, Any] = {}

Class inheriting from `pystac.Item` that adds some convenience methods for listing and fetching from the Radiant MLHub API.

radiant_mlhub.**get_session**(***, *api_key: Optional[str] = None*, *profile: Optional[str] = None*) → *radiant_mlhub.session.Session*

Gets a `Session` object that uses the given `api_key` for all requests. Resolves an API key by trying each of the following (in this order):

1. Use the *api_key* argument provided (Optional).
2. Use an *MLHUB_API_KEY* environment variable.
3. Use the profile argument provided (Optional).
4. Use the *MLHUB_PROFILE* environment variable.
5. Use the default profile

If none of the above strategies results in a valid API key, then an APIKeyNotFound exception is raised. See Using Profiles section for details.

Parameters

- **api_key** (*str, optional*) – The API key to use for all requests from the session. See description above for how the API key is resolved if not provided as an argument.
- **profile** (*str, optional*) – The name of a profile configured in the `.mlhub/profiles` file. This will be passed directly to `from_config()`.

Returns session

Return type *Session*

Raises ***APIKeyNotFound*** – If no API key can be resolved.

Examples

```
>>> from radiant_mlhub import get_session
# Get the API from the "default" profile
>>> session = get_session()
# Get the session from the "project1" profile
# Alternatively, you could set the MLHUB_PROFILE environment variable to "project1"
>>> session = get_session(profile='project1')
# Pass an API key directly to the session
# Alternatively, you could set the MLHUB_API_KEY environment variable to "some-api-
↪key"
>>> session = get_session(api_key='some-api-key')
```

CHAPTER

SEVEN

CLI TOOLS

7.1 mlhub

CLI tool for the radiant_mlhub Python client.

```
mlhub [OPTIONS] COMMAND [ARGS]...
```

Options

--version

Show the version and exit.

7.1.1 configure

Interactively set up radiant_mlhub configuration file.

This tool walks you through setting up a ~/.mlhub/profiles file and adding an API key. If you do not provide a –profile option, it will update the “default” profile. If you do not provide an –api-key option, you will be prompted to enter an API key by the tool.

If you need to change the location of the profiles file, set the MLHUB_HOME environment variable before running this command.

For details on profiles and authentication for the radiant_mlhub client, please see the official Authentication documentation:

https://radiant-mlhub.readthedocs.io

```
mlhub configure [OPTIONS]
```

Options

--profile <profile>

The name of the profile to configure.

--api-key <api_key>

The API key to use for this profile.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

Symbols

A

B

C

D

E

F

G

I

L

M

O

P

R

S

T

U

W