xyzspaces

HERE Europe B.V.

Sep 18, 2020

CONTENTS:

At the moment this consists only of this API reference which is automatically generated from the source code. Apart from this there are Jupyter notebooks to get started with some examples. A more prosaic documentation will hopefully follow, soon.

CHAPTER

ONE

XYZSPACES PACKAGE

The XYZ Spaces for Python - manage your XYZ Hub server or HERE Data Hub.

XYZ Spaces for Python allows you to manage XYZ spaces, projects and tokens with the Hub API, Project API, and Token API, respectively. The Hub API provides the most features to let you read and write GeoJSON data (features) from and to an XYZ space, and perform some higher-level operations like return features inside or clipped by some bounding box or map tile. The Project API and Token API let you manage your XYZ projects and tokens.

See also: - XYZ Hub server: https://github.com/heremaps/xyz-hub - HERE Data Hub: https://developer.here.com/products/data-hub

class xyzspaces.**XYZ**(*credentials=None*)

Bases: object

A single interface to interact with your XYZ Hub server or HERE Data Hub.

__init__(*credentials=None*)

Instantiate an XYZ object, optionally with access credentials.

Parameters **credentials** (*Optional[str]*) –

1.1 Subpackages

1.1.1 xyzspaces.datasets package

This package provides access to some public datasets used.

xyzspaces.datasets.**get_countries_data**()

Pull countries example GeoJSON from the net or a locally cached file.

If this is not locally cached, yet, it will be after the first call, unless the file cannot be saved, in which case it will be re-downloaded again with every call.

Source (under http://unlicense.org): https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json

The data contains 180 countries, and does not cover all existing countries, ca. 200. For example the Vatican is missing.

Returns A JSON object.

xyzspaces.datasets.**get_chicago_parks_data**()

Create GeoJSON from file chicago_parks.geo.json stored locally.

xyzspaces.datasets.**get_microsoft_buildings_space**()

Create a space object for the MS “US Buildings Footprints” dataset.

The original source for this dataset can be found on https://github.com/Microsoft/USBuildingFootprints.

Returns A space object.

1.2 Submodules

1.2.1 xyzspaces.__version__ module

Project version information.

1.2.2 xyzspaces.apis module

This module for accessing the APIs of your XYZ Hub server or HERE Data Hub.

It provides classes like *HubApi*, *ProjectApi*, and *TokenApi* to interact with the respective XYZ APIs in a programmatic way.

class xyzspaces.apis.**Api**(*server*, *credentials=None*)

Bases: object

A low-level HTTP RESTful API client.

This uses requests to make HTTP requests with typical parameters and will return the entire response when calling instances directly, or when using the aliased HTTP methods like *Api.get()*, *Api.put()* etc. provided for convenience.

All these methods like *Api.get()*, *Api.put()* etc. will raise ApiError if the status code of the HTTP response is not in the interval [200, 300).

This class is a base class for concrete HERE XYZ APIs, but can also be used outside of that particular context for any RESTful API, maybe with slight changes regarding authentication.

__init__(*server*, *credentials=None*)

Instantiate an *Api* object.

Parameters

- **server** (*Optional[str]*) – The URL of the server implementing the API.
- **credentials** (*Optional[Any]*) – Any object to serve as authentication (not used in this class).

__call__(*method*, *path=''*, *params=None*, *headers=None*, *cookies=None*, *json=None*, *data=None*, *proxies=None*)

Make an API call with parameters passed to requests.

Parameters

- **method** (*str*) – The HTTP method name, e.g. “GET”, “PUT”, etc.
- **path** (*Optional[str]*) – The HTTP path to be appended to the server attribute.
- **params** (*Optional[Dict]*) – A dict holding the HTTP query parameters.
- **headers** (*Optional[Dict]*) – A dict holding the HTTP request headers.
- **cookies** (*Optional[Dict]*) – A dict holding the HTTP request cookies.
- **json** (*Optional[Dict]*) – A JSON object (usually a dict) to be passed as request body with content-type application/json.

- **data** (*Optional[Dict]*) – A str to be passed as request body with content-type `application/x-www-form-urlencoded`.
- **proxies** (*Optional[Dict]*) – A dict holding the HTTP proxies to be used.

Returns The HTTP response returned by the `requests` package.

Raises ***ApiError*** – If the status code of the HTTP response is not in the interval [200, 300).

Return type requests.models.Response

get(***kwargs*)
Send a HTTP GET request.

Parameters **kwargs** – Keyword arguments passed when sending the HTTP request.

Returns The HTTP response.

Return type requests.models.Response

put(***kwargs*)
Send a HTTP PUT request.

Parameters **kwargs** – Keyword arguments passed when sending the HTTP request.

Returns The HTTP response.

Return type requests.models.Response

patch(***kwargs*)
Send a HTTP PATCH request.

Parameters **kwargs** – Keyword arguments passed when sending the HTTP request.

Returns The HTTP response.

Return type requests.models.Response

post(***kwargs*)
Send a HTTP POST request.

Parameters **kwargs** – Keyword arguments passed when sending the HTTP request.

Returns The HTTP response.

Return type requests.models.Response

delete(***kwargs*)
Send a HTTP DELETE request.

Parameters **kwargs** – Keyword arguments passed when sending the HTTP request.

Returns The HTTP response.

Return type requests.models.Response

class xyzspaces.apis.**ProjectApi**(*server='https://xyz.api.here.com'*, *credentials=None*)
Bases: *xyzspaces.apis.Api*

XYZ RESTful Project API abstraction.

Instances of this class allow to manage XYZ Hub projects.

API calls must be authenticated via a bearer token which needs to be provided as a `credentials` parameter when initialising an instance.

__init__(*server='https://xyz.api.here.com'*, *credentials=None*)
Instantiate a *ProjectApi* object.

Parameters

- **server** (*Optional[str]*) – The URL of the server implementing the API.
- **credentials** (*Optional[str]*) – A string to serve as authentication (a bearer token). Will be looked up in environment variable `XYZ_TOKEN` if not provided.

get_projects (*paginate=None*, *handle=None*, *limit=None*)
List the projects either for the token owner or list the published projects.

Parameters

- **paginate** (*Optional[bool]*) – A Boolean telling if the results should be paginated (default: ?).
- **handle** (*Optional[int]*) – A string to be used when running the next request in a sequence of pagninated responses. Works only when `paginate` is `True`.
- **limit** (*Optional[int]*) – The max. number of projects to return. Works only when `paginate` is `True`.

Returns A JSON object containing information about the projects.

Return type dict

get_project (*project_id*)
Get the project by ID.

Parameters **project_id** (*str*) – A string representing the project ID.

Returns A JSON object with information about the requested project.

Return type dict

post_project (*data*)
Create a project.

Parameters **data** – A JSON object describing the project to be created.

Returns A JSON object with all information about the created project.

Return type dict

put_project (*project_id*, *data*)
Update a project by ID.

Update the project with the provided project ID and create it if it does not exist. This will replace the whole project definition.

Parameters

- **project_id** (*str*) – A string representing the desired project ID.
- **data** – A JSON object describing the project details to be updated.

Returns A JSON object with information about the updated project.

Return type dict

patch_project (*project_id*, *data*)
Update parts of a project by ID.

Parameters

- **project_id** (*str*) – A string representing the desired project ID.
- **data** – A JSON object describing the project parts to be updated.

Returns A JSON object with information about the updated project.

Return type dict

delete_project(*project_id*)
Delete a project by ID.

Parameters **project_id** (*str*) – A string representing the desired project ID.

Returns An empty string if the operation was successful.

Return type str

class xyzspaces.apis.**TokenApi**(*server='https://xyz.api.here.com'*, *credentials=None*)
Bases: *xyzspaces.apis.Api*

XYZ RESTful Token API abstraction.

Instances of this API class allow to manage HERE XYZ tokens.

API calls must be authenticated via authenticated access cookies derived from the username and password of an existing HERE developer account with access to HERE XYZ. These need to be provided as a `credentials` parameter when initialising an instance.

Example:

```
>>> from xyzspaces.apis import TokenApi
>>> api = TokenApi(credentials=dict(username="foo", password="bar"))
>>> api.get_tokens()
[...]
```

__init__(*server='https://xyz.api.here.com'*, *credentials=None*)
Instantiate a *TokenApi* object.

Parameters

- **server** (*Optional[str]*) – The URL of the server implementing the API.
- **credentials** (*Optional[Dict[str, str]]*) – A dict to hold the authentication details (the fields `username` and `password` with valid values for the user's existing HERE developer account). Will be looked up in environment variables `HERE_USER` and `HERE_PASSWORD` if not provided.

get_token(*token_id*, ***kwargs*)
Get info for given token ID.

Parameters

- **token_id** (*str*) – A string representing the requested token.
- **kwargs** – A dict with additional parameters passed to the HTTP request made when provided.

Returns A JSON object with the information about the requested token.

Return type dict

get_tokens(***kwargs*)
Get list of tokens for the authenticated user.

Parameters **kwargs** – A dict with additional parameters passed to the HTTP request made when provided.

Returns A list with information about all available tokens.

Return type list

post_token (*json={}*, ***kwargs*)
Create a new permanent or temporary token.

Parameters

- **json** (*Dict*) – A dict with information about the token to be created.
- **kwargs** – A dict with additional parameters passed to the HTTP request made when provided.

Returns A JSON object with all information about the created token.

Return type dict

delete_token (*token_id*, ***kwargs*)
Delete the token with the provided ID.

Parameters

- **token_id** (*str*) – A string representing a valid token.
- **kwargs** – A dict with additional parameters passed to the HTTP request made when provided.

Returns An empty string if the operation was successful.

Return type str

class xyzspaces.apis.**HubApi** (*server='https://xyz.api.here.com'*, *credentials=None*)
Bases: *xyzspaces.apis.Api*

XYZ RESTful Hub API abstraction.

Instances of this API class allow to manage HERE XYZ spaces.

API calls must be authenticated via a bearer token which needs to be provided as a `credentials` parameter when initialising an instance.

A few convenience methods for calling HTTP methods directly are inherited from the *Api* base class, and can also be used, although this class provides dedicated methods for the Hub API, like *HubApi.get_spaces()* starting with HTTP method names and an underscore, followed by a name resembling the respective API endpoint, in this case `GET /hub/spaces`.

Example:

```
>>> from xyzspaces.apis import HubApi
>>> api = HubApi(credentials="MY_XYZ_TOKEN")
>>> api.get_spaces()
[...]
```

This is based on the HERE XYZ Hub API specification defined here: https://xyz.api.here.com/hub/static/swagger/#/

__init__ (*server='https://xyz.api.here.com'*, *credentials=None*)
Instantiate an *HubApi* object.

Parameters

- **server** (*Optional[str]*) – The URL of the server implementing the API.
- **credentials** (*Optional[str]*) – A string to serve as authentication (a bearer token). Will be looked up in environment variable `XYZ_TOKEN` if not provided.

get_hub (*params=None*)
Get basic information about the XYZ Hub.

Parameters **params** (*dict*) – A dict holding the HTTP query parameters.

Returns A JSON object with hub information.

Return type dict

get_spaces(*params=None*)
Get Spaces information.

Parameters **params** (*dict*) – A dict holding the HTTP query parameters.

Returns A JSON object with list of spaces.

Return type dict

get_space(*space_id*, *params=None*)
Get a space by ID.

Parameters

- **space_id** (*str*) – The desired space ID.
- **params** (*dict*) – A dict for query params.

Returns A JSON object with information about the space_id.

Return type dict

post_space(*data*)
Create a space.

Parameters **data** (*dict*) – A dict describing the space to be created.

Returns A JSON object with all information about the created space.

Return type dict

patch_space(*space_id*, *data*)
Update a space.

Parameters

- **space_id** (*str*) – A string representing the desired space ID.
- **data** (*dict*) – A JSON object describing the space attributes to be updated.

Returns A JSON object with information about the updated space.

Return type dict

delete_space(*space_id*)
Delete a space.

Parameters **space_id** (*str*) – A string representing desired space ID.

Returns An empty string if the operation was successful.

Return type str

get_space_features(*space_id*, *feature_ids*)
Get features by ID.

Parameters

- **space_id** (*str*) – The desired space ID.
- **feature_ids** (*List[str]*) – A list of feature_ids.

Returns A feature collection with all features inside the specified space.

Return type dict

Example: Get single feature from space

```
>>> feats = api.get_space_features(
...          space_id=space_id, feature_ids=["GER", "BRA"])
>>> print(feats)
```

get_space_feature(*space_id*, *feature_id*)
Get a feature by ID.

Parameters

- **space_id** (*str*) – The desired space ID.
- **feature_id** (*str*) – The desired feature ID.

Returns A feature with the specified feature ID inside the space with the specified ID.

Return type dict

Example: Read the feature from the space.

```
>>>     feature = api.get_space_feature(
...     space_id=space_id, feature_id=feature_id)
>>>     print(json.dumps(feature, indent=4, sort_keys=True))
```

get_space_statistics(*space_id*)
Get statistics.

Parameters **space_id** (*str*) – The desired space ID.

Returns A JSON object with some statistics about the specified space.

Return type dict

Example:

```
>>>     stats = api.get_space_statistics(space_id=space_id)
>>>     print(json.dumps(stats, indent=4, sort_keys=True))
```

get_space_bbox(*space_id*, *bbox*, *tags=None*, *clip=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*, *clustering=None*, *clusteringParams=None*)
Get features inside some given bounding box.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **bbox** (*List[Union[int, float]]*) – A list of four numbers representing the West, South, East and North margins, respectively, of the bounding box.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **clip** (*Optional[bool]*) – A Boolean indicating if the result should be clipped (default: False).
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – ...
- **selection** (*Optional[List[str]]*) – ...
- **skip_cache** (*Optional[bool]*) – ...
- **clustering** (*Optional[str]*) – ...

- **clusteringParams** (*Optional[dict]*) – ...

Returns A dict representing a feature collection.

Return type dict

Example:

```
>>>     bb = [0, 0, 20, 20]
>>>     bbox = api.get_space_bbox(space_id=space_id, bbox=bb)
>>>     print(len(bbox["features"]))
>>>     print(bbox["type"])
```

get_space_tile(*space_id*, *tile_type*, *tile_id*, *tags=None*, *clip=None*, *params=None*, *selection=None*, *skip_cache=None*, *clustering=None*, *clusteringParams=None*, *margin=None*, *limit=None*)

Get features in tile.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **tile_type** (*str*) – A string with the name of a tile type, one of "quadkeys", "web", "tms" or "here". See below.
- **tile_id** (*str*) – A string holding a valid tile ID according to the specified `tile_type`.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **clip** (*Optional[bool]*) – A Boolean indicating if the result should be clipped (default: False).
- **margin** (*Optional[int]*) – ...
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – ...
- **selection** (*Optional[List[str]]*) – ...
- **skip_cache** (*Optional[bool]*) – ...
- **clustering** (*Optional[str]*) – ...
- **clusteringParams** (*Optional[dict]*) – ...

Returns A dict representing a feature collection.

Return type dict

Available tile types are:

- quadkeys, Virtual Earth Tile System,
- web, Tiled Web Map.
- tms, OSGEO Tile Map Service,
- here, ?

get_space_search(*space_id*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*)

Search for features.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.

- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – ...
- **selection** (*Optional[List[str]]*) – ...
- **skip_cache** (*Optional[bool]*) – ...

Returns A dict representing a feature collection.

Return type dict

Example:

```
>>>     feats = api.get_space_search(space_id=space_id)
>>>     print(feats["type"] )
>>>     print(len(feats["features"]) )
```

get_space_iterate(*space_id*, *limit*)

Iterate features in the space (yielding them one by one).

Parameters

- **space_id** (*str*) – A string representing desired space ID.
- **limit** (*int*) – A max. number of features to return in the result.

Yields A feature in space.

Return type Generator

get_space_all(*space_id*, *limit*, *max_len=1000*)

Get all features as one single GeoJSON feature collection.

This is a convenience method, not directly available in the XYZ API. It hides the API paging mechanism and returns all data in one chunk. So be careful if you don't know how much data you will get.

Parameters

- **space_id** (*str*) – A string representing desired space ID.
- **limit** (*int*) – A max. number of features to return in the result.
- **max_len** – A max. number of features to return in the result.

Returns A dict representing a feature collection.

Return type dict

Example:

```
>>>     fc = api.get_space_all(space_id=space_id, limit=100)
>>>     print(len(fc["features"]) )
>>>     print(fc["type"])
```

get_space_count(*space_id*)

Get feature count.

Parameters **space_id** (*str*) – A string with the ID of the desired XYZ space.

Returns A dict containing the number of features inside the specified space.

Return type dict

put_space_features(*space_id*, *data*, *add_tags=None*, *remove_tags=None*)
Create or replace multiple features.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **data** (*dict*) – A JSON object describing one or more features to add.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the features.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the features.

Returns A dict representing a feature collection.

Return type dict

Example:

```
>>> from xyzspaces.datasets import get_countries_data
>>>     gj_countries = get_countries_data()
>>>     features = api.put_space_features(
...     space_id=space_id,
...     data=gj_countries,
...     add_tags=["foo", "bar"],
...     remove_tags=["bar"],
... )
>>> print(features)
```

post_space_features(*space_id*, *data*, *add_tags=None*, *remove_tags=None*)
Modify multiple features in the space.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **data** (*dict*) – A JSON object describing one or more features to modify.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the features.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the features.

Returns A dict representing a feature collection.

Return type dict

Example:

```
>>> data = dict(type="FeatureCollection", features=[deu, ita])
>>> space_features = api.post_space_features(
...     space_id=space_id, data=data)
>>> print(space_features)
```

delete_space_features(*space_id*, *id=None*, *tags=None*)
Delete multiple features from the space.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **id** (*Optional[List[str]]*) – A list of feature IDs to delete.

- **tags** (*Optional[List[str]]*) – A list of strings describing tags the features to be deleted must have.

Returns A response from API call.

Return type str

Example:

```
>>>     deu = api.get_space_feature(space_id=space_id, feature_id="DEU")
>>>     ita = api.get_space_feature(space_id=space_id, feature_id="ITA")
>>>     deleted_features = api.delete_space_features(
...     space_id=space_id, id=["DEU", "ITA"])  # noqa: E501
```

put_space_feature(*space_id*, *data*, *feature_id=None*, *add_tags=None*, *remove_tags=None*)

Create or replace a single feature.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **data** (*dict*) – A JSON object describing the feature to be added.
- **feature_id** (*Optional[str]*) – A string with the ID of the feature to be created.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the feature.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the feature.

Returns A dict representing a feature.

Return type dict

Example:

```
>>>     api.put_space_feature(
...     space_id=space_id, feature_id=feature_id, data=fra)
```

patch_space_feature(*space_id*, *feature_id*, *data*, *add_tags=None*, *remove_tags=None*)

Patch a single feature in the space.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **feature_id** (*str*) – A string with the ID of the feature to be modified.
- **data** (*dict*) – A JSON object describing the feature to be changed.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the feature.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the feature.

Returns A dict representing a feature.

Return type dict

delete_space_feature(*space_id*, *feature_id*)

Delete a single feature from the space.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **feature_id** (*str*) – A string with the ID of the feature to be deleted.

Returns An empty string if the operation was successful.

Return type str

get_space_spatial(*space_id*, *lat=None*, *lon=None*, *ref_space_id=None*, *ref_feature_id=None*, *radius=None*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*)

Get features with radius search.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **lat** (*Optional[float]*) – A float in WGS'84 decimal degree (-90 to +90) of the center Point.
- **lon** (*Optional[float]*) – A float in WGS'84 decimal degree (-180 to +180) of the center Point.
- **ref_space_id** (*Optional[str]*) – A string as alternative for defining center coordinates, it is possible to reference a geometry in a space, hence it is needed to provide the `ref_space_id` where the referenced feature is stored. Always to use in combination with `ref_feature_id`.
- **ref_feature_id** (*Optional[str]*) – A string as unique identifier of a feature in the referenced space. The geometry of that feature gets used for the spatial query. Always to use in combination with `ref_space_id`.
- **radius** (*Optional[int]*) – An int in meter which defines the diameter of the search request.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – A dict holding the HTTP query parameters.
- **selection** (*Optional[List[str]]*) – A list of strings holding properties values.
- **skip_cache** (*Optional[bool]*) – A Boolean if set to `True` the response is not returned from cache.

Returns A dict representing a feature collection.

Raises **ValueError** – If incorrect params are passed, either `lat` and `lon` or `ref_space_id` and `ref_feature_id` must have a value.

Return type dict

post_space_spatial(*space_id*, *data*, *radius=None*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*)

Post features which intersect the provided geometry.

Parameters

- **space_id** (*str*) – A string with the ID of the desired XYZ space.
- **data** (*dict*) – A JSON object which is getting used for the spatial search.
- **radius** (*Optional[int]*) – An int which defines the diameter(meters) of the search request.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.

- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – A dict holding the HTTP query parameters.
- **selection** (*Optional[List[str]]*) – A list of strings holding properties values.
- **skip_cache** (*Optional[bool]*) – A Boolean if set to `True` the response is not returned from cache.

Returns A dict representing a feature collection.

Return type dict

1.2.3 xyzspaces.auth module

This module provides HERE authentication via cookies for the Token API.

This *get_auth_cookies()* function simulates the login process on http://developer.here.com to obtain an access cookie for authenticating with certain RESTful APIs like the XYZ Token API.

This implementation is inspired by the Open Source HERE XYZ CLI: https://github.com/heremaps/here-cli/blob/master/src/sso.ts.

`xyzspaces.auth.`**filter_cookies**(*cookies*, *prefix*)

Filter `requests` cookies with some given name prefix into a new dict.

Parameters

- **cookies** (*requests.cookies.RequestsCookieJar*) – A `requests` cookies object.
- **prefix** (*str*) – A prefix string to search in cookies.

Returns A dict.

Return type dict

Example:

Input:

```
<RequestsCookieJar[
    <Cookie locale=en-US for .here.com/>,
    <Cookie here_account=foobar for account.here.com/>,
    <Cookie here_account.sig=barfoo for account.here.com/>]
>
```

Output:

```
{'here_account': 'foobar', 'here_account.sig': 'barfoo'}
```

`xyzspaces.auth.`**get_auth_cookies**(*username*, *password*)

Get authentication cookies from name and password of a HERE account.

Parameters

- **username** (*str*) – Username for HERE account.
- **password** (*str*) – Password for HERE account.

Returns A dict.

Raises ***AuthenticationError*** – If status_code for HTTP response returned by `requests` is not equal to 200.

Return type dict

1.2.4 xyzspaces.curl module

This module contains functionality related to `curl` commands.

It provides methods for creating/executing `curl` commands which mimic the `requests` signatures.

This module has been mainly designed to be used in the `Api` module for logging purposes.

The `curl` module could be also used as stand alone:

Example:

```
>>> import xyzspaces.curl as curl
>>> command = curl.get(url='https://xkcd.com/552/info.0.json')
['curl', '--request', 'GET', 'https://xkcd.com/552/info.0.json']
>>> curl.execute(command)
b'{"month": "3", "num": 552, "link": "", "year": "2009", "news": "", "safe_title":
↪"Correlation", "transcript": "[[A man is talking to a woman]]\\nMan: I used to␣
↪think correlation implied causation.\\nMan: Then I took a statistics class.  Now I␣
↪don\'t.\\nWoman: Sounds like the class helped.\\nMan: Well, maybe.\\n{{Title text:␣
↪Correlation doesn\'t imply causation, but it does waggle its eyebrows suggestively␣
↪and gesture furtively while mouthing \'look over there\'.}}", "alt": "Correlation␣
↪doesn\'t imply causation, but it does waggle its eyebrows suggestively and gesture␣
↪furtively while mouthing \'look over there\'.", "img": "https://imgs.xkcd.com/
↪comics/correlation.png", "title": "Correlation", "day": "6"}'  # noqa: E501
[...]
```

`xyzspaces.curl.`**`get`**(*url*, *params=None*, ***kwargs*)

Run `curl GET` mimicking the `requests.get()` signature.

This completes a `curl GET` command.

Parameters

- **`url`** – The URL of the server including the path.
- **`params`** – The query string params.
- **`kwargs`** – Further keyword arguments that should match those expected by `requests`.

Returns The `List[str]` representation of the `curl GET` command.

Return type List[str]

Example:

```
>>> import xyzspaces.curl as curl
>>> curl.get(url="https://xkcd.com/552/info.0.json")
['curl', '--request', 'GET', 'ttps://xkcd.com/552/info.0.json']
```

`xyzspaces.curl.`**`put`**(*url*, *data=None*, ***kwargs*)

Run `curl PUT` command mimicking the `requests.put()` signature.

This completes a `curl PUT` command.

Parameters

- **`url`** – The URL of the server including the path for the new object.
- **`data`** – (optional) Dictionary, list of tuples, bytes, or file-like object.

- **kwargs** – Further keyword arguments that should match those expected by `requests`.

Returns The `List[str]` representation of the `curl PUT` command.

Return type List[str]

`xyzspaces.curl.`**patch**(*url*, *data=None*, ***kwargs*)

Run `curl PATCH` command mimicking the `requests.patch()` signature.

This completes a `curl PATCH` command.

Parameters

- **url** – The URL of the server including the path for the new object.
- **data** – (optional) Dictionary, list of tuples, bytes, or file-like object.
- **kwargs** – Further keyword arguments that should match those expected by `requests`.

Returns The `List[str]` representation of the `curl PATCH` command.

Return type List[str]

`xyzspaces.curl.`**post**(*url*, *data=None*, *json=None*, ***kwargs*)

Run `curl POST` command mimicking the `requests.post()` signature.

This completes a `curl POST` command.

Parameters

- **url** – The URL of the server including the path for the new object.
- **data** – (optional) Dictionary, list of tuples, bytes, or file-like object.
- **json** – (optional) json data.
- **kwargs** – Further keyword arguments that should match those expected by `requests`.

Returns The `List[str]` representation of the `curl POST` command.

Return type List[str]

`xyzspaces.curl.`**delete**(*url*, ***kwargs*)

Run `curl DELETE` command mimicking the `requests.delete()` signature.

This completes a `curl DELETE` command.

Parameters

- **url** – The URL of the server including the path.
- **kwargs** – Further keyword arguments that should match those expected by `requests`.

Returns The `List[str]` representation of the `curl DELETE` command.

Return type List[str]

`xyzspaces.curl.`**command**(*url*, *method*, *params=None*, ***kwargs*)

Return a `curl` command equivalent from the params for `requests`.

This builds and returns a list of strings representing a `curl` command that can be directly passed to functions like `subprocess.check_output()`. When joined with blanks into a single string it can also be used for logging or pasting to a terminal.

To be used like `requests.get()`, passing the same params for headers, cookies, etc. So the function signature is similar to `requests.get()`, with additional `method` parameter.

Parameters

- **url** (*str*) – The URL of the server including the path.
- **method** (*str*) – The HTTP method name, e.g. “GET”, “PUT”, etc.
- **params** (*Optional[dict]*) – The query string params.
- **kwargs** (*Optional[Mapping]*) – Further keyword arguments that should match those expected by `requests`.

Returns The generated `curl` command (a list of strings).

Return type List[str]

Example:

```
>>> import xyzspaces.curl as curl
>>> curl.command(method="get", url="https://xkcd.com/552/info.0.json")
['curl', '--request', 'GET', 'https://xkcd.com/552/info.0.json']
```

`xyzspaces.curl.`**execute**(*command*)

Execute a `command` and create the `requests.models.Response` object.

The Python’s `subprocess` module will be used for the executing a `command`.

In the `requests.models.Response` object will be initialized following attributes: _content: The response data from the `stdout` status_code: Simplified conversion from the `subprocess.CompletedProcess.returncode` to `HTTP` ones 200 ~ 0, 500 ~ >0.

Parameters **command** (*List[str]*) – The `curl` to be executed in the `List[str]` type.

Returns The generated `requests.models.Response` object.

Return type requests.models.Response

Example:

```
>>> import xyzspaces.curl as curl
>>> command = curl.get(url='https://xkcd.com/552/info.0.json')
['curl', '--request', 'GET', 'ttps://xkcd.com/552/info.0.json']
>>> curl.execute(command)
b'{"month": "3", "num": 552, "link": "", "year": "2009", "news": "", "safe_title
↪": "Correlation", "transcript": "[[A man is talking to a woman]]\\nMan: I used␣
↪to think correlation implied causation.\\nMan: Then I took a statistics class. ␣
↪Now I don\'t.\\nWoman: Sounds like the class helped.\\nMan: Well, maybe.\\n{
↪{Title text: Correlation doesn\'t imply causation, but it does waggle its␣
↪eyebrows suggestively and gesture furtively while mouthing \'look over there\'.}
↪}", "alt": "Correlation doesn\'t imply causation, but it does waggle its␣
↪eyebrows suggestively and gesture furtively while mouthing \'look over there\'.
↪", "img": "https://imgs.xkcd.com/comics/correlation.png", "title": "Correlation
↪", "day": "6"}'  # noqa: E501
[...]
```

1.2.5 xyzspaces.exceptions module

This module defines API exceptions.

exception xyzspaces.exceptions.**AuthenticationError**
Bases: Exception

Exception raised when authentication fails.

exception xyzspaces.exceptions.**ApiError**
Bases: Exception

Exception raised for API HTTP response status codes not in [200...300).

The exception value will be the response object returned by requests which provides access to all its attributes, eg. status_code, reason and text, etc.

Example:

```
>>> try:
>>>     api = HubApi(credentials="MY-XYZ-TOKEN")
>>>     api.get("/hub/nope").json()
>>> except ApiError as e:
>>>     resp = e.value.args[0]
>>>     if resp.status_code == 404 and resp.reason == "Not Found":
>>>         ...
```

__str__()
Return a string from the HTTP response causing the exception.

The string simply lists the repsonse's status code, reason and text content, separated with commas.

1.2.6 xyzspaces.logconf module

This module configures logging.

xyzspaces.logconf.**setup_logging**(*default_path='config/logconfig.json'*, *default_level=40*, *env_key='XYZ_LOG_CONFIG'*)
Set up logging configuration.

Parameters

- **default_path** (*str*) – A string representing the path of the config file in JSON format.
- **default_level** (*int*) – An int representing logging level.
- **env_key** (*str*) – A string representing environment variable to enable logging to file.

class xyzspaces.logconf.**NullHandler**(*level=0*)
Bases: logging.Handler

NullHandler class is a 'no-op' handler for use by library developers.

emit(*record*)
Skip the emit record. This is used to give preference to user.

1.2.7 xyzspaces.spaces module

An more Pythonic abstraction for XYZ Spaces.

This contains only one class for an XYZ "space" which in turn provides access to "features". There is no separate class abstraction for single features, but they are taken to be valid `geojson.GeoJSON` objects. Various other bits of the XYZ Hub API are simply returned as-is, usually wrapped in dictionaries, like the "statistics" of some given XYZ space.

class xyzspaces.spaces.**Space**(*api=None*)

Bases: `object`

An abstraction for XYZ Spaces.

A space object is created with an existing, authenticated XYZ Hub API instance.

Example:

```
>>> space = Space.new(...)
>>> space.delete()
>>> for feat in space.iter_feature():
....   print(feat["id"])
```

classmethod from_id(*space_id*)

Instantiate a space object for an existing space ID.

Parameters **space_id** (*str*) –

Return type *xyzspaces.spaces.Space*

classmethod new(*title*, *description=None*, *space_id=None*, *schema=None*, *enable_uuid=None*, *listeners=None*, *shared=None*)

Create new space object with given title and description.

Optionally, the desired space ID can be provided as well, and will be used if still available.

Parameters

- **title** (*str*) – A string representing the title of the space.
- **description** (*Optional[str]*) – A string representing a description of the space.
- **space_id** (*Optional[str]*) – A string representing space_id.
- **schema** (*str*) – JSON object or URL to be added as a schema for space.
- **enable_uuid** (*Optional[bool]*) – A boolean if set `True` it will create additional activity log space to log the activities in current space.
- **listeners** (*Optional[Dict[str, Union[int, str]]]*) – A dict for activity log listener params.
- **shared** (*Optional[bool]*) – A boolean, if set to `True`, space will be shared with other users having XYZ account, they will be able to read from the space using their own token. By default space will not be a shared space.

Returns A object of *Space*.

Return type *xyzspaces.spaces.Space*

classmethod virtual(*title*, *description=None*, ***kwargs*)

Create a new virtual-space.

Virtual-space references one or multiple spaces. A virtual-space is described by definition which references other existing spaces (the upstream spaces). Queries being done to a virtual-space will return the features

of its upstream spaces combined. There are different predefined operations of how to combine the features of the upstream spaces. In order to use virtual spaces feature user needs to have HERE Data Hub paid plan. Plans can be found here: HERE Data Hub.

Parameters

- **title** (*str*) – A string representing the title of the virtual-space.
- **description** (*Optional[str]*) – A string representing a description of the virtual-space.
- **kwargs** (*Dict[str, Dict]*) – A dict for the operation to perform on upstream spaces.

Returns An object of *Space*.

Return type *xyzspaces.spaces.Space*

__init__(*api=None*)
Instantiate a space object, optionally with authenticated api instance.

Parameters **api** (*Optional[xyzspaces.apis.HubApi]*) –

__repr__()
Return string representation of this instance.

property info
Space config information.

list(*owner='me'*, *include_rights=False*)
Return list of spaces for given owner with access rights if desired.

This does not modify the space object itself.

Parameters

- **owner** (*str*) – A string representing the owner.
- **include_rights** (*bool*) – A boolean. If set to True, the access rights for each space are included in the response.

Returns A JSON object with list of spaces.

Return type Dict

read(*id*)
Read existing space object for given space ID.

Parameters **id** (*str*) –

Return type *xyzspaces.spaces.Space*

update(*title=None*, *description=None*, *tagging_rules=None*, *schema=None*, *shared=None*)
Update space attributes.

This method updates title, description, schema, shared status or tagging rules of space, at least one of these params should have a non-default value to update the space.

Does update the space in the XYZ storage and this object mirroring it. Also apply the tags based on rules mentioned in *tagging_rules* dict.

Parameters

- **title** (*Optional[str]*) – A string representing the title of the space.
- **description** (*Optional[str]*) – A string representing a description of the space.

- **tagging_rules** (*Optional[Dict[str, str]]*) – A dict where the key is the tag to be applied to all features matching the JSON-path expression being the value.
- **schema** (*str*) – JSON object or URL to be added as schema for space.
- **shared** (*Optional[bool]*) – A boolean, if set to `True`, space will be shared with other users having XYZ account, they will be able to read from the space using their own token. If set to `False` space will be unshared.

Returns A response from API.

Return type Dict

Example:

```
>>> from xyzspaces import XYZ
>>> xyz = XYZ(credentials="XYZ_TOKEN")
>>> space = xyz.spaces.new(title="new space", description="new space")
>>> tagging_rules = {"large": "$.features[?(@.properties.area>=500)]"}
>>> space.update(title="updated title",
...              description="updated description",
...              tagging_rules=tagging_rules)
```

delete()
Delete this space object.

get_statistics()
Get statistics for this space object.

Returns A JSON object with some statistics about the specified space.

Return type dict

search(*tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*, *geo_dataframe=None*)
Search features for this space object.

Parameters

- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – A dict to represent additional filter on features to be searched. Examples:
  - `params={"p.name": "foo"}` returns all features with a value of property `name` equal to `foo`.
  - `params={"p.name!": "foo"}` returns all features with a value of property `name` not qual to `foo`.
  - `params={"p.count=gte": "10"}` returns all features with a value of property `count` greater than or equal to `10`.
  - `params={"p.count=lte": "10"}` returns all features with a value of property `count` less than or equal to `10`.
  - `params={"p.count=gt": "10"}` returns all features with a value of property `count` greater than `10`.
  - `params={"p.count=lt": "10"}` returns all features with a value of property `count` less than `10`.

- params={"p.name=cs": "bar"} returns all features with a value of property name which contains bar.

- **selection** (*Optional[List[str]]*) – A list, only these properties will be returned in features result list.
- **skip_cache** (*Optional[bool]*) – If set to True the response is not returned from cache. Default is False.
- **geo_dataframe** (*Optional[bool]*) – A boolean if set to True searched features will be yield as single Geopandas Dataframe.

Yields A Feature object by default. If param geo_dataframe is True then yields single Geopandas Dataframe.

Return type Generator[geojson.feature.Feature, None, None]

iter_feature (*limit=100*)
Iterate over features in this space object.

Parameters **limit** (*int*) – A max. number of features to return in the result.

Yields A Feature object.

Return type Generator[geojson.feature.Feature, None, None]

get_feature (*feature_id*)
Retrieve one GeoJSON feature with given ID from this space.

Parameters **feature_id** (*str*) – Feature id which is to fetched.

Returns A GeoJSON representing a feature with the specified feature ID inside the space.

Return type geojson.base.GeoJSON

add_feature (*data*, *feature_id=None*, *add_tags=None*, *remove_tags=None*)
Add one GeoJSON feature with given ID in this space.

Parameters

- **data** (*Union[geojson.base.GeoJSON, Dict]*) – A JSON object describing the feature to be added.
- **feature_id** (*Optional[str]*) – A string with the ID of the feature to be created.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the feature.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the feature.

Returns A GeoJSON representing a feature.

Return type geojson.base.GeoJSON

update_feature (*feature_id*, *data*, *add_tags=None*, *remove_tags=None*)
Update one GeoJSON feature with given ID in this space.

Parameters

- **feature_id** (*str*) – A string with the ID of the feature to be modified.
- **data** (*dict*) – A JSON object describing the feature to be changed.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the feature.

- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the feature.

Returns A GeoJSON representing a feature.

Return type geojson.base.GeoJSON

delete_feature(*feature_id*)

Delete one GeoJSON feature with given ID in this space.

Parameters **feature_id** (*str*) – A string with the ID of the feature to be deleted.

Returns An empty string if the operation was successful.

get_features(*feature_ids*, *geo_dataframe=None*)

Retrieve one GeoJSON feature with given ID from this space.

Parameters

- **feature_ids** (*List[str]*) – A list of feature_ids.
- **geo_dataframe** (*Optional[bool]*) – A boolean if set to `True` features will be returned as single Geopandas Dataframe.

Returns A feature collection with all features inside the specified space. If param `geo_dataframe` is set to `True` then return features in single Geopandas Dataframe.

Return type Union[geojson.base.GeoJSON, geopandas.geodataframe.GeoDataFrame]

add_features(*features*, *add_tags=None*, *remove_tags=None*, *features_size=2000*, *chunk_size=1*, *id_properties=None*, *mutate=True*)

Add GeoJSON features to this space.

As API has a limitation on the size of features, features are divided into chunks, and multiple processes will process those chunks. Each chunk has a number of features based on the value of `features_size`. Each process handles chunks based on the value of `chunk_size`.

Parameters

- **features** (*Union[geojson.base.GeoJSON, Dict]*) – A JSON object describing one or more features to add.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the features.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the features.
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1 to get better results in terms of performance.
- **id_properties** (*Optional[List[str]]*) – List of properties name from which id to be generated if id does not exists for a feature.
- **mutate** (*Optional[bool]*) – If True will update the existing features object passed, this will prevent making copy of the features object which may help to improving performance.

Returns A GeoJSON representing a feature collection.

Return type geojson.base.GeoJSON

_upload_features(*features*, *ids_map*, *add_tags=None*, *remove_tags=None*, *id_properties=None*)

Parameters

- **add_tags** (*Optional[List[str]]*) –
- **remove_tags** (*Optional[List[str]]*) –
- **id_properties** (*Optional[List[str]]*) –

_process_features (*features*, *id_properties*, *ids_map*)

_gen_id_from_properties (*feature*, *id_properties*)

update_features (*features*, *add_tags=None*, *remove_tags=None*)
Update GeoJSON features in this space.

Parameters

- **features** (*geojson.base.GeoJSON*) – A JSON object describing one or more features to modify.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the features.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the features.

Returns A GeoJSON representing a feature collection.

Return type geojson.base.GeoJSON

delete_features (*feature_ids*, *tags=None*)
Delete GeoJSON features in this space.

Parameters

- **feature_ids** (*List[str]*) – A list of feature IDs to delete.
- **tags** (*Optional[List[str]]*) – A list of strings describing tags the features to be deleted must have.

Returns A response from API.

features_in_bbox (*bbox*, *tags=None*, *clip=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*, *clustering=None*, *clustering_params=None*, *geo_dataframe=None*)
Get features inside some given bounding box.

Parameters

- **bbox** (*List[Union[int, float]]*) – A list of four numbers representing the West, South, East and North margins, respectively, of the bounding box.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **clip** (*Optional[bool]*) – A Boolean indicating if the result should be clipped (default: False).
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – A dict to represent additional filter on features to be searched. Examples:
  - `params={"p.name": "foo"}` returns all features with a value of property `name` equal to `foo`.
  - `params={"p.name!": "foo"}` returns all features with a value of property `name` not qual to `foo`.

  - `params={"p.count=gte": "10"}` returns all features with a value of property `count` greater than or equal to `10`.
  - `params={"p.count=lte": "10"}` returns all features with a value of property `count` less than or equal to `10`.
  - `params={"p.count=gt": "10"}` returns all features with a value of property `count` greater than `10`.
  - `params={"p.count=lt": "10"}` returns all features with a value of property `count` less than `10`.
  - `params={"p.name=cs": "bar"}` returns all features with a value of property `name` which contains `bar`.
- **selection** (*Optional[List[str]]*) – A list, only these properties will be returned in features result list.
- **skip_cache** (*Optional[bool]*) – If set to `True` the response is not returned from cache. Default is `False`.
- **clustering** (*Optional[str]*) – ...
- **clustering_params** (*Optional[dict]*) – ...
- **geo_dataframe** (*Optional[bool]*) – A boolean if set to `True` searched features will be yield as single Geopandas Dataframe.

Yields A Feature object by default. If param `geo_dataframe` is True then yields single Geopandas Dataframe.

Return type Generator[geojson.feature.Feature, None, None]

features_in_tile(*tile_type*, *tile_id*, *tags=None*, *clip=None*, *params=None*, *selection=None*, *skip_cache=None*, *clustering=None*, *clustering_params=None*, *margin=None*, *limit=None*, *geo_dataframe=None*)

Get features in tile.

Parameters

- **tile_type** (*str*) – A string with the name of a tile type, one of "quadkeys", "web", "tms" or "here". See below.
- **tile_id** (*str*) – A string holding a valid tile ID according to the specified `tile_type`.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **clip** (*Optional[bool]*) – A Boolean indicating if the result should be clipped (default: False).
- **params** (*Optional[dict]*) – A dict to represent additional filter on features to be searched. Examples:
  - `params={"p.name": "foo"}` returns all features with a value of property `name` equal to `foo`.
  - `params={"p.name!": "foo"}` returns all features with a value of property `name` not qual to `foo`.
  - `params={"p.count=gte": "10"}` returns all features with a value of property `count` greater than or equal to `10`.
  - `params={"p.count=lte": "10"}` returns all features with a value of property `count` less than or equal to `10`.

- `params={"p.count=gt": "10"}` returns all features with a value of property `count` greater than `10`.
- `params={"p.count=lt": "10"}` returns all features with a value of property `count` less than `10`.
- `params={"p.name=cs": "bar"}` returns all features with a value of property `name` which contains `bar`.

- **selection** (*Optional[List[str]]*) – A list, only these properties will be returned in features result list.
- **skip_cache** (*Optional[bool]*) – If set to `True` the response is not returned from cache. Default is `False`.
- **clustering** (*Optional[str]*) – ...
- **clustering_params** (*Optional[dict]*) – ...
- **margin** (*Optional[int]*) – ...
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **geo_dataframe** (*Optional[bool]*) – A boolean if set to `True` searched features will be yield as single Geopandas Dataframe.

Yields A Feature object by default. If param `geo_dataframe` is True then yields single Geopandas Dataframe.

Raises **ValueError** – If *tile_type* is invalid, valid tile_types are *quadkeys*, *web*, *tms* and *here*.

Return type Generator[geojson.feature.Feature, None, None]

spatial_search(*lat=None*, *lon=None*, *ref_space_id=None*, *ref_feature_id=None*, *radius=None*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*, *geo_dataframe=None*)

Get features with radius search.

Parameters

- **lat** (*Optional[float]*) – A float in WGS'84 decimal degree (-90 to +90) of the center Point.
- **lon** (*Optional[float]*) – A float in WGS'84 decimal degree (-180 to +180) of the center Point.
- **ref_space_id** (*Optional[str]*) – A string as alternative for defining center coordinates, it is possible to reference a geometry in a space, hence it is needed to provide the `ref_space_id` where the referenced feature is stored. Always to use in combination with `ref_feature_id`.
- **ref_feature_id** (*Optional[str]*) – A string as unique identifier of a feature in the referenced space. The geometry of that feature gets used for the spatial query. Always to use in combination with `ref_space_id`.
- **radius** (*Optional[int]*) – An int in meter which defines the diameter of the search request.
- **tags** (*Optional[List[str]]*) – A list of strings holding tag values.
- **limit** (*Optional[int]*) – A max. number of features to return in the result.
- **params** (*Optional[dict]*) – A dict to represent additional filter on features to be searched. Examples:

- `params={"p.name": "foo"}` returns all features with a value of property `name` equal to `foo`.
- `params={"p.name!": "foo"}` returns all features with a value of property `name` not qual to `foo`.
- `params={"p.count=gte": "10"}` returns all features with a value of property `count` greater than or equal to `10`.
- `params={"p.count=lte": "10"}` returns all features with a value of property `count` less than or equal to `10`.
- `params={"p.count=gt": "10"}` returns all features with a value of property `count` greater than `10`.
- `params={"p.count=lt": "10"}` returns all features with a value of property `count` less than `10`.
- `params={"p.name=cs": "bar"}` returns all features with a value of property `name` which contains `bar`.

- **`selection`** (*`Optional[List[str]]`*) – A list of strings holding properties values.
- **`skip_cache`** (*`Optional[bool]`*) – A Boolean if set to `True` the response is not returned from cache.
- **`geo_dataframe`** (*`Optional[bool]`*) – A boolean if set to `True` searched features will be yield as single Geopandas Dataframe.

Yields A Feature object by default. If param `geo_dataframe` is True then yields single Geopandas Dataframe.

Return type Generator[geojson.feature.Feature, None, None]

`spatial_search_geometry`(*data*, *radius=None*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*, *divide=False*, *cell_width=None*, *units='m'*, *chunk_size=1*, *geo_dataframe=None*)

Search features which intersect the provided geometry.

Parameters

- **`data`** (*`dict`*) – A JSON object which is getting used for the spatial search.
- **`radius`** (*`Optional[int]`*) – An int which defines the diameter(meters) of the search request.
- **`tags`** (*`Optional[List[str]]`*) – A list of strings holding tag values.
- **`limit`** (*`Optional[int]`*) – A max. number of features to return in the result.
- **`params`** (*`Optional[dict]`*) – A dict to represent additional filter on features to be searched. Examples:
  - `params={"p.name": "foo"}` returns all features with a value of property `name` equal to `foo`.
  - `params={"p.name!": "foo"}` returns all features with a value of property `name` not qual to `foo`.
  - `params={"p.count=gte": "10"}` returns all features with a value of property `count` greater than or equal to `10`.
  - `params={"p.count=lte": "10"}` returns all features with a value of property `count` less than or equal to `10`.

- `params={"p.count=gt": "10"}` returns all features with a value of property `count` greater than `10`.
- `params={"p.count=lt": "10"}` returns all features with a value of property `count` less than `10`.
- `params={"p.name=cs": "bar"}` returns all features with a value of property `name` which contains `bar`.

- **selection** (*Optional[List[str]]*) – A list of strings holding properties values.
- **skip_cache** (*Optional[bool]*) – A Boolean if set to `True` the response is not returned from cache.
- **divide** (*Optional[bool]*) – To divide geometry if the resultant features count is large.
- **cell_width** (*Optional[float]*) – Width of each cell in which geometry is to be divided in units specified, default values is meters.
- **units** (*Optional[str]*) – Unit for cell_width please refer, https://github.com/omanges/turfpy/blob/master/measurements.md#units-type
- **chunk_size** (*int*) – Number of chunks each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1 to get better results in terms of performance.
- **geo_dataframe** (*Optional[bool]*) – A boolean if set to `True` searched features will be yield as single Geopandas Dataframe.

Yields A Feature object by default. If param `geo_dataframe` is True then yields as single Geopandas Dataframe.

Return type Generator[geojson.feature.Feature, None, None]

_spatial_search_geometry(*data*, *feature_list*, *radius=None*, *tags=None*, *limit=None*, *params=None*, *selection=None*, *skip_cache=None*)

Parameters

- **data** (*dict*) –
- **feature_list** (*List[dict]*) –
- **radius** (*Optional[int]*) –
- **tags** (*Optional[List[str]]*) –
- **limit** (*Optional[int]*) –
- **params** (*Optional[dict]*) –
- **selection** (*Optional[List[str]]*) –
- **skip_cache** (*Optional[bool]*) –

add_features_geojson(*path*, *encoding='utf-8'*, *features_size=2000*, *chunk_size=1*)

Add features in space from a GeoJSON file.

As API has a limitation on the size of features, features are divided into chunks, and multiple processes will process those chunks. Each chunk has a number of features based on the value of `features_size`. Each process handles chunks based on the value of `chunk_size`.

Parameters

- **path** (*str*) – Path to the GeoJSON file.

- **encoding** (*str*) – A string to represent the type of encoding.
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.

add_features_csv(*path*, *lon_col*, *lat_col*, *id_col=''*, *alt_col=''*, *delimiter=','*, *add_tags=None*, *remove_tags=None*, *features_size=2000*, *chunk_size=1*, *id_properties=None*)

Add features in space from a csv file.

As API has a limitation on the size of features, features are divided into chunks, and multiple processes will process those chunks. Each chunk has a number of features based on the value of `features_size`. Each process handles chunks based on the value of `chunk_size`.

Parameters

- **path** (*str*) – Path to csv file.
- **lon_col** (*str*) – Name of the column for longitude coordinates.
- **lat_col** (*str*) – Name of the column for latitude coordinates.
- **id_col** (*Optional[str]*) – Name of the column for feature id's.
- **alt_col** (*Optional[str]*) – Name of the column for altitude, if not provided altitudes with default value 0.0 will be added.
- **delimiter** (*Optional[str]*) – delimiter which should be used to process the csv file.
- **add_tags** (*Optional[List[str]]*) – A list of strings describing tags to be added to the features.
- **remove_tags** (*Optional[List[str]]*) – A list of strings describing tags to be removed from the features.
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1 to get better results in terms of performance.
- **id_properties** (*Optional[List[str]]*) – List of properties name from which id to be generated if id does not exists for a feature.

Raises **Exception** – If values of params *lat_col*, *lon_col*, *id_col* do not match with column names in csv file.

cluster(*clustering*, *clustering_params=None*)

Apply clustering algorithm for the space data.

Parameters

- **clustering** (*str*) – Name of the clustering algorithm. Available values : hexbin, quadbin
- **clustering_params** (*Optional[dict]*) – Parameters for the clustering algorithm. Please refer : https://www.here.xyz/api/devguide/usingclustering/

Returns GeoJSON.

Raises **Exception** – If bounding box is not present in space statistics.

Return type dict

Example:

```
>>> from xyzspaces import XYZ
>>> xyz = XYZ(credentials="XYZ_TOKEN")
>>> space = xyz.spaces.from_id(space_id="existing-space-id")
>>> space.cluster(clustering="hexbin")
```

isshared()
Return the shared status of the space.

Returns A boolean to indicate shared status of the space.

Return type bool

add_features_shapefile(*path*, *features_size=2000*, *chunk_size=1*, *encoding='utf-8'*)
Upload shapefile to the space.

Parameters

- **path** (*str*) – A string representing full path of the shapefile. For zipped shapefile prepend `zip://` before path of shapefile.
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.
- **encoding** (*str*) – A string to represent the type of encoding.

Example: >>> from xyzspaces import XYZ >>> xyz = XYZ(credentials="XYZ_TOKEN") >>> space = xyz.spaces.from_id(space_id="existing-space-id") >>> space.add_features_shapefile(path="shapefile.shp")

add_features_wkt(*path*)
To upload data from wkt file to a space

Parameters **path** (*str*) – Path to wkt file

add_features_gpx(*path*, *features_size=2000*, *chunk_size=1*)
Upload data from gpx file to the space.

Parameters

- **path** (*str*) – A string representing full path of the gpx file.
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.

add_features_kml(*path*, *features_size=2000*, *chunk_size=1*)
To upload data from kml file to a space

Parameters

- **path** (*str*) – Path to kml file
- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.

add_features_geobuf(*path*, *features_size=2000*, *chunk_size=1*)
To upload data from geobuff file to a space.

Parameters

- **path** (*str*) – Path to geobuff file.

- **features_size** (*int*) – An int representing a number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.

add_features_geopandas(*data*, *features_size=2000*, *chunk_size=1*)
Add features from GeoPandas dataframe to a space.

Parameters

- **data** (*geopandas.geodataframe.GeoDataFrame*) – GeoPandas dataframe to be uploaded
- **features_size** (*int*) – The number of features to upload at a time.
- **chunk_size** (*int*) – Number of chunks for each process to handle. The default value is 1, for a large number of features please use *chunk_size* greater than 1.

1.2.8 xyzspaces.tools module

This is a preliminary collection of little tools that use XYZ.

xyzspaces.tools.**subset_geojson**(*token*, *gj*, *bbox=None*, *tile_type=None*, *tile_id=None*, *clip=False*, *lat=None*, *lon=None*, *radius=None*)
Return a subset of the GeoJSON object inside some bbox or map tile or radius.

This will create a temporary space, add the provided GeoJSON object, perform the bbox or tile subsetting and return the resulting GeoJSON object after deleting the temporary space again.

Parameters

- **token** (*str*) – A string containing the XYZ API token.
- **gj** (*dict*) – The GeoJSON data object.
- **bbox** (*Optional[List[float]]*) – The bounding box described as a list of four numbers (its West, South, East, and North margins).
- **tile_type** (*Optional[str]*) – The tile type, for now one of: ...
- **tile_id** (*Optional[str]*) – The tile ID, a string composed of digits to identify the tile according to the specified tile type.
- **clip** (*Optional[bool]*) – A Boolean to indicate if the features should be clipped at the tile or bbox.
- **lat** (*Optional[float]*) – A float to represent latitude.
- **lon** (*Optional[float]*) – A float to represent longitude.
- **radius** (*Optional[int]*) – An int in meter which defines the diameter of the search request. Should be provided with `lat` and `lon` for spatial search.

Returns A GeoJSON object covering the desired tile or bbox subset of the original GeoJSON object.

Raises **ValueError** – If the wrong combination of `bbox`, `tile_type` and `tile_id`, `lat` and `lon` was provided.

Return type dict

1.2.9 xyzspaces.utils module

This is a collection of utilities for using XYZ Hub.

Actually, they are almost unspecific to any XYZ Hub functionality, apart from *feature_to_bbox()*, but convenient to use.

xyzspaces.utils.**join_string_lists**(***kwargs*)
Convert named lists of strings to one dict with comma-separated strings.

Parameters **kwargs** – Lists of strings

Returns Converted dict.

Return type dict

Example:

```
>>> join_string_lists(foo=["a", "b", "c"], bar=["a", "b"], foobar=None)
{"foo": "a,b,c", "bar": "a,b"}
```

xyzspaces.utils.**feature_to_bbox**(*feature*)
Extract bounding box from GeoJSON feature rectangle.

Parameters **feature** (*dict*) – A dict representing a GeoJSON feature.

Returns A list of four floats representing West, South, East and North margins of the resulting bounding box.

Return type List[float]

xyzspaces.utils.**get_xyz_token**()
Read and return the value of the environment variable XYZ_TOKEN.

Returns The string value of the environment variable or an empty string if no such variable could be found.

Return type str

xyzspaces.utils.**grouper**(*size*, *iterable*, *fillvalue=None*)
Create groups of *size* each from given iterable.

Parameters

- **size** – An int representing size of each group.
- **iterable** – An iterable.
- **fillvalue** – Value to put for the last group.

Returns A generator.

xyzspaces.utils.**wkt_to_geojson**(*wkt_data*)
Converts wkt to geojson

Parameters **wkt_data** (*str*) – wkt data to be converted

Returns Geojson

Return type dict

xyzspaces.utils.**grid**(*bbox*, *cell_width*, *cell_height*, *units*)
This function generates the grids for the given bounding box

Parameters

- **bbox** – bounding box coordinates

- **cell_width** – Cell width in specified in units
- **cell_height** – Cell height in specified in units
- **units** – Units for given sizes

Returns FeatureCollection of grid boxes genarated for the giving bounding box

xyzspaces.utils.**divide_bbox**(*feature*, *cell_width=None*, *units='m'*)

Divides the given feature into grid boxes as per given cell width

Parameters

- **feature** (*dict*) – Feature to be divide in grid
- **cell_width** (*Optional[float]*) – Width of each grid boxs
- **units** (*Optional[str]*) – Units for the width of grid boxs

Returns List of features in which the input feature is divided

xyzspaces.utils.**flatten_geometry**(*data*)

Flatten the geometries in the given GeoPandas dataframe. Flatten geometry is formed by extracting individual geometries from GeometryCollection, MultiPoint, MultiLineString, MultiPolygon.

Parameters **data** (*geopandas.geodataframe.GeoDataFrame*) – GeoPandas dataframe to be flatten

Returns Flat GeoPandas dataframe

Return type geopandas.geodataframe.GeoDataFrame

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

X

INDEX

Symbols

A

C

D

E

F

G

H

I

J

L

M

N

P