helios Documentation

Release 3.0.1

Michael Bayer

May 22, 2019

User Guide

Python SDK for the Helios APIs.

Helios® weather analytics from Harris Corporation provide fast and accurate local ground weather intelligence to assist organizations with real-time decision making. Helios analyzes content from thousands of existing public and private video cameras, providing immediate confirmation of ground weather condition changes at a detailed local level. For more details, refer to helios.earth.

The Helios SDK brings the core API functionality along with extensions to Python. Many of the capabilities are thread-enabled allowing for batch jobs. The overall goal is to provide the tools necessary to quickly begin using the Helios product.

For further developer information, refer to the Helios developer documentation.

CHAPTER 1

Installation

1.1 Install from PyPI (recommended)

```
pip install helios-sdk
```

1.2 Install from source (bleeding edge)

Clone the GitHub repository:

```
git clone https://github.com/harris-helios/helios-sdk-python.git
```

Then `cd` to the helios-sdk-python directory and run the install command:

```
cd helios-sdk-python
pip install .
```

CHAPTER 2

Authentication

All Helios API methods require valid authentication and are protected using the OAuth 2.0 "client credentials" flow. The general process for authenticating requests involves first requesting an access token using the developer API key pair, and then requesting protected data using the access token. Request access if you would like to obtain an API key.

2.1 Using Environment Variables

1. Add **"helios_client_id"**: "your ID key"
2. Add **"helios_client_secret"**: "your secret key"
3. Add **"helios_api_url"**: "API URL associated with your account credentials"
    - **"helios_api_url"** is optional.

2.2 Using an Authentication File

1. Create a **".helios"** directory in your home directory.
2. Create a **"credentials.json"** file in your **".helios"** directory.
3. Copy and paste the following into the **"credentials.json"** file and fill in your authentication values.
    - **"helios_api_url"** is optional. If you do not need a custom API URL leave this out of your json file or set to null.

```
{
    "helios_client_id" : "your ID key" ,
    "helios_client_secret" : "your secret key",
    "helios_api_url" : null
}
```

For more information refer to the Helios authentication documentation.

CHAPTER 3

HeliosSession

Manually creating a *HeliosSession* provides advanced access to a session configuration.

A Helios *HeliosSession* depends on properly established authentication procedures. See *Authentication* for more information. It also stores configuration parameters and your authentication information and will fetch an API token. This token is required for any API calls.

3.1 Creating a Session

If authentication is stored on your machine starting a session is straightforward. Create a *HeliosSession* instance without any inputs.

```
import helios

sess = helios.HeliosSession()
```

This will automatically fetch a token.

If successful, the `sess` instance will now have all the authentication information needed to being using the core APIs.

3.2 HeliosSession Configuration Parameters

A *HeliosSession* can be initialized with various configuration parameters.

E.g. Limit the maximum number of threads:

```
import helios

sess = helios.HeliosSession(max_threads=50)
```

E.g. Override the base directory for storing tokens/credentials.json files:

```
import helios

sess = helios.HeliosSession(base_dir='/tmp/custom')
```

E.g. Using custom credentials outside of the standard *Authentication* methods:

```
helios_client_id = '*your ID key*',
helios_client_secret = '*your secret key*',
helios_api_url = '*optional API URL override*'

sess = helios.Session(
    client_id=helios_client_id,
    client_secret=helios_client_secret,
    api_url=helios_api_url
)
```

3.3 Creating Core API Instances

Using a custom *HeliosSession* to create core API instances is straightforward.

```
import helios

sess = helios.HeliosSession(max_threads=50)
alerts = sess.client('alerts')
cameras = sess.client('cameras')
collections = sess.client('collections')
observations = sess.client('observations')
```

3.4 Default HeliosSession

For most cases the default *HeliosSession* will suffice. The default is used when creating instances via the top-level `client` call.

E.g.

```
import helios

alerts = helios.client('alerts')
```

CHAPTER 4

Using the Core APIs

4.1 Creating Core API Instances

Creating an instance of one of the core APIs is done via `client`.

```
import helios

alerts = helios.client('alerts')
cameras = helios.client('cameras')
collections = helios.client('collections')
observations = helios.client('observations')
```

4.2 Examples

4.2.1 Find alerts

```
import helios

alerts = helios.client('alerts')

# Retrieve ressults for New York.
ny_alert_results, failed = alerts.index(state='New York')

# Gather the camera IDs from the results.
ny_alert_ids = ny_alert_results.id

return ny_alert_ids
```

- `ny_alert_results` is an instance of *AlertsFeatureCollection*.

4.2.2 Find camera times and download images

```
import helios
import numpy as np

cameras = helios.client('cameras')

# Find cameras in Maryland.
md_cam_results, failures = cameras.index(state='Maryland')
cam_id = md_cam_results.id[0]

# Find image times for the given camera id.
image_times = cameras.images(cam_id, '2018-01-01')

# Download the images.
show_image_results, failures = cameras.show_image(
    cam_id, image_times, out_dir='/temp/data', return_image_data=True
)

return show_image_results, failures
```

- `md_cam_results` is an instance of *CamerasFeatureCollection*.
  - Access the list of individual features by calling `md_cam_results.features`.
- `show_image_results` is an instance of `ImageCollection`.

4.2.3 Find observations and work with collections

```
import helios
import requests
from helios.utilities import parsing_utils

observations = helios.client('observations')
collections = helios.collections('collections')

# Find Observations
index_results, failures = observations.index(
    state='georgia',
    sensors='sensors[visibility]=0',
    time_min='2018-02-10T18:00Z',
    time_max='2018-02-10T18:15Z'
)

# Get id for each observation feature.
ids = [x.id for x in index_results.features]

# Convenience properties also exist for combining attributes from all features.
ids = index_results.id

# Create new collection.
new_id = collections.create(
    'Temp Collection', 'example collection', ['test', 'temp']
)

# Add Observations to collection.
```

(continues on next page)

(continued from previous page)

```
payload = [{'observation_id': x} for x in ids]
add_result, failures = collections.add_image(new_id, payload)

# Check for http failures.
if len(add_result.failed) > 0:
    print('Failures occurred!')

# Simple data analysis - find all unique cameras for the added observation images.
ims = collections.images(new_id)
cams = set([parsing_utils.parse_camera(x) for x in ims])
```

- `index_results` is an instance of *ObservationsFeatureCollection*.
  - Access the list of individual features by calling `index_results.features`.

4.2.4 Find Observations Based on Sensor Value

```
import helios

obs_inst = helios.client('observations')
state = 'Maryland'
bbox = [-169.352,1.137,-1.690,64.008]
sensors = 'sensors[visibility][min]=0&sensors[visibility][max]=1'
results, failures = obs.index(state=state, bbox=bbox, sensors=sensors)
```

4.2.5 Find Observations Transitions

Example for transition from dry/wet to partial/full-snow road conditions:

```
import helios
obs_inst = helios.client('observations')
# transition from dry/wet to partial/fully-covered snow roads
sensors = 'sensors[road_weather][data][min]=6&sensors[road_weather][prev][max]=3'
results, failures = obs.index(sensors=sensors_query)
```

CHAPTER 5

License

MIT License

Copyright (c) 2017 Harris Helios

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER 6

HeliosSession

Manager for the authorization token required to access the Helios API.

class helios.core.session.**Authentication**(*client_id*, *client_secret*, *api_url=None*)

Authentication API.

Parameters

- **client_id** (*str*) – API key ID.
- **client_secret** (*str*) – API key secret.
- **api_url** (*str*) – Override the default API URL.

get_access_token()

Gets a new access token.

Returns Helios API access token.

Return type *Token*

get_current_user(*token*)

Return a compact list of current token attributes, such as the token expiration date.

Parameters **token** (Token) – A Helios Token.

Returns Token attributes.

Return type dict

Raises ValueError – If current token is invalid or expired.

class helios.core.session.**HeliosSession**(*client_id=None*, *client_secret=None*, *api_url=None*, *base_directory=None*, *max_threads=64*, *token_expiration_threshold=60*, *ssl_verify=True*)

Manages configuration and authentication details for the Helios APIs.

The session will handle acquiring and verifying tokens as well as various developer options.

Parameters

- **client_id** (*str, optional*) – API key ID.
- **client_secret** (*str, optional*) – API key secret.
- **api_url** (*str, optional*) – Override the default API URL.
- **base_directory** (*str, optional*) – Override the base directory for storage of tokens and credentials.json file.
- **max_threads** (*int, optional*) – The maximum number of threads allowed for batch calls.
- **token_expiration_threshold** (*int, optional*) – The number of minutes to allow before token expiration. E.g. if the token will expire in 29 minutes and the threshold is set to 30, a new token will be acquired because expiration time is below the threshold.
- **ssl_verify** (*bool, optional*) – Use SSL verification for all HTTP requests. Defaults to True.

auth_header
Authentication header for API requests.

client(*name*)
Gets a core API instance using the current HeliosSession.

Parameters **name** (*str*) – Name of API.

Returns Core API instance.

get_new_token()
Gets a new token for the current Helios session.

read_token()
Reads token from file.

verify_token()
Verifies that token has not expired.

If the token has expired a new token will be acquired.

write_token()
Writes token to file.

class helios.core.session.**Token**(*access_token*, *expires_in=None*, ***kwargs*)
Helios token.

Parameters

- **access_token** (*str*) – Access token.
- **expires_in** (*int*) – Seconds until expiration. WARNING: this value can be misleading. If reading from a token file the value will not be current.
- **kwargs** – Any other optional token attributes.

auth_header
Authentication header for API requests.

expiration
The expiration time.

expired
True if token is expired, False otherwise.

to_json()
Serializes token to JSON.

Returns JSON serialized string.

Return type str

CHAPTER 7

Alerts API

Helios Alerts API.

Methods are meant to represent the core functionality in the developer documentation. Some may have additional functionality for convenience.

class helios.alerts_api.**Alerts**(*session=None*)

Helios alerts provide real-time severe weather alerts.

National Weather Service:

- Severe weather alerts for the United States are provided by the National Weather Service. These alerts cover events such as Flood Warnings, Severe Thunderstorm Warnings, and Special Weather Statements.

Helios:

- Alerts generated by Helios are based on the sensor measurements from the Observations API. These alerts represent regional areas with a high detection confidence and currently include: Road Wetness Watch, Poor Visibility Watch, and Heavy Precip Watch.

index(***kwargs*)

Get alerts matching the provided spatial, text, or metadata filters.

The maximum skip value is 4000. If this is reached, truncated results will be returned. You will need to refine your query to avoid this.

Parameters ****kwargs** – Any keyword arguments found in the alerts_index_documentation.

Returns

A tuple containing:

feature_collection (*AlertsFeatureCollection*): Alerts feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

show(*alert_ids*)

Get attributes for alerts.

Parameters **alert_ids** (*str or list of strs*) – Helios alert ID(s).

Returns

A tuple containing:

feature_collection (*AlertsFeatureCollection*): Alerts feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

class helios.alerts_api.**AlertsFeature**(*feature*)

Individual Alert GeoJSON feature.

area_description

'areaDesc' value for the feature.

Type str

bbox

'bbox' value for the feature.

Type list of floats

category

'category' value for the feature.

Type str

certainty

'certainty' value for the feature.

Type str

country

'country' value for the feature.

Type str

description

'description' value for the feature.

Type str

effective

'effective' value for the feature.

Type str

event

'event' value for the feature.

Type str

expires

'expires' value for the feature.

Type str

headline

'headline' value for the feature.

Type str

id

'id' value for the feature.

Type str

json
Raw 'json' for the feature.

Type dict

origin
'origin' value for the feature.

Type str

severity
'severity' value for the feature.

Type str

states
'states' value for the feature.

Type list of strs

status
'status' value for the feature.

Type str

urgency
'urgency' value for the feature.

Type str

class helios.alerts_api.**AlertsFeatureCollection**(*features*)
Collection of GeoJSON features obtained via the Alerts API.

Convenience properties are available to extract values from every feature.

features
All features returned from a query.

Type list of *AlertsFeature*

area_description
'areaDesc' values for every feature.

bbox
'bbox' values for every feature.

category
'category' values for every feature.

certainty
'certainty' values for every feature.

country
'country' values for every feature.

description
'description' values for every feature.

effective
'effective' values for every feature.

event
'event' values for every feature.

expires
'expires' values for every feature.

headline
'headline' values for every feature.

id
'id' values for every feature.

json
Raw 'json' for every feature.

origin
'origin' values for every feature.

severity
'severity' values for every feature.

states
'states' values for every feature.

status
'status' values for every feature.

urgency
'urgency' values for every feature.

CHAPTER 8

Cameras API

Helios Cameras API.

Methods are meant to represent the core functionality in the developer documentation. Some may have additional functionality for convenience.

class helios.cameras_api.**Cameras**(*session=None*)

The Cameras API provides access to all cameras in the Helios Network.

images(*camera_id*, *start_time*, *end_time=None*, *limit=500*)

Get the image times available for a given camera in the media cache.

The media cache contains all recent images archived by Helios, either for internal analytics or for end user recording purposes.

Parameters

- **camera_id** (*str*) – Camera ID.
- **start_time** (*str*) – Starting image timestamp, specified in UTC as an ISO 8601 string (e.g. 2014-08-01 or 2014-08-01T12:34:56.000Z).
- **end_time** (*str, optional*) – Ending image timestamp, specified in UTC as an ISO 8601 string (e.g. 2014-08-01 or 2014-08-01T12:34:56.000Z).
- **limit** (*int, optional*) – Number of images to be returned, up to a max of 500. Defaults to 500.

Returns Image times.

Return type list of strs

index(***kwargs*)

Get cameras matching the provided spatial, text, or metadata filters.

The maximum skip value is 4000. If this is reached, truncated results will be returned. You will need to refine your query to avoid this.

Parameters ****kwargs** – Any keyword arguments found in the cameras_index_documentation.

Returns

A tuple containing:

feature_collection (*CamerasFeatureCollection*): Cameras feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

show(*camera_ids*)
Get attributes for cameras.

Parameters **camera_ids** (*str or list of strs*) – Helios camera ID(s).

Returns

A tuple containing:

feature_collection (*CamerasFeatureCollection*): Cameras feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

show_image(*times*, *camera_id*, *out_dir=None*, *return_image_data=False*)
Get images from the media cache.

The media cache contains all recent images archived by Helios, either for internal analytics or for end user recording purposes.

Parameters

- **times** (*str or list of strs*) – Image times, specified in UTC as an ISO 8601 string (e.g. 2017-08-01 or 2017-08-01T12:34:56.000Z). The image with the closest matching timestamp will be returned.
- **camera_id** (*str*) – Camera ID.
- **out_dir** (*optional, str*) – Directory to write images to. Defaults to None.
- **return_image_data** (*optional, bool*) – If True images will be available as PIL images in the returned ImageRecords. Defaults to False.

Returns

A tuple containing:

images (list of *ImageRecord*): All received images.

failed (list of :*ImageRecord*): Failed API calls.

Return type tuple

class helios.cameras_api.**CamerasFeature**(*feature*)
Individual Camera GeoJSON feature.

city
'city' value for the feature.

Type str

country
'country' value for the feature.

Type str

description
'description' value for the feature.

Type str

direction
'direction' value for the feature.

Type str

id
'id' value for the feature.

Type str

json
Raw 'json' for the feature.

Type dict

region
'region' value for the feature.

Type str

state
'state' value for the feature.

Type str

video
'video' value for the feature.

Type bool

class helios.cameras_api.**CamerasFeatureCollection**(*features*)
Collection of GeoJSON features obtained via the Cameras API.

Convenience properties are available to extract values from every feature.

features
All features returned from a query.

Type list of *CamerasFeature*

city
'city' values for every feature.

coordinates
'coordinate' values for every feature.

country
'country' values for every feature.

description
'description' values for every feature.

direction
'direction' values for every feature.

id
'id' values for every feature.

json
Raw 'json' for every feature.

region
'region' values for every feature.

state
'state' values for every feature.

CHAPTER 9

Collections API

Helios Collections API.

Methods are meant to represent the core functionality in the developer documentation. Some may have additional functionality for convenience.

class helios.collections_api.**Collections**(*session=None*)

The Collections API allows users to group and organize individual image frames.

Collections are intended to be short-lived resources and will be accessible for 90 days from the time the collection was created. After that time period has expired, the collection and all associated imagery will be removed from the system.

add_image(*assets*, *collection_id*)

Add images to a collection from Helios assets.

assets dictionary templates:

```
# Asset examples that can be included in the `assets` input list.
data = {'camera_id': ''}
data = {'camera_id': '', 'time': ''}
data = {'observation_id': ''}
data = {'collection_id': '', 'image': ''}
```

Usage example:

```
import helios

coll_inst = helios.client('collections')
camera_id = '...'
times = [...] # List of image times.
destination_id = '...'
data = [{'camera_id': camera_id, 'time': x} for x in times]
results, failures = coll_inst.add_image(data, destination_id)
```

Parameters

- **assets** (*dict or list of dicts*) – Data containing any of these payloads (camera_id), (camera_id, time), (observation_id), (collection_id, image). E.g. data = [{'camera_id': 'cam_01', time: '2017-01-01T00:00:000Z'}]
- **collection_id** (*str*) – Collection ID.

Returns

A tuple containing:

succeeded (list of *Record*): Successful API call records.

failed (list of *Record*): Failed API call records.

Return type tuple

copy(*collection_id*, *new_name*)

Copy a collection and its contents to a new collection.

Parameters

- **collection_id** (*str*) – Collection ID.
- **new_name** (*str*) – New collection name.

Returns New collection ID.

Return type str

create(*name*, *description*, *tags=None*)

Create a new collection.

Parameters

- **name** (*str*) – Display name for the collection.
- **description** (*str*) – Description for the collection.
- **tags** (*str or list of strs, optional*) – Optional comma-delimited keyword tags to be added to the collection.

Returns New collection ID.

Return type str

destroy(*collection_id*)

Delete an empty collection.

If the collection is not empty, delete will fail. Use the *empty* method to remove all imagery before calling this method.

Parameters **collection_id** (*str*) – Collection to delete.

Returns {ok: true}

Return type dict

empty(*collection_id*)

Bulk remove (up to 1000) images from a collection.

Parameters **collection_id** (*str*) – Collection to empty.

Returns {ok: true, total: 1000}

Return type dict

images(*collection_id*, *camera=None*, *old_flag=False*)
Get all image names in a given collection.

When using the optional camera input parameter only images from that camera will be returned.

Parameters

- **collection_id** (*str*) – Collection ID.
- **camera** (*str, optional*) – Camera ID to be found.
- **old_flag** (*bool, optional*) – Flag for finding old format image names. When True images that do not contain md5 hashes at the start of their name will be found.

Returns Image names.

Return type list of strs

index(***kwargs*)
Get collections matching the provided spatial, text, or metadata filters.

The maximum skip value is 4000. If this is reached, truncated results will be returned. You will need to refine your query to avoid this.

Parameters ****kwargs** – Any keyword arguments found in the collections_index_documentation.

Returns

A tuple containing:

feature_collection (*CollectionsFeatureCollection*): Collections feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

remove_image(*names*, *collection_id*)
Remove images from a collection.

Parameters

- **names** (*str or list of strs*) – List of image names to be removed.
- **collection_id** (*str*) – Collection ID to remove images from.

Returns

A tuple containing:

succeeded (list of *Record*): Successful API call records.

failed (list of *Record*): Failed API call records.

Return type tuple

show(*collection_id*, *limit=200*, *marker=None*)
Get the attributes and image list for collections.

The results will also contain image names available in the collection. These are limited to a maximum of 200 per query.

Parameters

- **collection_id** (*str*) – Collection ID.

- **limit** (*int, optional*) – Number of image names to be returned with each response. Defaults to 200. Max value of 200 is allowed.
- **marker** (*str, optional*) – Pagination marker. If the marker is an exact match to an existing image, the next image after the marker will be the first image returned. Therefore, for normal linked list pagination, specify the last image name from the current response as the marker value in the next request. Partial file names may be specified, in which case the first matching result will be the first image returned.

Returns A single collection feature.

Return type *CollectionsFeature*

show_image(*image_names*, *collection_id*, *out_dir=None*, *return_image_data=False*)
Get images from a collection.

Parameters

- **image_names** (*str or list of strs*) – Image names.
- **collection_id** (*str*) – Collection ID to add images into.
- **out_dir** (*optional, str*) – Directory to write images to. Defaults to None.
- **return_image_data** (*optional, bool*) – If True images will be available as PIL images in the returned ImageRecords. Defaults to False.

Returns

A tuple containing:

images (list of *ImageRecord*): All received images.

failed (*ImageRecord*): Failed API calls.

Return type tuple

update(*collections_id*, *name=None*, *description=None*, *tags=None*)
Update a collection.

Parameters

- **collections_id** (*str*) – Collection ID.
- **name** (*str, optional*) – Name to be changed to.
- **description** (*str, optional*) – Description to be changed to.
- **tags** (*str or list of strs, optional*) – Optional comma-delimited keyword tags to be changed to.

Returns Json response.

Return type dict

class helios.collections_api.**CollectionsFeature**(*feature*)
Individual Collection JSON result.

bucket
'bucket' value for the result.

Type str

created_at
'city' value for the result.

Type str

description
'created_at' value for the result.

Type str

id
'_id' value for the result.

Type str

images
'images' value for the result.

Type list of strs

json
Raw JSON result.

Type dict

name
'name' value for the result.

Type str

tags
'tags' value for the result.

Type list of strs

updated_at
'updated_at' value for the result.

Type str

user_id
'user_id' value for the result.

Type str

class helios.collections_api.**CollectionsFeatureCollection**(*features*)
Collection of features obtained via the Collections API.

Convenience properties are available to extract values from every feature.

features
All features returned from a query.

Type list of *CollectionsFeature*

bucket
'bucket' values for every result.

created_at
'city' values for every result.

description
'created_at' values for every result.

id
'_id' values for every result.

json
Raw 'json' for every feature.

name
'name' values for every result.

tags
'tags' values for every result.

updated_at
'updated_at' values for every result.

user_id
'user_id' values for every result.

CHAPTER 10

Observations API

Helios Observations API.

Methods are meant to represent the core functionality in the developer documentation. Some may have additional functionality for convenience.

class helios.observations_api.**Observations**(*session=None*)

The Observations API provides ground-truth data generated by the Helios analytics.

index(***kwargs*)

Get observations matching the provided spatial, text, or metadata filters.

The maximum skip value is 4000. If this is reached, truncated results will be returned. You will need to refine your query to avoid this.

Parameters ****kwargs** – Any keyword arguments found in the observations_index_documentation.

Returns

A tuple containing:

feature_collection (*ObservationsFeatureCollection*): Observations feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

preview(*observation_ids*, *out_dir=None*, *return_image_data=False*)

Get preview images from observations.

Parameters

- **observation_ids** (*str or list of strs*) – list of observation IDs.
- **out_dir** (*optional, str*) – Directory to write images to. Defaults to None.
- **return_image_data** (*optional, bool*) – If True images will be available as PIL images in the returned ImageRecords. Defaults to False.

Returns

A tuple containing:

images (list of *ImageRecord*): All received images.

failed (list of *ImageRecord*): Failed API calls.

Return type tuple

show(*observation_ids*)
Get attributes for observations.

Parameters **observation_ids** (*str or list of strs*) – Helios observation ID(s).

Returns

A tuple containing:

feature_collection (*ObservationsFeatureCollection*): Observations feature collection.

failed (list of *Record*): Failed API call records.

Return type tuple

class helios.observations_api.**ObservationsFeature**(*feature*)
Individual Observation GeoJSON feature.

city
'city' value for the feature.

Type str

country
'country' value for the feature.

Type str

description
'description' value for the feature.

Type str

id
'id' value for the feature.

Type str

json
Raw JSON feature.

Type dict

prev_id
'prev_id' value for the feature.

Type str

region
'region' value for the feature.

Type str

sensors
'sensors' value for the feature.

Type dict

state
'state' value for the feature.

Type str

time
'time' value for the feature.

Type str

class helios.observations_api.**ObservationsFeatureCollection**(*features*)
Collection of GeoJSON features obtained via the Observations API.

Convenience properties are available to extract values from every feature.

features
All features returned from a query.

Type list of `ObservationsFeature`

city
'city' values for every feature.

country
'country' values for every feature.

description
'description' values for every feature.

id
'id' values for every feature.

json
Raw 'json' for every feature.

observations
Observation data from the sensor block of each feature.

Data will be returned as a dictionary with a key for each sensor. Observation data for each sensor is a named tuple ease-of-use.

Each named tuple contains the sensor, time, data, prev, id, and prev_id.

prev_id
'prev_id' values for every feature.

region
'region' values for every feature.

sensors
'sensors' values for every feature.

state
'state' values for every feature.

time
'time' values for every feature.

CHAPTER 11

Structure

Base data structures for the SDK.

class helios.core.structure.**ImageRecord**(*name=None*, *filename=None*, ***kwargs*)

Record class for images.

Parameters

- **name** (*str*) – Name of image.
- **filename** (*str*) – Full path to image file that was written.

image

Alias for Record content attribute.

Returns Image data.

Return type PIL.Image.Image

ok

Check if failure occurred.

Returns False if error occurred, and True otherwise.

Return type bool

parameters

Function call parameters.

Returns Parameters dictionary.

Return type dict

class helios.core.structure.**Record**(*url=None*, *parameters=None*, *content=None*, *error=None*)

Individual query record.

Parameters

- **url** (*str*) – API URL.
- **parameters** (*dict*) – All parameters for current function or method call.

- **content** – Returned content. To be defined by method.
- **error** (*exception*) – Exception that occurred, if any.

ok
Check if failure occurred.

Returns False if error occurred, and True otherwise.

Return type bool

parameters
Function call parameters.

Returns Parameters dictionary.

Return type dict

class helios.core.structure.**RecordCollection**(*records=None*)
Class for handling query records. .. attribute:: records

Raw record data for debugging purposes.

type list of *Record*

failed
Records for queries that failed.

succeeded
Records for queries that succeeded.

CHAPTER 12

Utilities

12.1 json_utils

Helper functions for JSON objects.

helios.utilities.json_utils.**merge_json**(*data*, *keys*)
Merge JSON fields into a single list.

Keys can either be a single string or a list of strings signifying a chain of "keys" into the dictionary.

Parameters

- **data** (*list*) – Dictionary to merge data from.
- **keys** (*str or sequence of strs*) – A chain of keys into the dictionary to get to the field that will be merged.

Returns Merged values.

Return type list

helios.utilities.json_utils.**read_json_file**(*json_file*, ***kwargs*)
Read a json file.

Parameters

- **json_file** (*str*) – Full path to JSON file.
- ****kwargs** – Any keyword argument from the json.load method.

Returns JSON formatted dictionary.

Return type dict

helios.utilities.json_utils.**read_json_string**(*json_string*, ***kwargs*)
Convert JSON formatted string to JSON.

Parameters

- **json_string** (*str*) – JSON formatted string.

- ****kwargs** – Any keyword argument from the json.loads method.

Returns JSON formatted dictionary.

Return type dict

helios.utilities.json_utils.**write_json**(*json_dict*, *file_name*, ***kwargs*)
Write JSON dictionary to file.

Parameters

- **json_dict** (*dict*) – JSON formatted dictionary.
- **file_name** (*str*) – Output file name.
- ****kwargs** – Any keyword argument from the json.dump method.

Returns None

12.2 parsing_utils

Helper functions for paths and URLs.

helios.utilities.parsing_utils.**parse_camera**(*data*)
Parse camera name from a URL or image name.

Parameters **data** (*str*) – Image URL or name.

Returns Camera name.

Return type str

helios.utilities.parsing_utils.**parse_image_name**(*url*)
Parse image name from a URL.

Parameters **url** (*str*) – Image URL.

Returns Image name.

Return type str

helios.utilities.parsing_utils.**parse_time**(*data*)
Parse time from a URL or image name.

Parameters **data** (*str*) – Image URL or name.

Returns The parsed time as a datetime object.

Return type datetime.datetime

helios.utilities.parsing_utils.**parse_url**(*url*)
Parse a URL into its components.

Parameters **url** (*str*) – Image URL.

Returns Parsed URL.

Return type urllib.parse.ParseResult

12.3 data_utils

Utilities for working with SDK results.

helios.utilities.data_utils.**concatenate_feature_collections**(**args*)

Concatenates FeatureCollections.

```
import helios
from helios.utilities.data_utils import concatenate_feature_collections

cameras = helios.client('cameras')
results1, failed = cameras.index(state='new york')
results2, failed = cameras.index(state='maryland')
combined = concatenate_feature_collections((results1, results2))
```

Parameters **args** – (fc0, fc1, fc2, . . .) FeatureCollections to be combined. All FeatureCollections must be of the same type.

Returns FeatureCollection of the same API type as the input.

Return type FeatureCollection

CHAPTER 13

Indices and tables

- genindex
- modindex

Python Module Index

h

Index

E

F

G

H

I

J

M

N

O

P

R

S

T

U

V

W