helios Documentation

Release 3.0.1

Michael Bayer

Sep 27, 2018

User Guide

Use the Helios APIs in Python.

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

Using the Core APIs

3.1 Creating Instances

Instances of the core APIs are easy to create.

```
import helios
alerts = helios.Alerts()
cameras = helios.Cameras()
observations = helios.Observations()
collections = helios.Collections()
```

Each instance will internally initialize a Helios *Session* and call *start_session*.

3.2 Examples

3.2.1 Find alerts

```
import helios
alerts = helios.Alerts()

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

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

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

3.2.2 Find camera times and download images

```
import helios
import numpy as np

cameras = helios.Cameras()

# Find cameras in Maryland.
md_cam_results = 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 = cameras.show_image(cam_id,
                                        image_times,
                                        out_dir='/temp/data',
                                        return_image_data=True)

# Get a list of image data. (return_image_dat was True)
img_data = show_image_results.image_data
```

- `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*.

3.2.3 Find observations and work with collections

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

observations = helios.Observations()
collections = helios.Collections()

# Find Observations
index_results = 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_1 = index_results.id

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

# Add Observations to collection.
payload = [{'observation_id': x} for x in ids]
add_result = collections.add_image(new_id, payload)
```

(continues on next page)

(continued from previous page)

```
# 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`.
- `add_result` is an instance of *RecordCollection*.

CHAPTER 4

Session Instances

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

Once a session has been created, the token will be written to a *.helios_token* file in your home directory. This token will be reused until it becomes invalid.

4.1 Creating a Session

If authentication is stored on your machine starting a session is simple. Create a *Session* instance without any inputs. The authentication information stored on your machine will automatically be applied.

```
import helios
sess = helios.Session()
```

This will automatically make a call to the *start_session* method to fetch the token.

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

4.1.1 Token Expiration

Restarting Python if your token expires while the SDK is in use is not necessary. Call *start_session* to perform the token verification process. This will acquire a new token if it has expired.

After the a token has been re-acquired you will need to create new core API instances using the session.

4.2 Reusing a Session

Creating a *Session* instance allows you to use a single instance across all Core APIs. This avoids multiple token verifications with the initialization of every Core API instance.

```
import helios
sess = helios.Session()
alerts = helios.Alerts(session=sess)
cameras = helios.Cameras(session=sess)
```

In the above code `sess` is started once and used across *`Alerts`* and *`Cameras`*.

4.3 Using a Custom `env`

When creating a *`Session`* instance an optional input variable, `env`, can be used for dynamic credential usage.

This optional input must consist of a dictionary containing all necessary information for authentication.

```
custom_env = {'helios_client_id': 'your ID key',
              'helios_client_secret': 'your secret key',
              'helios_api_url': 'optional API URL override'}
sess = helios.Session(env=custom_env)
sess.start_session()
```

CHAPTER 5

License

The HeliosSDK is released under terms of MIT license.

CHAPTER 6

Core APIs

6.1 Alerts

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 *AlertsFeatureCollection*

show(*alert_ids*)

Get attributes for alerts.

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

Returns *AlertsFeatureCollection*

class helios.alerts_api.**AlertsFeature**(*feature*)
Individual Alert GeoJSON feature.

area_description
str – 'areaDesc' value for the feature.

bbox
list of floats – 'bbox' value for the feature.

category
str – 'category' value for the feature.

certainty
str – 'certainty' value for the feature.

country
str – 'country' value for the feature.

description
str – 'description' value for the feature.

effective
str – 'effective' value for the feature.

event
str – 'event' value for the feature.

expires
str – 'expires' value for the feature.

headline
str – 'headline' value for the feature.

id
str – 'id' value for the feature.

json
dict – Raw 'json' for the feature.

origin
str – 'origin' value for the feature.

severity
str – 'severity' value for the feature.

states
list of strs – 'states' value for the feature.

status
str – 'status' value for the feature.

urgency
str – 'urgency' value for the feature.

class helios.alerts_api.**AlertsFeatureCollection**(*features*, *records=None*)
Collection of GeoJSON features obtained via the Alerts API.

Convenience properties are available to extract values from every feature.

features
list of *AlertsFeature* – All features returned from a query.

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.

6.2 Cameras

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 *CamerasFeatureCollection*

show (*camera_ids*)

Get attributes for cameras.

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

Returns *CamerasFeatureCollection*

show_image (*camera_id*, *times*, *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

- **camera_id** (*str*) – Camera ID.
- **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.
- **out_dir** (*optional, str*) – Directory to write images to. Defaults to None.
- **return_image_data** (*optional, bool*) – If True images will be returned as numpy.ndarrays. Defaults to False.

Returns *ImageCollection*

class helios.cameras_api.**CamerasFeature** (*feature*)

Individual Camera GeoJSON feature.

city

str – 'city' value for the feature.

country
str – 'country' value for the feature.

description
str – 'description' value for the feature.

direction
str – 'direction' value for the feature.

id
str – 'id' value for the feature.

json
dict – Raw 'json' for the feature.

region
str – 'region' value for the feature.

state
str – 'state' value for the feature.

video
bool – 'video' value for the feature.

class helios.cameras_api.**CamerasFeatureCollection**(*features*, *records=None*)
Collection of GeoJSON features obtained via the Cameras API.

Convenience properties are available to extract values from every feature.

features
list of *CamerasFeature* – All features returned from a query.

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.

video
'video' values for every feature.

6.3 Collections

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(*collection_id*, *assets*)

Add images to a collection from Helios assets.

assets dictionary templates:

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

Usage example:

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

Parameters

- **collection_id** (*str*) – Collection ID.
- **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'}]

Returns *RecordCollection*

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 *CollectionsFeatureCollection*

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

Parameters

- **collection_id** (*str*) – Collection ID to remove images from.

- **names** (*str or list of strs*) – List of image names to be removed.

Returns *RecordCollection*

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 *CollectionsFeature*

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

Parameters

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

Returns *ImageCollection*

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.

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

bucket
str – 'bucket' value for the result.

created_at
str – 'city' value for the result.

description
str – 'created_at' value for the result.

id
str – '_id' value for the result.

images
list of strs – 'images' value for the result.

json
dict – Raw JSON result.

name
str – 'name' value for the result.

tags
list of strs – 'tags' value for the result.

updated_at
str – 'updated_at' value for the result.

user_id
str – 'user_id' value for the result.

class helios.collections_api.**CollectionsFeatureCollection**(*features*, *records=None*)

Collection of features obtained via the Collections API.

Convenience properties are available to extract values from every feature.

features
list of *CollectionsFeature* – All features returned from a query.

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.

6.4 Observations

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.

Usage example:

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

Usage example for transitions:

```
import helios
obs = helios.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 = obs.index(sensors=sensors_query)
```

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

Returns *ObservationsFeatureCollection*

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 returned as numpy.ndarrays. Defaults to False.

Returns *ImageCollection*

show(*observation_ids*)

Get attributes for observations.

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

Returns *ObservationsFeatureCollection*

class helios.observations_api.**ObservationsFeature**(*feature*)

Individual Observation GeoJSON feature.

city

str – 'city' value for the feature.

country
str – 'country' value for the feature.

description
str – 'description' value for the feature.

id
str – 'id' value for the feature.

json
dict – Raw JSON feature.

prev_id
str – 'prev_id' value for the feature.

region
str – 'region' value for the feature.

sensors
dict – 'sensors' value for the feature.

state
str – 'state' value for the feature.

time
str – 'time' value for the feature.

class helios.observations_api.**ObservationsFeatureCollection**(*features*, *records=None*)

Collection of GeoJSON features obtained via the Observations API.

Convenience properties are available to extract values from every feature.

features
list of `ObservationsFeature` – All features returned from a query.

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 7

Session

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

class helios.core.session.**Session**(*env=None*)

Manages API tokens for authentication.

Authentication credentials can be specified using the env input parameter, environment variables, or a credentials.json file in your ~/.helios directory. See the official documentation for more authentication information.

Required keys:

- helios_client_id: Client ID from API key pair.
- helios_client_secret: Client Secret ID from API key pair.

Optional keys:

- helios_api_url: Optional, URL for API endpoint.

A session can be established and reused for multiple core API instances.

```
import helios
sess = helios.Session()
alerts = helios.Alerts(session=sess)
cameras = helios.Cameras(session=sess)
```

If a session is not specified before hand, one will be initialized automatically. This is less efficient because each core API instance will try to initialize a session.

```
import helios
alerts = helios.Alerts()
cameras = helios.Cameras()
```

start_session()

Begins Helios session.

This will establish and verify a token for the session. If a token file exists the token will be read and verified. If the token file doesn't exist or the token has expired then a new token will be acquired.

verify_token()
Verifies the token.

If the token is bad or if the expiration time is less than the threshold False will be returned.

Returns True if current token is valid, False otherwise.

Return type bool

CHAPTER 8

Structure

Base data structures for the SDK.

class helios.core.structure.**ImageCollection**(*image_data*, *records*)

Stores all image content and associated metadata.

image_data

list of ndarray – All image data.

image_names

Names of all images.

output_files

Full paths to all saved images.

class helios.core.structure.**ImageRecord**(*message=None*, *query=None*, *content=None*, *error=None*, *name=None*, *output_file=None*)

Record class for images.

Parameters

- **message** (*tuple*) – Original message. This will be a namedtuple containing all the inputs for an individual call within a batch job.
- **query** (*str*) – API query.
- **content** (*numpy.ndarray*) – Image as a Numpy ndarray.
- **error** (*exception*) – Exception that occurred, if any.
- **name** (*str*) – Name of image.
- **output_file** (*str*) – Full path to image file that was written.

ok

Check if failure occurred.

Returns False if error occurred, and True otherwise.

Return type bool

class helios.core.structure.**Record**(*message=None*, *query=None*, *content=None*, *error=None*)

Individual query record.

Parameters

- **message** (*tuple*) – Original message. This will be a namedtuple containing all the inputs for an individual call within a batch job.
- **query** (*str*) – API query.
- **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

class helios.core.structure.**RecordCollection**(*records=None*)

Class for handling query records.

records

list of *Record* – Raw record data for debugging purposes.

failed

Records for queries that failed.

succeeded

Records for queries that succeeded.

CHAPTER 9

Utilities

9.1 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

9.2 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

9.3 data_utils

Utilities for working with SDK results.

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

Concatenates FeatureCollections.

```
import helios
from helios.utilities.data_utils import concatenate_feature_collections
cams_inst = helios.Cameras()
results1 = cams_inst.index(state='new york')
results2 = cams_inst.index(state='maryland')
combined = concatenate_feature_collections((results1, results2))
```

Parameters **fc_tuple** (*tuple*) – (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 10

Indices and tables

- genindex
- modindex

Python Module Index

h

Index

E

F

H

I

J

M

N

O