podaacpy Documentation

Release 2.4.0

Lewis John McGibbney

Aug 24, 2019

Contents

Contents:

CHAPTER 1

Introduction to podaacpy

1.1 Introduction

podaacpy is a python utility library for interacting with NASA JPL's PO.DAAC

1.2 What is PO.DAAC?

The Physical Oceanography Distributed Active Archive Center (PO.DAAC) is an element of the Earth Observing System Data and Information System (EOSDIS). The EOSDIS provides science data to a wide community of users for NASA's Science Mission Directorate.

1.3 What does podaacpy offer?

The library provides a Python toolkit for interacting with all of PO.DAACs API's, namely

- PO.DAAC Web Services: services include
- Dataset Metadata - retrieves the metadata of a dataset
- Dataset Search - searches PO.DAAC's dataset catalog, over Level 2, Level 3, and Level 4 datasets
- Dataset Variables - provides list of dataset variables for the dataset
- Granule Metadata - retrieves the metadata of a granule
- Granule Search - does granule searching on PO.DAAC level 2 swath datasets (individual orbits of a satellite), and level 3 & 4 gridded datasets (time averaged to span the globe)
- Granule Preview - the PODAAC preview Image service retrieves pre-generated preview images for selected granules
- Granule Subset - Subset Granule service allows users to submit subset jobs

- Subset Status - Subset Granule Status service allows users to check the status of submitted subset job
- Metadata Compliance Checker: an online tool and web service designed to check and validate the contents of netCDF and HDF granules for the Climate and Forecast (CF) and Attribute Convention for Dataset Discovery (ACDD) metadata conventions.
- PO.DAAC Drive: an HTTP based data access service. PO.DAAC Drive replicates much of the functionality of FTP while addressing many of its issues.

Additionally, Podaacpy provides the following ocean-related data services * NASA OceanColor Web:

- File Search - locate publically available files within the NASA Ocean Data Processing System (ODPS)
- Bulk data downloads via HTTP - mimic FTP bulk data downloads using the HTTP-based data distribution server.

CHAPTER 2

Quickstart

2.1 Purpose

The following document explains how to quickly get up and running with podaac. It explains how to execute the key commands and explains (at a high level) what those commands are doing e.g. what input and output we can expect. More detail on expressive use of the various API's including function level API documentation can be found in subsequent pages of this documentation guide.

2.2 Working with Data Webservices

2.2.1 Importing podaac

This is very simple...

```
# import the podaac package
import podaac.podaac as podaac

# then create an instance of the Podaac class
p = podaac.Podaac()
```

More on using Podaac functions later... first lets look at some convenience functionality.

2.2.2 Convenience Functions

There are a number of convenience functions which aid various types of search. These help decypher the rather cryptic dataset id's, dataset short names, etc. present within PO.DAAC. These functions accept no parameters. They do however account for the fact that availability of certain datasets within PO.DAAC is not constant. Additionally some services are only available for certain datasets. The functions encapsulate those underlying variables and always return current, available results which can be interpreted and used within the other functions in this file.

First lets define the relevant import

```
# import the podaac_utils package
import podaac.podaac_utils as utils

# then create an instance of the PodaacUtils class
u = utils.PodaacUtils()
```

The convenience functions are;

```
result = u.list_available_granule_search_level2_dataset_ids()

result = u.list_available_granule_search_level2_dataset_short_names()

result = u.list_available_granule_search_dataset_ids()

result = u.list_available_granule_search_dataset_short_names()

result = u.list_available_extract_granule_dataset_ids():

result = u.list_available_extract_granule_datasetShortNames():
```

For all of the above, the variable **result** now contains a Python List containing comma-separated values which can be processed appropriately. For more information on these functions, see *podaacpy utilities API*

2.2.3 Retrieving Dataset Metadata

Dataset Metadata - retrieves the metadata of a dataset. In the following code snippet lets retrieve dataset metadata for GHRSST Level 2P Atlantic Regional Skin Sea Surface Temperature from the Spinning Enhanced Visible and InfraRed Imager (SEVIRI) on the Meteosat Second Generation (MSG-2) satellite e.g. dataset id **PODAAC-GHMG2-2PO01**

```
result = p.dataset_metadata(dataset_id='PODAAC-GHMG2-2PO01')
```

The variable **result** now contains an XML response which can be processed appropriately. For more information on this function, see *podaacpy webservices API*

2.2.4 Retrieving Granule Metadata

Granule Metadata - retrieves the metadata of a granule. In the following code snippet we retrieve granule metadata for the above dataset e.g. granule_name **20120912-MSG02-OSDPD-L2P-MSG02_0200Z-v01.nc**

```
result = p.granule_metadata(dataset_id='PODAAC-GHMG2-2PO01', granule_name='20120912-
↪MSG02-OSDPD-L2P-MSG02_0200Z-v01.nc')
```

The variable **result** now contains an XML response which can be processed appropriately. For more information on this function, see *podaacpy webservices API*

Additionally, we can search metadata for list of granules archived within the last 24 hours in Datacasting format.

```
result = p.last24hours_datacasting_granule_md(dataset_id='PODAAC-GHMG2-2PO01')
```

The variable **result** now contains an XML response containing a list of data granules which can be processed appropriately. For more information on this function, see *podaacpy webservices API*

2.2.5 Retrieving Dataset Variables

Dataset Variables - provides a list of variable for the datset. In the following code snippet we retrieve the dataset variables for the dataset id **PODAAC-ASOP2-25X01**

```
result = p..dataset_variables(dataset_id='PODAAC-ASOP2-25X01')
```

The variable **result** now contains a dictionary of variables of the respective dataset. For more information on this function, see *podaacpy webservices API*

2.2.6 Searching for Datasets

Search Dataset - searches PO.DAAC's dataset catalog, over Level 2, Level 3, and Level 4 datasets. In the following code snippet we will search using a keyword e.g. **modis**

```
result = p.dataset_search(keyword='modis')
```

The variable **result** now contains an XML response containing a list of datasets which can be processed appropriately. For more information on this function, see *podaacpy webservices API*

2.2.7 Searching for Granules

Search Granule - does granule searching on PO.DAAC level 2 swath datasets (individual orbits of a satellite), and level 3 & 4 gridded datasets (time averaged to span the globe). In the following code snippet we will search for granules within a specific dataset e.g. **PODAAC-ASOP2-25X01**

```
result = p.granule_search(dataset_id='PODAAC-ASOP2-25X01', bbox='0,0,180,90',start_
↪time='2013-01-01T01:30:00Z',end_time='2014-01-01T00:00:00Z',start_index='1'))
```

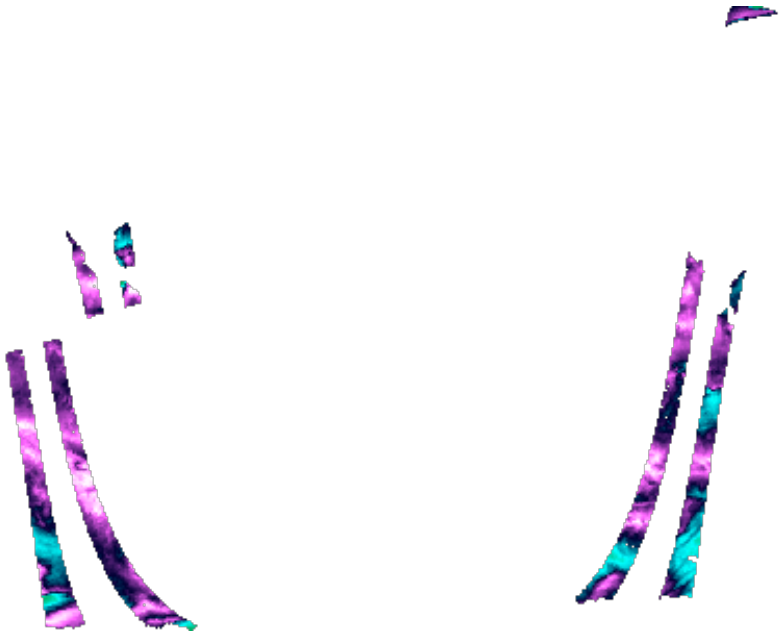
The variable **result** now contains an XML response containing a list of granules for the given dataset which can be processed appropriately. For more information on this function, see *podaacpy webservices API*

2.2.8 Retrieve granule images

Granule Preview - renders granules in the PO.DAAC's catalog to images such as jpeg and/or png. In the following code snippet we display a request using the dataset id **PODAAC-ASOP2-25X01** and image variable of the dataset **wind_speed**

```
result = p.granule_preview(dataset_id='PODAAC-ASOP2-25X01', image_variable='wind_speed
↪')
```

The above request downloads us a nice image shown below



For more information on this function, see *podaacpy webservices API*

2.2.9 Subsetting Granules

Granule Subset - the Granule Subset web service sets up a granule subsetting job using HTTP POST request. Upon a successful request, a token is returned which can be used to check the status of the subsetting job. In the following code snippet we will subset a granule using an input.json file which contains

```
query={
 "email":"abc@abcd.com",
 "query": [
     {
         "compact":false,
         "datasetId":"PODAAC-ASOP2-25X01",
         "bbox":"-180,-90,0,90",
         "variables" : ["lat" , "lon","time","wind_speed" ],
         "granuleIds": ["ascat_20140520_005700_metopa_39344_eps_o_250_2300_ovw.l2.nc",
↪"ascat_20140411_175700_metopa_38800_eps_o_250_2300_ovw.l2.nc"]
     }
  ]
}

result = p.granule_subset(input_file_path='/path/to/input.json')
```

The variable **result** contains a token on successful request reception. This can be further used to check the status of the request. For more information on this function, see *podaacpy webservices API*

2.2.10 Subset Status

Subset Status - the subset status checks the status on the existing job. In the following code snippet we check the status using the token received from PO.DAAC when we submitted a job for subsetting

```
result = p.granule_preview(dataset_id='PODAAC-ASOP2-25X01', image_variable='wind_speed
↪')
```

(continues on next page)

(continued from previous page)

The variable **result** contains the status of the subset request. For more information on this function, see *podaacpy webservices API*

2.2.11 Extract level4 granule

Right now the *Extract Granule <https://podaac.jpl.nasa.gov/ws/extract/granule/index.html>* supports only level 2 granules. Extract l4 granule is an add-on over extract granule to extract level 4 gridded datasets from the PODAAC data source. In the following code snippet we extract a level4 granule with Dataset ID = **PODAAC-CCF30-01XXX**, short_name of **CCMP_MEASURES_ATLAS_L4_OW_L3_0_WIND_VECTORS_FLK** and provide a path to the directory you want to have it saved as **netcdf**

```
result = p.extract_l4_granule(dataset_id='PODAAC-CCF30-01XXX', short_name='CCMP_
↪MEASURES_ATLAS_L4_OW_L3_0_WIND_VECTORS_FLK', path='path/to/the/destination/directory
↪')
```

The above request downloads the relevant .netcdf file. For more information on this function, see *podaacpy webservices API*

2.3 Working with Metadata Compliance Webservices (mcc)

2.3.1 Importing mcc

This is very simple...

```
# import the mcc package
import podaac.mcc as mcc

# then create an instance of the MCC class
m = mcc.MCC()
```

2.3.2 Compliance Check a Local File

The following example displays how to use the MCC to check and validate the contents of a local granule (netCDF or HDF) given the relevant input parameters.

```
result = m.check_local_file(acdd_version='1.3', gds2_parameters='L4', file_upload=
↪'someLocalFile.nc', response='json')
```

The result variable contains a JSON encoded report response which can be used for compliance checking activities. For more information on this function, see *podaacpy metadata compliance checker (mcc) API*

2.3.3 Compliance Check a Remote File

The following example displays how to use the MCC to check and validate the contents of a remote granule (netCDF or HDF) given the relevant input parameters.

```
result = m.check_remote_file(checkers='CF', url_upload='http://test.opendap.org/
↪opendap/data/ncml/agg/dated/CG2006158_120000h_usfc.nc', response='json')
```

The result variable contains a JSON encoded report response which can be used for compliance checking activities. For more information on this function, see *podaacpy metadata compliance checker (mcc) API*

2.4 Conclusion

That concludes the quick start. Hopefully this has been helpful in providing an overview of the main podaacpy features. If you have any issues with this document then please register them at the issue tracker. Please use labels to classify your issue.

CHAPTER 3

podaacpy utilities API

class podaac_utils.**PodaacUtils**

list_all_available_extract_granule_dataset_ids()
Convenience function which returns an up-to-date list of all available granule dataset id's which can be used in the granule extraction service.

Returns a comma-seperated list of granule dataset id's.

list_all_available_extract_granule_dataset_short_names()
Convenience function which returns an up-to-date list of all available granule dataset short names which can be used in the granule extraction service.

Returns a comma-seperated list of granule dataset short names.

list_all_available_granule_search_dataset_ids()
Convenience function which returns an up-to-date list of available all granule dataset id's.

Returns a comma-seperated list of granule dataset id's

list_all_available_granule_search_dataset_short_names()
Convenience function which returns an up-to-date list of available granule dataset short names.

Returns a comma-seperated list of granule dataset short names.

list_available_granule_search_level2_dataset_ids()
Convenience function which returns an up-to-date list of available level2 granule dataset id's.

Returns a comma-seperated list of granule dataset id's

list_available_granule_search_level2_dataset_short_names()
Convenience function which returns an up-to-date list of available level2 granule dataset short names.

Returns a comma-seperated list of granule dataset short names.

list_level4_dataset_ids()
Convenience function which returns an up-to-date list of level4 dataset id's.

Returns a comma-seperated list of level4 dataset id's

list_level4_dataset_short_names()
Convenience function which returns an up-to-date list of level4 dataset short names.

Returns a comma-seperated list of level4 dataset short names.

mine_granules_from_granule_search(*granule_search_response=''*)
Convenience function which extracts the granule names for a given granule search obtained using podaac.granule_search(). The response of this function is an array of strings denoting the granule names for the granule search.

Parameters **granule_search_response** – the output response of a podaac.granule_search()

Returns prints an array of granule names.

static mine_opendap_urls_from_granule_search(*granule_search_response=''*)

Convenience function which extracts the PO.DAAC OPeNDAP URLs from a given granule search obtained using podaac.granule_search(). The response of this function is an array of strings denoting the PO.DAAC OPeNDAP URLs to the granules.

Parameters **granule_search_response** – the output response of a podaac.granule_search()

Returns prints an array of PO.DAAC OPeNDAP URLs.

CHAPTER 4

podaacpy webservices API

class podaac.**Podaac**

dataset_metadata(*dataset_id=''*, *short_name=''*, *_format='iso'*)

Dataset metadata service retrieves the metadata of a dataset on PO.DAACs dataset catalog using the following parameters: dataset_id, short_name, and format.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for this metadata service.
- **short_name** (string) – dataset short_name. dataset_id or short_name is required for this metadata service.
- **_format** (string) – metadata format. Default format is iso.

Returns an xml response based on the requested 'format'. Options are 'iso' and 'gcmd'.

dataset_search(*keyword=''*, *start_time=''*, *end_time=''*, *start_index=''*, *dataset_id=''*, *short_name=''*, *instrument=''*, *satellite=''*, *file_format=''*, *status=''*, *process_level=''*, *sort_by=''*, *bbox=''*, *items_per_page='50'*, *_format='atom'*, *full='False'*)

Dataset Search service searches PO.DAAC's dataset catalog, over Level 2, Level 3, and Level 4 datasets, using the following parameters: dataset_id, short_name, start_time, end_time, bbox, and others.

Parameters

- **keyword** (string) – keyword specifies search text to search for datasets. Example: 'modis'
- **start_time** (time) – start time in the format of YYYY-MM-DDTHH:mm:ssZ. 'Z' is the time-offset, where 'Z' signifies UTC or an actual offset can be used. Example: 2012-01-22T01:21:21Z
- **end_time** (time) – stop time in the format of YYYY-MM-DDTHH:mm:ssZ. 'Z' is the time-offset, where 'Z' signifies UTC or an actual offset can be used. Example: 2012-01-22T01:21:21Z

- **start_index** (int) – start index of entries found for search. Example: 1
- **items_per_page** (int) – number of results per page for opensearch result. If format is not specified, format is set to 50. The value range is from 0 to 400
- **dataset_id** (string) – dataset persistent ID. Example: PODAAC-MODSA-T8D9N
- **short_name** (string) – dataset short_name. Example: MODIS_AQUA_L3_SST_THERMAL_8DAY_9KM_NIGHTTIME
- **instrument** (string) – dataset instrument. Example: MODIS
- **satellite** (string) – dataset satellite. Example: AQUA
- **file_format** (string) – dataset data format. Possible values: HDF, NetCDF
- **status** (string) – dataset status. Possible values: OPEN, PREVIEW, SIMULATED, RETIRED
- **processLevel** (string) – dataset processing level. Possible values: 1B, 2, 2P, 3, 4
- **_format** (string) – response format. If format is not specified, format is set to atom. Possible values: atom, html.
- **sort_by** (string) – determines ordering of response. If sort_by is not specified, sort order is by score (most relevant dataset first). Possible values: timeAsc, timeDesc, popularityAsc, popularityDesc.
- **bbox** (string) – bounding box for spatial search. format should look like "bbox=0.0,-45.0,180.0,40.0" which is in order of west, south, east, north. Longitude values needs to be in range of [-180.0,180.0]. Example: 0.0,-45.0,180.0,40.0
- **full** (string) – "true" to return response with complete PO.DAAC metadata per entry. If full is not specified, full is set to false. Possible values: true, false

Returns the specified response format. If format is not specified, format is set to atom. Possible values: atom, html

dataset_variables(*dataset_id*)

Given a PO.DAAC dataset identifier this function will return list of dataset variables.

Parameters **dataset_id** (string) – dataset persistent ID. dataset_id is required for this metadata service.

Returns a list of dataset variables for the dataset.

extract_l4_granule(*dataset_id=''*, *path=''*)

This is an additional function that we have provided apart from the available webservices. The extract_l4_granule helps retrieve the level 4 datasets from OPeNDAP server directly, accompanied by the search granule for retrieving granule name related to the specific dataset_id.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for a granule search. Example: PODAAC-ASOP2-25X01
- **path** – Destination directory into which the granule needs to be downloaded.

Returns string representation of granule name.

granule_metadata(*dataset_id=''*, *short_name=''*, *granule_name=''*, *_format='iso'*)

Granule metadata service retrieves the metadata of a granule on PO.DAACs catalog in ISO-19115.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for this metadata service.
- **short_name** (string) – dataset short_name. dataset_id or short_name is required for this metadata service.
- **granule_name** (string) – granule name. granule name is required for this metadata service.
- **_format** (string) – metadata format. Default format is iso.

Returns an xml response based on the requested 'format'.

granule_preview(*dataset_id=''*, *image_variable=''*, *path=''*)

The PODAAC Image service renders granules in the PO.DAACs catalog to images such as jpeg and/or png. This image service also utilizes OGC WMS protocol. (http://www.opengeospatial.org/standards/wms). If the granule does not have any data in the given selected bounding box, HTTP 500 will be thrown since there is no data to be imaged. Granule Search service can be used to find level 2 swath data. However, the level 2 spatial search uses coverage footprint polygons generated for each granule, and this footprint can contain no data or gaps. If the selected bounding box resides on no data or gaps, HTTP 500 will be thrown. There are three request methods in this service. They are GetCapabilities, GetLegendGraphic, and GetMap.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for a granule search. Example: PODAAC-ASOP2-25X01 string
- **image_variable** – variables of the granule which have 'Preview Images'. Image variables can be found from Dataset Variable service. Use "id" from "imgVariable" element. :type image_variable: string
- **path** – Destination directory into which the granule needs to be downloaded.

Returns a png image file.

granule_search(*dataset_id=''*, *start_time=''*, *end_time=''*, *bbox=''*, *start_index=''*, *sort_by='timeAsc'*, *items_per_page='50'*, *_format='atom'*)

Search Granule does granule searching on PO.DAAC level 2 swath datasets (individual orbits of a satellite), and level 3 & 4 gridded datasets (time averaged to span the globe). Coverage footpritnt polygons are used to enable spatial search on level 2 swath dataset. Currently, our footprints can contain no data and also gaps in the swath data. Spatial search on level 2 data can return granules where actual data does not intersect the selected bounding box but its footprint intersects the selected bounding box. The following parameters are supported: dataset_id, short_name, start_time, end_time, bbox, and others.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for a granule search. Example: PODAAC-ASOP2-25X01
- **start_time** (time) – start time in the format of YYYY-MM-DDTHH:mm:ssZ. 'Z' is the time-offset, where 'Z' signifies UTC or an actual offset can be used. Example: 2013-01-01T01:30:00Z
- **end_time** (time) – stop time in the format of YYYY-MM-DDTHH:mm:ssZ. 'Z' is the time-offset, where 'Z' signifies UTC or an actual offset can be used. Example: 2014-01-01T00:00:00Z
- **bbox** (string) – bounding box for spatial search. format should look like "bbox=0,0,180,90" which is in order of west, south, east, north. Longitude values needs to be in range of [-180, 180]. Latitude values needs to be in range of [-90, 90]. For level 2 datasets, spatial search is available for a subset. Call the list_available_Level2_dataset_ids

and list_available_level2_datasetShortNames functions to see the subset. BBox example: 0,0,180,90

- **start_index** (int) – start index of entries found for search. Example: 1
- **items_per_page** (int) – number of results per page for opensearch result. If format is not specified, format is set to 50. The value range is from 0 to 400
- **sort_by** (string) – determines ordering of response. Possible values: timeAsc, timeDesc.
- **_format** (string) – response format. If format is not specified, format is set to atom. Possible values: atom, html.

Returns an xml response based on the requested 'format'. Options are 'atom' and 'html'.

granule_subset(*input_file_path*, *path=''*)

Subset Granule service allows users to Submit subset jobs. Use of this service should be preceded by a Granule Search in order to identify and generate a list of granules to be subsetted. NOTE : At present PODAAC's granule subsetting service is only restricted to Level2 granules.

Parameters

- **input_file_path** (string) – path to a json file which contains the the request that you want to send to PO.DAAC
- **path** (string) – path to a directory where you want the subsetted dataset to be stored.

Returns a string token which can be used to determine the status of a subset task.

last24hours_datacasting_granule_md(*dataset_id=''*, *short_name=''*, *_format='datacasting'*, *items_per_page=50*)

Granule metadata service retrieves metadata for a list of granules archived within the last 24 hours in Datacasting format.

Parameters

- **dataset_id** (string) – dataset persistent ID. dataset_id or short_name is required for this metadata service.
- **short_name** (string) – dataset short_name. dataset_id or short_name is required for this metadata service.
- **_format** (string) – metadata format. Must set to 'datacasting'.
- **items_per_page** (int) – number of results per page. Default value is 50. The value range is from 0 to 5000.

Returns an xml response based on the requested 'format'. Options are 'iso' and 'gcmd'.

subset_status(*token=''*)

Subset Granule Status service allows users to check the status of submitted subset job. The possible status that it returns include the following..

```
* "submitted" : returned on successful submission of the request.
* "error" : returned when there is error in the JSON POST request.
* "unknown" : returned when the datasetId you sent is not valid.
* "done" : returned when subsetting job you submitted is done.
```

Parameters **token** (string) – string token that is returned by PO.DAAC whilst submitting a subset request.

Returns the a status string of the subset request.

CHAPTER 5

podaacpy metadata compliance checker (mcc) API

class mcc.**MCC**

check_local_file(*acdd_version*, *gds2_parameters*, *file_upload*, *response='json'*)

POST a local file to the metadata compliance checker endpoint at https://podaac-uat.jpl.nasa.gov/mcc/check

Parameters

- **acdd_version** (string) – Must be present and and set to either 1.1 or 1.3. 'acdd' tag must also be present and must be set to 'on'.
- **gds2_parameters** (string) – Must be present and set to either 'L2P', 'L3', 'L4'.
- **file_upload** (string) – A valid location of a netCDF file; maximum 5.00 GB.
- **response** (string) – Specify 'html', 'json', or 'pdf' result output. Default is 'json'.

Returns one of 'html', 'json', or 'pdf'.

Raises **ValueError** – If no dataset can be found for the supplied url_upload or if the requested dataset is a multi-file dataset.

check_remote_file(*checkers*, *url_upload*, *response='json'*)

GET a remote file e.g. from an OPeNDAP URL and compliance check it against the endpoint at https://podaac-uat.jpl.nasa.gov/mcc/check.

Parameters

- **checkers** (string) – Must specify at least one test. Multiple tests are delimited by commas. Possible values include 'ACDD-x.x', 'CF' and 'GDS2' which also requires 'GDS2-parameters:levelAvailable'. Available levels are 'L2P', 'L3', and 'L4'.
- **url_upload** (string) – A valid url to a netCDF file; maximum 5.00 GB
- **response** (string) – (Optional) Specify 'html', 'json', or 'pdf' result output.

Returns one of 'html', 'json', or 'pdf'.

Raises `ValueError` – If no dataset can be found for the supplied url_upload or if the requested dataset is a multi-file dataset.

CHAPTER 6

podaacpy Level 2 Subsetting (L2SS) API

class l2ss.**L2SS**

dataset_search(*dataset_id=''*, *variable=None*, *sensor=None*, *provider=None*, *start_time=''*, *end_time=''*, *start_index=''*, *items_per_page='50'*)

Dataset search service lists available datasets and returns them.

Parameters

- **dataset_id** (string) – Search dataset belong to given PODAAC Dataset persistent ID.
- **variable** (list) – Search for datasets with variable name. For multi-value input, this input is taken as a list. Example: ['Sea Surface Temperature', 'Surface Wind']
- **sensor** (list) – Search for datasets with sensor. For multi-value input, this input is taken as a list.
- **provider** (list) – Search for datasets with provider. For multi-value input, this input is taken as a list.
- **start_time** (string) – Lower time bound. If not specified, lower time bound of the dataset will be used. Example: '2011-12-31T23:59:59-06:00Z'
- **end_time** (string) – Upper time bound. If not specified, upper time bound of the dataset will be used. Example: 2019-12-31T23:59:59-06:00Z
- **items_per_page** (string) – number of results to return. Defaults to 50.
- **start_index** (string) – start index of result.

Returns a json response containing the datasets.

dataset_variables(*dataset_id*)

Dataset Variable retrieves dataset configuration information including variables.

Parameters **dataset_id** (string) – datasetId for the configuration information.

Returns a json response containing the dataset variables.

granule_download(*query_string*, *path=''*)
Granule Download service submits a job to subset and download. Upon a successful request, token will be returned which can be used to check status.

Parameters

- **query_string** (string) – data collection query json as a string.
- **path** (string) – path to a directory where you want the subsetted dataset to be stored.

Returns a zip file downloaded and extracted in the destination directory path provided.

granule_preview_image(*dataset_id*, *granule*, *year*, *day*, *variable*, *path=''*)
Granule Preview Image Service provides thumbnail image of selected variable for selected granule.

Parameters

- **dataset_id** (string) – Search granules belong to given PODAAC Dataset persistent ID.
- **granule** (string) – string granule name.
- **year** (string) – year in 4 digits. Example= '2014'
- **day** (string) – day of year in 3 digits. Example= '140'
- **variable_id** (string) – Variable id described in dataset variable service.

Returns returns thumbnail image of selected variable for selected granule.

granule_search(*dataset_id=''*, *bbox=''*, *start_time=''*, *end_time=''*, *name=''*, *sort=''*, *start_index=''*, *items_per_page='50'*)
Granule Search retrieves all base granule information (datasetId, start time, end time) matching the specified datasetId, date, and region. This approach may change if the data/querying turns out to be too expensive. Response is structured in a minimalistic way to cut down on the file size.

Parameters

- **dataset_id** (string) – Search granules belong to given PODAAC Dataset persistent ID.
- **bbox** (string) – Search granules with Bounding box Ex: '-180,-90,180,90'
- **start_time** (string) – Lower time bound. If not specified, lower time bound of the dataset will be used. Example: '2011-12-31T23:59:59-06:00Z'
- **end_time** (string) – Upper time bound. If not specified, upper time bound of the dataset will be used. Example: 2019-12-31T23:59:59-06:00Z

:param name : Search granules with exact name or name pattern using wildcard search Example: ascat* this matches name that starts with "ascat" :type name: string

Parameters

- **sort** (string) – Sort output. There are two strings delimited by space. The first string is the field name, and the second string is 'asc' or 'desc' Example: sort='Granule-Name asc'
- **items_per_page** (string) – number of results to return. Default to 50.
- **start_index** (string) – start index of result.

Returns a json response containing the dataset granules.

granules_availability(*dataset_id=''*, *start_time=''*, *end_time=''*, *gap=''*, *bbox=''*)
Granules Availability calculates granule counts per day or month from given date range.

Parameters

- **dataset_id** (string) – Search granules belong to given PODAAC Dataset persistent ID.
- **start_time** (string) – Lower time bound. If not specified, lower time bound of the dataset will be used. Example: '2011-12-31T23:59:59-06:00Z'
- **end_time** (string) – Upper time bound. If not specified, upper time bound of the dataset will be used. Example: 2019-12-31T23:59:59-06:00Z
- **gap** (string) – The size of each date range expressed as an interval to be added to the lower bound. Example: 'DAY', 'MONTHS'
- **bbox** (string) – Search granules with Bounding box Ex: '-180,-90,180,90'

Returns a json response containing the granule count and other relevant information.

image_palette(*palette_name*)

Image Palette service retrieves palette descriptor in json format

Parameters **palette_name** (string) – palette_name whose palette descriptor we want to retrieve.

Returns returns palette descriptor in json format.

subset_status(*token*)

Subset Status service check status on existing download job. The possible status that it returns include the following..

```
* "queued"
* "processing"
* "partial error"
* "done"
* "error"
```

Parameters **token** (string) – job token. job token is provided when submitting the job.

Returns the status of the subset request.

CHAPTER 7

NASA OceanColor API

class oceancolor.**OceanColor**

file_search(*sensor=''*, *sdate=''*, *edate=''*, *dtype=''*, *add_url='1'*, *results_as_file='1'*, *search=''*, *sub_id=''*, *std_only='1'*, *cksum=''*, *output_format='json'*)

File search service retrieves publically available files within the NASA Ocean Data Processing System.

Parameters

- **sensor** (string) – mission name. valid options include: aquarius, seawifs, aqua, terra, meris, octs, czcs, hico, viirs
- **sdate** (string) – start date for a search
- **edate** (string) – end date for a search
- **dtype** (string) – data type (i.e. level). valid options: L0, L1, L2, L3b (for binned data), L3m (for mapped data), MET (for ancillary data), misc (for sundry products)
- **add_url** (string) – include full url in search result (boolean, 1=yes, 0=no)
- **results_as_file** (string) – return results as a test file listing (boolean, 1=yes, 0=no, thus returns and HTML page)
- **search** (string) – text string search
- **sub_id** (string) – non-extracted subscription ID to search
- **std_only** (string) – restrict results to standard products (i.e. ignore extracts, regional processings, etc.; boolean)
- **cksum** (string) – return a checksum file for search results (boolean; sha1sums except for Aquarius soil moisture products which are md5sums; forces results_as_file; ignores addurl)
- **output_format** (string) – valid options are: 'json', 'txt' and 'html'

Returns by default a json response based on the requested 'output_format'. Options are 'json, 'txt' and 'html'.

get_file(*url=''*, *path=''*)

It is possible to mimic FTP bulk data downloads using the HTTP-based data distribution server at https://oceandata.sci.gsfc.nasa.gov.

Parameters

- **url** (`string`) – a single file name which can be obtained by calling #file_search() an example would be https://oceandata.sci.gsfc.nasa.gov/cgi/getfile/O1997001.L3b_DAY_CHL.nc
- **path** (`string`) – Destination directory into which the granule needs to be downloaded.

Returns a file object downloaded from the HTTP-based data distribution server at https://oceandata.sci.gsfc.nasa.gov.

CHAPTER 8

podaacpy drive API

class drive.**Drive**(*file*, *username*, *password*, *webdav_url='https://podaac-tools.jpl.nasa.gov/drive/files'*)

download_granules(*granule_collection=None*, *path=''*)

Granule download service downloads a granule collection from PO.DAAC Drive to the users' local machine at the given path. Note, as of https://github.com/nasa/podaacpy/issues/131 we now maintain the PO.DAAC Drive directory structure. This is to say, if the Drive URL was https://podaac-tools.jpl.nasa.gov/drive/files/allData/ghrsst/data/GDS2/L2P/AVHRR19_L/NAVO/v1/2019/088/20190329001403-NAVO-L2P_GHRSST-SST1m-AVHRR19_L-v02.0-fv01.0.nc then a directory structure would be created as follows allData/ghrsst/data/GDS2/L2P/AVHRR19_L/NAVO/v1/2019/088/20190329001403-NAVO-L2P_GHRSST-SST1m-AVHRR19_L-v02.0-fv01.0.nc

Parameters

- **granule_collection** (string) – a populated collection of PO.DAAC Drive Granule URLs. These can be obtained by using the drive.mine_drive_urls_from_granule_search() function which itself merely wraps a podaac.granule_search() request.
- **path** (string) – path to a directory where you want the data to be stored.

Returns a zip file downloaded and extracted in the destination directory path provided.

mine_drive_urls_from_granule_search(*granule_search_response=''*)

Convenience function which extracts the PO.DAAC Drive URLs from a given granule search obtained using podaac.granule_search(). The response of this function is an array of strings denoting the PO.DAAC Drive URLs to the granules.

Parameters **granule_search_response** – the output response of a podaac.granule_search() :type path: string

returns prints an array of PO.DAAC Drive URLs.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index