ai.cdas Documentation

Release

Alexey Isavnin

Dec 21, 2017

Contents

This library provides access to CDAS database from python in a simple and fluid way through CDAS REST api. It fetches the data either in CDF (Common Data Format) or ASCII format and returns it in the form of dictionaries of numpy arrays.

CHAPTER 1

Getting started

1.1 Dependencies

- Python 3
- numpy
- requests
- wget

1.2 Extra dependencies (at least one of the following)

- astropy
- CDF + spacepy

1.3 Installation

Starting from version 1.2 AI.CDAS officially supports only Python 3, so make sure that you have a working installation of it.

Assuming the above requirement is satisfied install the package with Python package manager:

```
$ pip install ai.cdas
```

1.4 Known issues

NASA CDAS REST API endpoint currently does not support IPv6 addressing. However, newer linux distros (for example, Ubuntu 16.04) are set up to prefer IPv6 addressing over IPv4 by default. This may result in unnecessary

delays in communication with the server. If you experience the issue it might be that this is the case with your system. Here is how it can be cured in Ubuntu 16.04:

```
$ sudoedit /etc/gai.conf
# Uncomment the line
# precedence ::ffff:0:0/96  100
```

Now you machine will attempt IPv4 prior to IPv6. For other distros refer to respective docs.

1.5 Examples

Example 1: Retrieving observatory groups and associated instruments which measure plasma and solar wind:

```
from ai import cdas
import json # for pretty output

obsGroupsAndInstruments = cdas.get_observatory_groups_and_instruments(
    'istp_public',
    instrumentType='Plasma and Solar Wind'
)
print(json.dumps(obsGroupsAndInstruments, indent=4))
```

Example 2: Getting STEREO-A datasets using regular expressions for dataset id and label:

```
from ai import cdas
import json # for pretty output

datasets = cdas.get_datasets(
    'istp_public',
    idPattern='STA.*',
    labelPattern='.*STEREO.*'
)
print(json.dumps(datasets, indent=4))
```

Example 3: Fetching a list of variables in one of STEREO datasets:

```
from ai import cdas
import json # for pretty output

variables = cdas.get_variables('istp_public', 'STA_L1_MAGB_RTN')
print(json.dumps(variables, indent=4))
```

Example 4: This snippet of code gets magnetic field data from STEREO-A spacecraft for one hour of 01.01.2010 and plots it (requires matplotlib):

```
from ai import cdas
from datetime import datetime
from matplotlib import pyplot as plt

data = cdas.get_data(
    'sp_phys',
    'STA_L1_MAG_RTN',
    datetime(2010, 1, 1),
    datetime(2010, 1, 1, 0, 59, 59),
    ['BFIELD']
)
```

```
plt.plot(data['EPOCH'], data['BTOTAL'])
plt.show()
```

Example 5: This snippet of code gets magnetic field data from STEREO-A spacecraft for one hour of 01.01.2010 and plots it (requires matplotlib). The data are downloaded in CDF format in this case. CDF format is binary and results in a much smaller filesize and hence faster downloads. In order for this to work you have to have NASA CDF library on your machine and spacepy installed afterwards:

```
from ai import cdas
from datetime import datetime
from matplotlib import pyplot as plt

data = cdas.get_data(
    'sp_phys',
    'STA_L1_MAG_RTN',
    datetime(2010, 1, 1),
    datetime(2010, 1, 1, 0, 59, 59),
    ['BFIELD'],
    cdf=True # download data in CDF format
)
# Note that variables identifiers are different than in the previous
# example. It often the case with CDAS data. You should check the
# variables names by printing out `data` dictionary.
plt.plot(data['Epoch'], data['BFIELD'][:, 3])
plt.show()
```

Example 6: This snippet of code gets magnetic field data from STEREO-A spacecraft for 01.01.2010 and saves it to cache directory. The next time the same data is requested it is taken from cache without downloading:

```
import os
from ai import cdas
from datetime import datetime

# For the sake of example we are using your current working
# directory as a cache directory
cache_dir = os.getcwd()
cdas.set_cache(True, cache_dir)
# this data is downloaded from CDAS
data = cdas.get_data(
    'sp_phys',
    'STA_L1_MAG_RTN',
    datetime(2010, 1, 1),
    datetime(2010, 1, 1, 0, 59, 59),
    ['BFIELD']
)
# this data is taken from cache
data = cdas.get_data(
    'sp_phys',
    'STA_L1_MAG_RTN',
    datetime(2010, 1, 1),
    datetime(2010, 1, 1, 0, 59, 59),
    ['BFIELD']
)
```

CHAPTER 2

API documentation

2.1 ai.cdas

The module implements an interface to the REST API of NASA CDAweb.

Refer to CDAS RESTful Web Services (https://cdaweb.gsfc.nasa.gov/WebServices/REST/) for detailed description of parameters.

exception ai.cdas.**NoDataError**
The exception that is raised when no data is found on the server.

ai.cdas.**get_data**(*dataview*, *dataset*, *startTime*, *stopTime*, *variables*, *cdf=False*, *progress=True*)
Queries server (and data cache) for data.

Parameters

- **dataview** (*string*) – Dataview.
- **dataset** (*string*) – Dataset.
- **startTime** (*datatime*) – First datetime for the requested data.
- **stopTime** (*datetime*) – Last datetime for the requested data.
- **variables** (*list*) – list of strings representing IDs of requested variables.
- **cdf** (*bool, optional*) – If True uses CDF data format for download, otherwise downloads in ASCII format. Defaults to False.
- **progress** (*bool, optional*) – If True displays the download progress bar. Defaults to True.

Returns Dictionary of data arrays or sequences.

Return type (dict)

ai.cdas.**get_datasets**(*dataview*, *observatoryGroup=None*, *instrumentType=None*, *observatory=None*, *instrument=None*, *startDate=None*, *stopDate=None*, *idPattern=None*, *labelPattern=None*, *notesPattern=None*)

Queries server for descriptions of the datasets.

Parameters

- **dataview** (*string*) – Dataview.
- **observatoryGroup** (*string, optional*) – Observatory group.
- **instrumentType** (*string, optional*) – Instrument type.
- **observatory** (*string, optional*) – Observatory.
- **instrument** (*string, optional*) – Instrument.
- **startDate** (*datetime, optional*) – Start date.
- **stopDate** (*datetime, optional*) – Stop date.
- **idPattern** (*string, optional*) – Id pattern.
- **labelPattern** (*string, optional*) – Label pattern.
- **notesPattern** (*string, optional*) – Notes pattern.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_dataviews**()

Queries server for descriptions of dataviews.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_instrument_types**(*dataview*, *observatory=None*, *observatoryGroup=None*)

Queries server for descriptions of instrument types.

Parameters

- **dataview** (*string*) – Dataview.
- **observatory** (*string, optional*) – Observatory.
- **observatoryGroup** (*string, optional*) – Observatory group.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_instruments**(*dataview*, *observatory=None*)

Queries server for descriptions of the instruments.

Parameters

- **dataview** (*string*) – Dataview.
- **Observatory** (*string, optional*) – Observatory.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_inventory**(*dataview*, *dataset*)

Queries server for descriptions of the data inventory within a dataset.

Parameters

- **dataview** (*string*) – Dataview.
- **dataset** (*string*) – Dataset.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_observatories**(*dataview*, *instrument=None*, *instrumentType=None*)
Queries server for descriptions of the observatories.

Parameters

- **dataview** (*string*) – Dataview.
- **instrument** (*string, optional*) – Instrument.
- **instrumentType** (*string, optional*) – Instrument type.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_observatory_groups**(*dataview*, *instrumentType=None*)
Queries server for descriptions of observatory groups.

Parameters

- **dataview** (*string*) – Dataview.
- **instrumentType** (*string, optional*) – Instrument type.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_observatory_groups_and_instruments**(*dataview*, *instrumentType=None*)
Queries server for descriptions of observatory groups (and associated instruments). This is a convenience/performance alternative to making multiple calls to get_observatory_groups, get_observatories, and get_instruments.

Parameters

- **dataview** (*string*) – Dataview.
- **instrumentType** (*string, optional*) – Instrument type.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**get_variables**(*dataview*, *dataset*)
Queries server for descriptions of variables in a dataset.

Parameters

- **dataview** (*string*) – Dataview.
- **dataset** (*string*) – Dataset.

Returns JSON response from the server.

Return type (dict)

ai.cdas.**set_cache**(*cache*, *directory=None*)
Sets the data cache.

Parameters

- **cache** (*bool*) – Flag for switching caching on/off.
- **directory** (*string, optional*) – Path to the cache directory.

Python Module Index

a

Index