Space Physics WebServices Client Documentation

Release 0.9.0

Alexis Jeandet

Nov 25, 2021

CONTENTS

CHAPTER

ONE

SPACE PHYSICS MADE EASY

Speasy is an open source Python client for Space Physics web services such as CDAWEB or AMDA. Most space physics data analysis starts with finding which server provides which dataset then figuring out how to download them. This can be difficult specially for students or newcomers, Speasy try to remove all difficulties by providing an unique and simple API to access them all. Speasy aims to support as much as possible web services and also cover a maximum of features they propose.

1.1 Quickstart

Installing Speasy with pip (*more details here*):

```
$ pip install speasy
# or
$ pip install --user speasy
```

Getting data is as simple as:

```
import speasy as spz
ace_mag = spz.get_data('amda/imf', "2016-6-2", "2016-6-5")
```

Where amda is the webservice and imf is the product id you will get with this request.

Using the dynamic inventory this can be even simpler:

```
import speasy as spz
amda_tree = spz.inventory.data_tree.amda
ace_mag = spz.get_data(amda_tree.Parameters.ACE.MFI.ace_imf_all.imf, "2016-6-2", "2016-6-
↪5")
```

Will produce the exact same result than previous example but has the advantage to be easier to manipulate since you can discover available data from your favourite Python environment completion such as IPython or notebooks (might not work from IDEs).

This also works with SSCWEB, you can easily download trajectories:

```
import speasy as spz
sscweb_tree = spz.inventory.data_tree.ssc
solo = spz.get_data(sscweb_tree.Trajectories.solarorbiter, "2021-01-01", "2021-02-01")
```

1.1.1 Documentation

Check out the documentation and examples at speasy documentation.

1.2 Features

- Simple and intuitive API (spz.get_data to get them all)
- Pandas DataFrame like interface for variables
- Quick functions to convert a variable to a Pandas DataFrame
- Local cache to avoid repeating twice the same request
- Can take advantage of SciQLop dedicated poxy as a community backed ultra fast cache
- Full support of AMDA API.

1.2.1 Examples

See here for a complete list of examples.

1.2.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Space Physics WebServices Client, run this command in your terminal:

```
$ pip install speasy
# or
$ pip install --user speasy
```

This is the preferred method to install Space Physics WebServices Client, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Space Physics WebServices Client can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/SciQLop/speasy
```

Or download the tarball:

```
$ curl  -OL https://github.com/SciQLop/speasy/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

MODULES

3.1 AMDA

3.1.1 AMDA_Webservice Products

List of AMDA_Webservice products.

3.1.1.1 AMDA_Webservice Datasets

Description of AMDAs dataset objects and their representation in `speasy`. In `speasy` dataset metadata is stored as dictionary objects. The dataset's attributes are :

- `id`
- `name`
- `dataStart` : begining of data
- `dataStop` : end of data
- `lastUpdate` : last modification
- `desc` : description string
- `dataSource` : data provider
- `spaseId` : SPASE identifier
- `sampling` : sampling rate
- `target` : mission target
- `att` : attributes
- `dataCenter` : data center identifier
- `mission` : mission identifier
- `instrument` : instrument identifier
- `dataset` : dataset identifier

3.1.1.2 AMDA_Webservice Parameter

Description of AMDAs parameter objects and their representation in `speasy`. In `speasy` parameter metadata is stored as dictionary objects. The parameter's attributes are :

- `id`
- `name`
- `units` : parameter units
- `description` : description string
- `displayType` : parameter display type (timeseries, spectral, etc....)
- `dataCenter` : data center identifier
- `mission` : mission identifier
- `instrument` : instrument identifier
- `dataset` : dataset identifier
- `parameter` : parameter identifier

3.1.1.3 AMDA_Webservice Timetables

Description of AMDAs timetable objects and their representation in `speasy`. In `speasy` timetable metadata are stored as dictionary objects. The timetable's attributes are :

- `id`
- `name`
- `buildchain` : the parameters formula as defined in AMDA_Webservice
- `timestep` : sampling rate in seconds
- `dim_1`
- `dim_2`

3.1.1.4 AMDA_Webservice Catalogs

Description of AMDAs catalog objects and their representation in `speasy`. In `speasy` catalog metadata is stored as dictionary objects. The catalog's attributes are :

- `id`
- `name`
- `created` : creation date
- `description` : description string
- `history`
- `parameters`
- `nbIntervals`
- `sharedBy`
- `sharedDate`

- `surveyStart` : begining of survey
- `surveyStop` : end of survey
- `folder` : parent folder ID
- `catalog`

3.1.2 AMDA_Webservice Examples

3.1.2.1 AMDA_Webservice Example 1: getting public dataset and parameter

In this script we download the `tao-ura-sw` dataset and the `imf` parameter on AMDA_Webservice.

```
import speasy as spz
from datetime import datetime

# define the parameter and dataset IDs and the data begining and end dates
parameter_id = "imf"
dataset_id = "ace-imf-all"
start, stop = datetime(2000, 1, 1), datetime(2000, 1, 2)

# get some information about the data time range
print("Parameter time range : {}".format(spz.amda.parameter_range(parameter_id)))
print("Dataset time range   : {}".format(spz.amda.parameter_range(dataset_id)))

# list parameters in dataset
print("Dataset parameters : {}".format(spz.amda.list_parameters(dataset_id=dataset_id)))

# download dataset contents, a list of SpeasyVariable objects
dataset = spz.amda.get_dataset(dataset_id, start, stop)
print(dataset)

# download parameter
param = spz.amda.get_parameter(parameter_id, start, stop)
print("Parameter : {}".format(param))
```

3.1.2.2 AMDA_Webservice Example 2: getting user defined parameter

The following example illustrates how to access parameters defined on your user account within AMDA_Webservice.

```
import speasy as spz
from datetime import datetime

# get list of users parameters
parameter_list = spz.amda.list_user_parameters()

# get each parameter between the 1st and 2nd of January 2000
start, stop = datetime(2000, 1, 1), datetime(2000, 1, 2)
for param in parameter_list:
    print("Parameter id : {}".format(param["id"]))
    # download data
```

(continues on next page)

(continued from previous page)

```
    p = spz.amda.get_user_parameter(param["id"], start, stop)
    print(p)
```

3.1.2.3 AMDA_Webservice Example 3: listing available parameters

Simple example illustrating how to list available parameters, datasets and timetables.

```
import speasy as spz

# get list of datasets
datasets = spz.amda.list_datasets()

# get parameter list
parameters = spz.amda.list_parameters()

# get time-table list
timetables = spz.amda.list_timetables()

print("AMDA_Webservice products")
print("\tDatasets   : {}".format(len(datasets)))
print("\tParameters : {}".format(len(parameters)))
print("\tTimetables : {}".format(len(timetables)))
```

Output

```
$ python amda_list_public_products.py
AMDA_Webservice products
     Datasets   : 935
     Parameters : 4696
     Timetables : 131
```

3.1.2.4 AMDA_Webservice Example 4: getting timetables

Download a public timetable.

```
import speasy as spz

ttid = "sharedtimeTable_0"

timetable = spz.amda.get_timetable(ttid)

print(timetable)
print("timetable id : {}".format(ttid))
print("time.shape   : {}".format(timetable.time.shape))
print("values.shape : {}".format(timetable.values.shape))
```

Output

```
$ python amda_get_public_timetable.py
<speasy.common.variable.SpeasyVariable object at 0x7f23cde66f90>
```

(continues on next page)

(continued from previous page)

```
timetable id : sharedtimeTable_0
time.shape   : (782,)
values.shape : (782, 2)
```

3.1.2.5 AMDA_Webservice Example 5: listing timetables

Listing public timetables.

```
import speasy as spz

# list timetable IDs
for ttid in spz.amda.list_timetables():
    print(ttid)
```

Output

```
$ python amda_list_public_timetables.py
sharedtimeTable_0
...
sharedtimeTable_130
{'xml:id': 'sharedtimeTable_139', 'id': 'sharedtimeTable_139', 'name': 'MMS_Burst_Mode_
↪2021July', 'created': '2020-08-26T00:00:55', 'modified': '1970-01-01T00:00:00',
↪'surveyStart': '2021-07-01T00:03:43', 'surveyStop': '2021-07-25T13:51:53', 'contact':
↪'MMS_Burst_Mode_2021July', 'description': 'Time intervals for which MMS burst mode␣
↪data are available. source: http://mmsburst.sr.unh.edu/', 'history': '', 'nbIntervals
↪': '441', 'sharedBy': 'AMDA_Webservice', 'sharedDate': '2021-08-02T03:25:05+00:00',
↪'folder': 'MMS_BURST_MODE_timeTable', 'timeTable': 'sharedtimeTable_139'}
...
```

See *AMDA_Webservice Timetables* for a description of the `timetable` fields.

3.1.2.6 AMDA_Webservice Example 6: listing catalogs

Listing public catalogs.

```
import speasy as spz

for catalog in spz.amda.list_catalogs():
    print(catalog)
```

Output

```
$ python amda_list_public_catalogs.py
sharedcatalog_0
...
sharedcatalog_130
{'xml:id': 'sharedcatalog_16', 'id': 'sharedcatalog_16', 'name': 'vex', 'created': '2020-
↪06-04T14:00:59', 'description': "Venus Express is a satellite optimised for studying␣
↪the atmosphere of Venus, from the surface right up to the ionosphere. Launched on␣
↪November 9th 2005, it arrived at Venus in April 2006 and continued operating for more␣
↪than eight years. The mission ended on December 16th 2014. This catalog has been␣
↪created using spice kernels provided by spdf and the JPL's orbnum tool (https://naif.
↪jpl.nasa.gov/pub/naif/utilities/PC_Linux_64bit/orbnum.ug)", 'history': '', 'parameters
↪': '', 'nbIntervals': '3162', 'sharedBy': 'AMDA_Webservice', 'sharedDate': '2020-09-
↪17T14:14:52+00:00', 'surveyStart': '2006-05-18T13:35:45', 'surveyStop': '2014-12-
↪31T12:23:54', 'folder': 'SPACECRAFT_ORBIT_NUMBERS_catalog', 'catalog': 'sharedcatalog_
↪16'}
```

(continues on next page)

(continued from previous page)

```
...
```

See *AMDA_Webservice Catalogs* for a description of the `catalog` fields.

3.1.2.7 AMDA_Webservice Example 7: getting catalogs

Download a public catalogs.

```
import speasy as spz

# loop over catalogs and download
catalog_id = spz.amda.list_catalogs()[0]
catalog = spz.amda.get_catalog(catalog_id)
print(catalog.values)
```

Output

```
$ python amda_get_public_catalog.py
[['2007-02-25 02:47:01' '2007-02-25 02:49:49' '3.0']
...
['2007-02-26 00:22:41' '2007-02-26 00:28:46' '3.0']]
```

3.1.2.8 AMDA_Webservice Example 8: listing user timetables

Listing private timetables.

```
import speasy as spz

# loop over user timetables
for utt in spz.amda.list_user_timetables():
    print(utt)
```

Output

```
$ python amda_list_user_timetables.py
{'name': 'output-1', 'intervals': '389', 'id': 'tt_0'}
{'name': 'output-12', 'intervals': '389', 'id': 'tt_1'}
{'name': 'output-newell', 'intervals': '55446', 'id': 'tt_2'}
{'name': 'output-newell-ext', 'intervals': '55446', 'id': 'tt_3'}
```

See *AMDA_Webservice Timetables* for a description of the `timetable` fields.

3.1.2.9 AMDA_Webservice Example 9: listing user catalogs

Listing private catalogs.

```
import speasy as spz

# loop over user catalogs
for ucat in spz.amda.list_user_catalogs():
    print(ucat)
```

Output

```
$ python amda_list_user_catalogs.py
{'name': 'mycata', 'intervals': '1457', 'id': 'cat_0'}
```

See *AMDA_Webservice Catalogs* for a description of the `catalog` fields.

3.1.2.10 AMDA_Webservice Example 10: getting user timetables

Downloading private timetables.

```
import speasy as spz


# get list of user timetables
for tt in spz.amda.list_user_timetables():
    print(spz.amda.get_user_timetable(tt["id"]))
```

Output

```
$ python amda_get_user_timetable.py
<speasy.common.variable.SpeasyVariable object at 0x7f4e8175bef0>
...
<speasy.common.variable.SpeasyVariable object at 0x7f4e80977f40>
```

See *AMDA_Webservice Timetables* for a description of the `timetable` fields.

3.1.2.11 AMDA_Webservice Example 11: getting user catalogs

Downloading private catalogs.

```
import speasy as spz

# get list of user catalog
for tt in spz.amda.list_user_catalogs():
    print(spz.amda.get_user_catalog(tt["id"]))
```

Output

```
$ python amda_get_user_catalog.py
<speasy.common.variable.SpeasyVariable object at 0x7f76472b76d0>
```

See *AMDA_Webservice Catalogs* for a description of the `catalog` fields.

3.1.3 Notebooks

Here are some jupyter-notebooks.

3.1.4 Getting AMDA_Webservice dataset and parameters

First import AMDA connection object:

```
>>> from speasy import amda
```

Downloading the data is done by using the `speasy.webservices.amda.ws.AMDA_Webservice.get_data()` or `speasy.webservices.amda.ws.AMDA_Webservice.get_parameter()` methods. For example getting *imf* data between 2000-01-01 and 2000-02-01:

```
>>> parameter = amda.get_data("imf", datetime.datetime(2000,1,1), datetime.datetime(2000,
↪2,1))
>>> parameter
<speasy.common.variable.SpeasyVariable object at 0x7f6c3b847bd0>
```

The resulting data is stored in a `speasy.common.variable.SpeasyVariable` object.

The parameters data is stored as a *numpy.ndarray* object:

```
>>> type(parameter.data)
<class 'numpy.ndarray'>
>>> parameter.data
array([[-3.432, -3.174,  5.714],
       [-3.407, -2.763,  5.72 ],
       [-4.38 , -1.437,  5.2  ],
       ...,
       [-4.435,  0.31 , -0.27 ],
       [-4.413,  0.141, -0.247],
       [-4.335,  0.087, -0.323]])
```

It is also possible to get all the parameters contained in a given dataset using the `speasy.amda.amda.AMDA_Webservice.get_dataset()` method which returns a list of `speasy.common.variable.SpeasyVariable` objects:

```
>>> amda.get_dataset("ace-imf-all", datetime.datetime(2000,1,1), datetime.datetime(2000,
↪2,1))
[<speasy.common.variable.SpeasyVariable object at 0x7f79fa8f3720>, <speasy.common.
↪variable.SpeasyVariable object at 0x7f79fa8fb950>, <speasy.common.variable.
↪SpeasyVariable object at 0x7f79fa859540>]
```

3.1.5 Parameter time range

Downloading data requires the user to know the desired start and end date of the data. The start and end dates for all available datasets and parameters are available through use of the `speasy.amda.amda.AMDA_Webservice.parameter_range()` method:

```
>>> t_range = amda.parameter_range("imf")
>>> type(t_range)
<class 'speasy.common.datetime_range.DateTimeRange'>
>>> t_range
1997-09-02T00:00:12->2021-07-17T23:59:55
```

3.1.6 Listing available products

Users can access the list of available parameters:

```
>>> parameter_list = amda.list_parameters()
>>> dataset_list = amda.list_datasets()
```

You can get the list of parameters contained in a dataset with:

```
>>> amda.get_dataset_parameters("ace-imf-all")
['imf_mag', 'imf', 'imf_gsm']
```

3.1.7 User parameters

Users with an account on AMDA_Webservice can access their private parameters. First store the users credentials using the `ConfigEntry` class:

```
>>> from speasy.config import ConfigEntry
>>> ConfigEntry("AMDA_Webservice", "username").set("your_username")
>>> ConfigEntry("AMDA_Webservice", "password").set("your_password")
```

The login credentials are stored locally, you only need to execute the previous lines of code once to save the credentials. The configuration file can be found at `/<user_config_dir>/speasy/config.ini`.

Parameters defined on your account can now be listed like follows:

```
>>> params = amda.list_user_parameters()
>>> for param in params:
>>>     print(param["id"], param["name"])
ws_0 your_first_parameter
ws_1 your_second_parameter
...
ws_n your_nth_parameter
```

Getting the parameter is then done with `get_user_parameter()`:

```
>>> from datetime import datetime
>>> start, stop = datetime(2000,1,1), datetime(2000,1,2)
>>> param = amda.get_user_parameter("ws_0", start, stop)
>>> param
```

(continues on next page)

(continued from previous page)

```
<speasy.common.variable.SpeasyVariable object at 0x7f8e6f31c220>
>>> param.data
array([[-6.156],
       [-6.15],
       [-6.088],
       ...,
       [-3.088],
       [-2.816],
       [-3.568]], dtype=object)
>>> p.time
array(['2001-01-01T00:00:00.000', '2001-01-01T00:00:16.000',
       '2001-01-01T00:00:32.000', ..., '2001-01-31T23:59:28.000',
       '2001-01-31T23:59:44.000', '2001-02-01T00:00:00.000'], dtype=object)
```

See `list_user_parameters()` for a list of user parameter attributes.

Calling `list_user_parameters()` or `get_user_parameter()` will raise an `UndefinedConfigEntry` exception if the credentials could not be found:

```
raise UndefinedConfigEntry(key1=self.key1, key2=self.key2, default=self.default)
```

3.1.8 Timetables and catalogs

The methods `get_timetable()` and `get_catalog()` allow you to download one of many timetables and catalogs available on AMDA_Webservice. Listing the publicly available products is achieved through using the `list_timetables()` and `list_catalogs()` to list products by ID:

```
>>> for ttid in amda.list_timetables():
>>>     print(ttid)
sharedtimeTable_0
...
sharedtimeTable_130
```

You can get more information about the timetables and catalogs through the `amda.timetable` and `amda.catalog` attributes. These attributes are dictionaries indexed by the ID of their respective products:

```
>>> for ttid in amda.timetable:
>>>     print(amda.timetable[ttid])
{'xml:id': 'sharedtimeTable_139', 'id': 'sharedtimeTable_139', 'name': 'MMS_Burst_Mode_
↪2021July', 'created': '2020-08-26T00:00:55', 'modified': '1970-01-01T00:00:00',
↪'surveyStart': '2021-07-01T00:03:43', 'surveyStop': '2021-07-25T13:51:53', 'contact':
↪'MMS_Burst_Mode_2021July', 'description': 'Time intervals for which MMS burst mode␣
↪data are available. source: http://mmsburst.sr.unh.edu/', 'history': '', 'nbIntervals
↪': '441', 'sharedBy': 'AMDA_Webservice', 'sharedDate': '2021-08-02T03:25:05+00:00',
↪'folder': 'MMS_BURST_MODE_timeTable', 'timeTable': 'sharedtimeTable_139'}
...
```

For a description of the `timetable` and `catalog` attributes see the `AMDA_Webservice` class documentation.

3.2 SSCWEB

3.3 CDAWEB

CHAPTER

FOUR

WELCOME TO SPEASY'S DEVELOPER DOCUMENTATION!

4.1 Indices and tables

- genindex
- modindex
- search

Go to developers doc