maser4py Documentation

Release 0.10.0

MASER Team

May 10, 2022

CONTENTS

MASER4PY is a Python 3 package to deal with the MASER portal services and data. The MASER portal provides a centralized access to services, data and tools relative to the low frequency radio astronomy.

For more information about MASER, please visit: http://maser.lesia.obspm.fr/

CHAPTER

ONE

OVERVIEW

maser4py modules are divided into three categories:

- **services**, Web service access clients and tools (i.e., Virtual Observatories, Data providers, etc.)
- **data**, radio astronomy data programs
- **utils**, generic utilities relative to the data formats and common tools

CHAPTER

TWO

CONTENT

The maser4py package is organized as follows:

maser/ services/ data/

- **cassini/** Module to handle the CASSINI mission data
- **cdpp/** Module to handle the data archived at CDPP (http://www.cdpp.eu/)
- **nancay/** Module to handle the Nancay Observatory data
- **padc/** Module to handle the PADC data (https://padc.obspm.fr/)
- **pds/** Module to handle the NASA PDS data (https://pds.nasa.gov/)
- **psa/** Module to handle the ESA PSA data (https://archives.esac.esa.int/psa/)
- **radiojove/** Module to handle the Radiojove project data (http://www.radiojove.org/)
- **wind/** Module related to the Wind mission data

utils/

- **cdf/** Module to handle the NASA Common Data Format (CDF)
- **das2stream/** Module related to DAS2 server (https://das2.org/)
- **time/** Module to handle time.
- **toolbox/** Module containing common tools for maser4py

maser4py support data files are stored in:

maser/

maser/support Directory containing support data

CHAPTER

THREE

DETAILS

3.1 Installation

3.1.1 System Requirements

In order to install maser4py, make sure to have Python 3.4 (or higher) as well as the pip and setuptools programs available on your system.

If there are not found, the following Python modules will be also installed:

- openpyxl
- numpy
- matplolib
- sphinx

maser4py has been tested on the following Operating Systems:

- Mac OS X 10.10 or higher
- Debian Jessie 8.2 or higher

Warning: Make sure that the NASA CDF software distribution is installed and configured on your system. Visit https://cdf.gsfc.nasa.gov/ for more details.

3.1.2 How to install maser4py?

Using pip

If the pip tool is installed on your system, just enter:

```
pip install maser4py
```

From source code

1. To get MASER4PY source files from Github, enter:

```
git clone https://github.com/maserlib/maser4py.git
```

Make sure to have Git (https://git-scm.com/) installed on your system.

If everything goes right, you should have a new local "maser4py" directory created on your disk.

2. From the "maser4py" main directory, install program dependencies:

```
pip install -r requirements.txt
```

3. To install the package on your system, enter the following command :

```
python3 setup.py install
```

This should install the maser4py package on your system.

To check that the installation ends correctly, you can enter:

```
maser4py
```

, which should return something like:

```
"This is maser4py package VX.Y.Z"
```

If you have an issue during installation, please read the "Troubleshooting" section for help.

3.1.3 How to run maser4py?

If the installation has ended correctly, you can run maser4py:

- From a Python interpreter session, by entering "import maser".
- Using the command line interface available for some maser4py modules.

For more details about the MASER4PY modules, please read the user manual.

3.2 Data

3.2.1 The *wind* module

The *Wind* module provides methods to deal with the Wind NASA mission data.

3.2.2 The *cdpp* module

The *CDPP* module provides classes and methods to deal with the CDPP deep archive data.

3.2.3 The *cassini* module

The *Cassini* module provides classes and methods to deal with the Cassini/RPWS/HFR (Kronos) data from LESIA/ObsParis.

3.2.4 The *nancay* module

The *Nancay* module provides classes and methods to deal with the Station de Radioastronomie de Nancay data.

The *nda* submodule

The *NDA* module provides classes and methods to deal with the Nancay Decameter Array data.

The *junon* submodule

3.2.5 The *radiojove* module

The *RadioJove* module provides methods to deal with the RadioJove data.

3.3 Services

No service implemented currently.

3.4 Utilities

The utils/ module of MASER4PY provides utilities to deal with data in MASER.

3.4.1 The *cdf* module

The *cdf* module contains the following tools:

- *cdf*, a backup of the spacepy.pycdf module (https://pythonhosted.org/SpacePy/pycdf.html), only used in the case where spacepy package is not installed in the system.
- *cdfcompare*, a tool to compare two CDFs
- *serializer*, to convert skeleton CDF files (in Excel or ASCII format) into master CDF binary files
- *validator*, to validate the content of a CDF file from a given data model.

For more information about the CDF format, please visit http://cdf.gsfc.nasa.gov/.

cdf.cdfcompare

The *cdf.cdfcompare* module can be used to compare the content of two CDFs.

To get the full list of input parameters, print the help message from command line:

```
maser cdf_compare --help
```

cdf.serializer

The *cdf.serializer* module allows users to convert CDF between the following formats:

- Skeleton table (ASCII)
- Binary CDF ("Master")
- Excel 2007 format (.xlsx)

Module can be imported in Python programs or called directly from a terminal using the dedicated command line interface.

Important: The conversions between skeleton table and binary CDF are performed calling the *skeletontable* and *skeletoncdf* programs from the NASA CDF software (visit https://cdf.gsfc.nasa.gov/ for more details).

Converting a binary CDF to a skeleton table

The *skeletontable* sub-command allows to convert CDF from binary CDF to skeleton table by calling the skeletontable NASA CDF software program.

An option also permits to export skeleton table into Excel 2007 format file (.xlsx).

To display the help, enter:

```
maser skeletontable --help
```

Examples

Convert all input CDFs named "input_*.cdf" into skeleton tables. Output files will be saved into the /tmp/cdf/build folder. Output files will have the same names than input CDFs but with the extension ".skt".

```
maser skeletontable "input_*.cdf" --output_dir /tmp/cdf/build
```

Convert the input binary CDF named "input1.cdf" into skeleton table. Output files will be saved into the /tmp/cdf/build folder. Output file will have the same name than input CDF but with the extension ".skt". An export file in Excel 2007 format (input1.xlsx) will also be saved.

```
maser skeletontable "input1.cdf" --output_dir /tmp/cdf/build --to-xlsx
```

Converting skeleton CDF to binary CDF

The *skeletoncdf* sub-command allows to convert CDF from skeleton table to binary CDF by calling the skeletoncdf NASA CDF software program.

An option also permits to use Excel 2007 format file (.xlsx) as an input to the *skeletoncdf* sub-command.

To display the help, enter:

```
maser skeletoncdf --help
```

Expected Excel file format description

This section describes the structure of the Excel format file that can be used by the cdf.serializer module.

Note that:

- Only the Excel 2007 format is supported (i.e., .xlsx).
- Only zVariables are supported

Warning: Make sure to respect the letter case!

The Excel file shall contain the following sheets:

- header
- GLOBALattributes
- zVariables
- VARIABLEattributes
- NRV

The first row of each sheet must be used to provide the name of the columns.

header sheet

The "header" sheet must contain the following columns:

CDF_NAME Name of the CDF master file (without the extension)

DATA ENCODING Type of data encoding

MAJORITY Majority of the CDF data parsing ("COLUMN" or "ROW")

FORMAT Indicates if the data are saved in a single ("SINGLE") or on multiple ("MULTIPLE") CDF files

CDF_COMPRESSION Type of compression applied to the CDF

CDF_CHECKSUM Checksum applied to the CDF

GLOBALattributes sheet

The "GLOBALattributes" sheet shall contain the following columns:

Attribute Name Name of the global attribute

Entry Number Index of the current entry starting at 1

Data Type CDF data type of the global attribute (only the "CDF_CHAR" type is supported)

Value Value of the current entry

zVariables sheet

The "zVariables" sheet shall contain the following columns:

Variable Name Name of the zVariable

Data Type CDF data type of the zVariable

Number Elements Number of elements of the zVariable (shall be always 1, except for CDF_[U]CHAR" type)

Dims Number of dimension of the zVariable (shall be 0 if the variable is a scalar)

Sizes If the variable is not a scalar, provides its dimension sizes.

Record Variance Indicates if the variable values can change ("T") or not ("F") from a record to another.

Dimension Variances Indicates how the variable values vary over each dimension.

VAR_COMPRESSION Compression algorithm applied to the variable.

VAR_SPARESERECORDS Spare record of the variable.

VAR_PADVALUE Pad value of the variable.

VARIABLEattributes sheet

The "VARIABLEattributes" sheet shall contain the following columns:

Variable Name Name of the zVariable

Attribute Name Name of the variable attribute

Data Type CDF data type of the variable attribute

Value Value of the variable attribute

NRV sheet

The "NRV" sheet shall contain the following columns:

Variable Name Name of the zVariable

Index Index of the current NR row

Value Value of the current NR row

Limitations & Known Issues

The *cdf.validator* tool

validator provides methods to validate a CDF format file from a given model.

It contains only one *Validate* class that regroups all of the validation methods.

The *Validate* class

To import the *Validate* class from Python, enter:

```
from maser.utils.cdf.cdfvalidator import Validate
```

The Model validation test

The *Validate* class allows user to check if a given CDF format file contains specific attributes or variables, by providing a so-called "cdfvalidator model file".

This model file shall be in the JSON format. All items and values are case sensitive. It can include the following JSON objects:

Table 1: CDFValidator JSON objects

JSON object	Description
GLOBALattributes	Contains the list of global attributes to check
VARIABLEattributes	Contains the list of variable attributes to check
zVariables	Contains the list of zvariables to check

Note that any additional JSON object will be ignored.

The table below lists the JSON items that are allowed to be found in the *GLOBALattributes*, *VARIABLEattributes* and *zVariables* JSON objects.

Table 2: CDFValidator JSON object items

JSON item	JSON type	Priority	Description
attributes	vector	optional	List of variable attributes. An element of the vector shall be a JSON object that can contain one or more of the other JSON items listed in this table
dims	integer	optional	Number of dimensions of the CDF item
entries	vector	optional	Entry value(s) of the CDF item to be found
hasvalue	boolean	optional	If it is set to true, then the current CDF item must have at least one nonzero entry value
name	string	mandatory	Name of the CDF item (attribute or variable) to check
sizes	vector	optional	Dimension sizes of the CDF item
type	attribute	optional	CDF data type of the CDF item

Command line interface

To display the help of the module, enter:

```
cdf_validator --help
```

The full calling sequence is:

```
maser cdf_validator [-h] [-m MODEL_FILE] [-c CDFVALIDATE_BIN] [-I] [-C] cdf_file
```

positional arguments: cdf_file Path of the CDF format file to validate

optional arguments:

-h, --help show this help message and exit

-m MODEL_FILE, --model-file MODEL_FILE Path to the model file in JSON format

-c CDFVALIDATE_BIN, --cdfvalidate-bin CDFVALIDATE_BIN Path of the cdfvalidate NASA CDF tool executable

-I, --istp Check the ISTP guidelines compliance

-C, --run-cdfvalidate Run the cdfvalidate NASA CDF tool

Examples

To test the cdf.validator program, use the dedicated scripts/test_cdfvalidator.sh bash script.

It should return something like:

```
INFO    : Opening /tmp/cdfconverter_example.cdf
INFO    : Loading /Users/xbonnin/Work/projects/MASER/Software/Tools/Git/maser-py/
↪scripts/../maser/support/cdf/cdfvalidator_model_example.json
INFO    : Checking GLOBALattributes:
INFO    : --> Project
WARNING : "Project"  has a wrong entry value: "Python>Python 2" ("Python>Python 3"␣
↪expected)!
INFO    : --> PI_name
INFO    : --> TEXT
INFO    : Checking VARIABLEattributes:
INFO    : --> FIELDNAM
INFO    : --> CATDESC
INFO    : --> VAR_TYPE
INFO    : Checking zVariables:
INFO    : --> Epoch
INFO    : --> Variable2
INFO    : Checking variable attributes of "Variable2":
INFO    : --> DEPEND_0
WARNING : DEPEND_0 required!
INFO    : Closing /tmp/cdfconverter_example.cdf
```

3.4.2 The *time* module

The *leapsec* tool

The *leapsec* tool allows users to handle the leap seconds.

Using the leapsec tool requires to read the CDFLeapSeconds.txt file. This file is available on the NASA CDF Web site (https://cdf.gsfc.nasa.gov).

Warning: Before using the leapsec tool, it is highly recommended to have the CDFLeapSeconds.txt file saved on the localdisk, and reachable from the $CDF_LEAPSECONDSTABLE env. variable. If the file is not on the disk, the tool will attempt to read the file directly from the NASA CDF Web site.

The *Lstable* class

The Lstable class provides the methods to deal with the CDFLeapSeconds.txt table file.

To import the *Lstable* class from Python, enter:

```
from maser.utils.time import Lstable
```

Then, to load the CDFLeapSeconds.txt table, first enter:

```
lstable = Lstable(file=path_to_the_file)
```

Note: Note that if the optional input keyword *file=* is not set, the tool will first check if the path is given in the $CDF_LEAPSECONDSTABLE environment variable. If not, then the program will look into the maser/support/data sub-folder of the package directory. Finally, if it is still not found, it will attempt to retrieve the table data from the file on the NASA CDF Web site (https://cdf.gsfc.nasa.gov/html/CDFLeapSeconds.txt)

Once the table is loaded, then to print the leap seconds table, enter:

```
print(lstable)
```

To get the total elapsed leap seconds for a given date, enter:

```
lstable.get_leapsec(date=date_time)
```

Where date_time is a datetime object of the datetime module.

Downloading the CDFLeapSeconds.txt file from the NASA Web site can be done by entering:

```
Lstable.get_lstable_file(target_dir=target_dir, overwrite=overwrite)
```

Where *target_dir* is the local directory where the CDFLeapSeconds.txt file will be saved. *overwrite* keyword can be used to replace existing file (default is *overwrite=False*)

Note: get_lstable_file is a staticmethod, which does not require to instanciate the Lstable class.

Note: If the method is called without the *target_dir=* input keyword (i.e., *get_lstable()*), then it will first check if the $CDF_LEAPSECONDSTABLE env. variable is defined, if yes the *target_dir* will be set with the $CDF_LEAPSECONDSTABLE value, otherwise the file is saved in the maser/support/data folder of the module.

Command line interface

To display the help of the module, enter:

```
maser leapsec --help
```

The full calling sequence is:

```
maser leapsec [-h] [-D] [-O] [-S] [-f FILEPATH] [-d DATE]
```

Input keywords:

-h, --help show this help message and exit

-f FILEPATH, --filepath FILEPATH CDFLeapSeconds.txt filepath. Default is [maser4py_rootdir]/support/data/CDFLeapSeconds.txt, where [maser4py_rootdir] is the maser4py root directory.

-d DATE, --date DATE Return the leap seconds for a given date and time. (Expected format is "YYYY-MM-DDThh:mm:ss")

-S, --SHOW-TABLE Show the leap sec. table

-O, --OVERWRITE Overwrite existing file

-D, --DOWNLOAD-FILE Download the CDFLeapSeconds.txt from the NASA CDF site. The file will be saved in the path defined in the –filepath argument..

The *time* tool

The *time* tool offers time conversion methods between the following time systems:

- UTC: Coordinated Universal Time
- JD: Julian Days
- MJD: Modified Julian Days
- TT: Terrestrial Time
- TAI: International Atomic Time
- TT2000: Terrestrial Time since J2000 (2000-01-01T12:00:00)

Note: The time conversion inside the methods is performed using numpy.timedelta64 and numpy.datetime64 objects for better time resolution.

Warning: The highest time resolution of JD and MJD systems are fixed to microsecond. The TT2000 system can reach the nanosecond resolution.

3.5 Troubleshooting

Here a list of known issues. If the problem persists, you can contact the MASER developer team at: maser.support@groupes.renater.fr.

- **//I have an error message "[Errno 13] Permission denied:" during installation//:**

This means that you don't have the right to install the package in your Python "site-packages" local directory. To solve this problem, install the package as a super user (e.g., using sudo command for instance), or modify the "site-packages" user access permissions.

- **//I have an error message "Exception: Cannot find CDF C library. Try os.putenv("CDF_LIB", library_directory) before import." during the installation"//:**

This means that the %%$CDF_LIB%% environmment variable is not set. This variable is required to run the CDF software distribution from Python. For more information about how to set up this software, visit the CDF home page at http://cdf.gsfc.nasa.gov/.

CHAPTER

FOUR

SUPPORT

The maser4py is developed at the LESIA (http://www.lesia.obspm.fr/) by:

- Xavier Bonnin - xavier dot bonnin at obspm dot fr
- Baptiste Cecconi - baptiste dot cecconi at obspm dot fr
- Sonny Lion - sonny dot lion at obspm dot fr
- Alan Loh - alan dot loh at obspm dot fr
- Quynh Nhu Nguyen - quynh-nhu dot nguyen at obspm dot fr

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search