
eph Documentation

Release 1.1.4

Flavio Grandin

Jul 20, 2019

Contents:

1	Documentation	3
1.1	Install	3
1.2	Getting started	3
1.3	License	5
1.4	eph	5
	Python Module Index	13
	Index	15

eph package aims to provide functions, classes and tools to *represent*, *retrieve* and *manipulate* ephemerides in a *astropy*-compatible way.

1.1 Install

Run

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

1.1.1 Dependencies

The `eph` package depends on

- `requests` (visit <http://docs.python-requests.org/en/master/> for more info).
- `astropy` (visit <http://astropy.org> for more info).

1.1.2 Package Managers

- `eph` is on the python package index [PyPI](#): you can install it using `pip`:

```
$ pip install eph
```

1.2 Getting started

The `eph` package provides some useful functions, classes and tools to *retrieve*, *parse* and *manipulate* ephemerides in an `astropy`-compatible way.

See [eph-howto](#) (jupyter notebook) for more info.

1.2.1 Basic Usage

```
import eph

req = eph.JplReq() # create the request
req.read('eph.ini', section='jplparams') # read parameters from 'jplparams' section_
↳in 'eph.ini'
req.set({
    'COMMAND': 'venus',
    'START_TIME': '2007-11-17',
    'STOP_TIME': '2017-4-22'
    'STEP_SIZE': '10d'
}) # set parameters from dictionary
req['OBJ_DATA'] = False # set parameter dict-like
req.csv = True # set parameter as attributes
req.set(
    TABLE_TYPE='V',
    VEC_LABELS=False,
    VEC_TABLE=1
) # set position vectors output

res = req.query() # perform the request obtaining a response from Jpl Horizons service
e = res.parse() # parse the ephemeris in an astropy QTable

from astropy.io import ascii

ascii.write(e, format='csv') # write output data
```

The content of `eph.ini` can be something like this (see ftp://ssd.jpl.nasa.gov/pub/ssd/horizons_batch_example.long for a complete description of JPL parameters)

```
[jplparams]
MAKE_EPHEM=YES
REFERENCE_PLANE=ECLIPTIC
REF_SYSTEM=J2000
OUT_UNITS=AU-D
```

1.2.2 Shortcuts

`eph` package defines also some useful shortcut functions to easily access Jpl Horizons data. Instead of building a `JplReq` and get back a `JplRes` to parse, you can get an astropy `QTable` with

```
from eph import *
from datetime import datetime

e = get('venus', dates=['2000-1-1', datetime.now()])
```

Shortcut functions accept also one-date queries (non-interval) and multiple target objects. Behind the scenes `eph` makes multiple calls to JPL Horizons system and merge the results in one table. In this case non-key (used to join) columns are renamed with a prefix referring to the object (e.g. column `X` for object `venus` becomes `venus_X`). Meta info are listified and collapsed in a single value only if they take the same value for all objects.

```
from eph import *

e = get(['venus', 'mars'], dates='2017-04-22')
```


Dates has `datetime.now()` as default value so it can be omitted if you want present data.

```
from eph import *

e = get(['venus', 'mars'], table_type='V', vec_table=1) # present vector positional_
↳data for Venus and Mars
```

There are other shortcut functions like `vec`, `pos`, `vel`, `elem`, `obs`, `radec`, `altaz`, etc.. to simplify parameter settings.

For example, if you want vectors, type

```
e = vec('venus', dates=['2018-1-1', '2020-1-1']).
```

1.2.3 Command line tool

eph package also provides a command line tool:

```
$ eph venus --dates 2007-11-07 2017-04-22
```

This command gives you an ephemeris table of Venus starting from 2007-11-17 to 2017-4-22. You can also change the reference frame, the time-step size, the output etc.. through the options provided or setting up a config file. Check available options typing

```
$ eph --help
```

1.3 License

Copyright 2017 Flavio Grandin

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.4 eph

1.4.1 eph package

Submodules

eph.cli module

eph console script module.

`eph.cli.main()`

eph.config module

`eph.config.create_config_file(out_filename='/home/docs/.ephrc')`

`eph.config.get_config_dir()`

`eph.config.get_config_file()`

`eph.config.get_default_config_file()`

`eph.config.read_config(filename=None, section=None)`

eph.exceptions module

Defines *eph* package related exceptions.

exception `eph.exceptions.ConfigError`

Bases: `eph.exceptions.EphError`

Base class for configuration related errors.

exception `eph.exceptions.ConfigNotFoundError`

Bases: `eph.exceptions.ConfigError`

A `ConfigNotFoundError` is raised when config file is not found.

exception `eph.exceptions.ConfigParserError`

Bases: `eph.exceptions.ConfigError`

A `ConfigParser` is raised when problems are encountered parsing a config file.

exception `eph.exceptions.EphError`

Bases: `Exception`

Base class for *eph* package related exceptions.

exception `eph.exceptions.JplBadParamError`

Bases: `eph.exceptions.JplError`

A `JplBadParamError` is raised when a `JplReq` tries to set a parameter that do not match with Jpl Horizons specifications.

exception `eph.exceptions.JplBadReqError`

Bases: `eph.exceptions.JplError`

A `JplBadReqError` exceptions is raised when a request to the Horizons service cannot be interpreted by the Horizons system.

exception `eph.exceptions.JplError`

Bases: `eph.exceptions.EphError`

Base class for JPL Horizons service related errors.

exception `eph.exceptions.ParserError`

Bases: `eph.exceptions.EphError`

A `ParserError` exception is raised when problems are found in parsing a response from the Horizons service.

eph.horizons module

Defines variables and functions used to interfacing with JPL Horizons system.

`eph.horizons.codify_obj(name)`

Tries to translate a human readable celestial object name to the corresponding Jpl Horizons code. If the name is not known the name itself will be returned.

Parameters `name` (*str*) – the name to be translated.

Returns the code of the object (stringified version of the id).

Return type `str`

`eph.horizons.codify_site(name)`

Tries to translate a human readable celestial object name to the corresponding Jpl Horizons site code.

If the name is not known the name itself will be returned preceded by a @ sign if @ is not already present in the name.

Parameters `name` (*str*) – the name to be translated.

Returns the code of the site.

Return type `str`

`eph.horizons.format_time(t)`

Modify time data `t` so that `str(t)` can be interpreted by Jpl.

Parameters

- `t` – the time data. It can be a `str`, an `astropy.time.Time` object
- **an object such as `str`** (*or*) –

Returns the final object.

`eph.horizons.get_col_dim(col)`

Get the physical dimension of a column by its name.

Parameters `col` (*str*) – the name of the column.

Returns the physical dimensions of the given column.

Return type `str`

`eph.horizons.humanify(code)`

Tries to interpret a Jpl object or site code as a human readable celestial object name.

Parameters `code` (*str*) – the code to be translated.

Returns the corresponding human readable name.

Return type `str`

`eph.horizons.is_jpl_param(key)`

Checks if a key is a Jpl Horizons parameter or a defined alias.

Parameters `key` (*str*) – the parameter to be checked.

Returns Whether key is or not a Jpl parameter.

Return type `boolean`

`eph.horizons.transform(key, value)`

Transforms an input key-value pair in a Jpl-compatible one.

Parameters

- **key** (*str*) – the key to be interpreted or translated.
- **value** – a *str* to be translated or the object such as `str(value)` is Jpl-compatible.

Returns the final key-value pair.

Return type tuple

`eph.horizons.transform_key(key)`

Transforms an input key to a Jpl-compatible parameter key.

Parameters **key** (*str*) – the key to be interpreted and translated.

Returns the interpreted Jpl-compatible key.

Return type str

Raises `JplBadParamError`

`eph.horizons.transform_value(key, value)`

Tries to transform an input value into a Jpl-compatible one or it leaves as is.

Parameters

- **key** (*str*) – the Jpl-compatible key.
- **value** – a *str* to be translated or an object such as `str(value)`
- **be interpreted by Jpl.** (*can*) –

Returns the transformed value.

Return type str

eph.interface module

Contains classes and functions useful to interact with the [Jpl Horizons service](#) from NASA.

class `eph.interface.JplReq(*args, **kwargs)`

Bases: `eph.models.BaseMap`

A requests to Jpl Horizons service.

It can be thought as a `dict` where key-value pairs represents Jpl Horizons parameters. Jpl parameters can be also set as attributes of the `JplReq` instance. Furthermore, keys and values are adjusted to match Jpl Horizons interface in order to enhance readability and usability.

query()

Performs the query to the Jpl Horizons service.

Returns the response from Jpl Horizons service.

Return type `JplRes`

Raises `ConnectionError`

read (*filename*, *section*='DEFAULT')

Reads configurations parameters from an ini file.

Reads the *section* section of the ini config file *filename* and set all parameters for the Jpl request.

Parameters

- **filename** (*str*) – the config file to be read.
- **section** (*str*) – the section of the ini config file to be read.

Returns the object itself.

Return type `JplReq`

url()

Calculate the Jpl Horizons url corresponding to the `JplReq` object.

Returns the url with the Jpl parameters encoded in the query string.

Return type `str`

class `eph.interface.JplRes` (*http_response*)

Bases: `object`

A response from the Jpl Horizons service.

get_data()

get_footer()

get_header()

parse (*target=<class 'astropy.table.table.QTable'>*)

Parse the http response from Jpl Horizons and return, according to target.

- an `astropy.table.Table` object.
- an `astropy.table.QTable` object.

raw()

Returns the content of the Jpl Horizons http response as is.

eph.models module

class `eph.models.BaseMap` (**args, **kwargs*)

Bases: `collections.abc.MutableMapping`

set (**args, **kwargs*)

eph.parsers module

Defines parsing functions to read Jpl Horizons ephemeris.

`eph.parsers.check_csv` (*source*)

`eph.parsers.get_sections` (*source*)

Split a Jpl Horizons ephemeris in header, data and footer.

Args: *source* (`str`): the content of the Jpl Horizons ephemeris data output.

Returns a tuple of strings containing header, data and footer sections respectively.

Return type `tuple`

`eph.parsers.get_subsections` (*source*)

Split a source string in a list of sections separated by one or more `*`.

Parameters *source* (`str`) – the source string to be splitted.

Returns the lists of subsections.

Return type `list`

`eph.parsers.parse` (*source, target=<class 'astropy.table.table.QTable'>*)

Parses an entire Jpl Horizons ephemeris and build an `astropy` table out of it.

Parameters

- **source** (*str*) – the content of the Jpl Horizons data file.
- **target** – the type of table to produce (Table or QTable).

Returns the table containing data from Jpl Horizons source ephemeris.

Return type table

`eph.parsers.parse_cols(header)`

Finds and parses ephemeris column names in a Jpl Horizons ephemeris.

Parameters **header** (*str*) – the header of a Jpl Horizons ephemeris.

Returns a tuple with the names of columns.

Return type tuple

`eph.parsers.parse_data(data, **kwargs)`

Parses the data section of a Jpl Horizons ephemeris in a *list of lists* table.

Parameters **data** (*str*) – the section containing data of a Jpl Horizons ephemeris.

Returns the list of lists representing a data table.

Return type list

`eph.parsers.parse_meta(header)`

`eph.parsers.parse_params(source)`

`eph.parsers.parse_units(meta)`

eph.shortcuts module

Defines shortcut functions useful to ease the access of Jpl Horizons data.

`eph.shortcuts.altaz(objs, site_coord='0, 0, 0', dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932163), **kwargs)`

Shortcut function to directly obtain an astropy QTable with ALT/AZ data.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.
- **site_coord** – comma separated value for longitude, latitude, altitude of a site.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.elem(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932160), **kwargs)`

Shortcut function to directly obtain an astropy QTable with orbital elements.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.get(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932145), **kwargs)`

Shortcut function to directly obtain an astropy QTable from Jpl Horizons parameters without building a JplReq and get a JplRes out of it to be parsed.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.obs(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932161), **kwargs)`

Shortcut function to directly obtain an astropy QTable with observable quantities.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.pos(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932158), **kwargs)`

Shortcut function to directly obtain an astropy QTable with position- only vector data.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.radec(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932162), **kwargs)`

Shortcut function to directly obtain an astropy QTable with RA/DEC data.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.vec(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932157), center='@0', **kwargs)`

Shortcut function to directly obtain an astropy QTable with vector data.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

`eph.shortcuts.vel(objs, dates=datetime.datetime(2019, 7, 20, 16, 50, 8, 932159), **kwargs)`

Shortcut function to directly obtain an astropy QTable with velocity- only vector data.

Parameters

- **objs** – The celestial objects to be targeted.
- **dates** – start and stop (optional) time.

Returns The data structure containing ephemeris data.

Return type `astropy.table.Qtable`

eph.util module

```
eph.util.addparams2url(url, params)
eph.util.clean_row(row)
eph.util.is_vector(obj)
eph.util.numberify(data)
eph.util.parse_row(raw, cols_del=', ')
eph.util.parse_table(raw, cols_del=', ', rows_del="\r?\n")
eph.util.path(filename)
eph.util.transpose(data)
eph.util.wrap(s, wrapper="", to_strip="")
eph.util.yes_or_no(value, yes='YES', no='NO')
```

Module contents

eph package aims to provide useful classes, functions and tools to *retrieve*, *represent* and *manipulate* ephemerides.

e

- `eph`, [12](#)
- `eph.cli`, [6](#)
- `eph.config`, [6](#)
- `eph.exceptions`, [6](#)
- `eph.horizons`, [7](#)
- `eph.interface`, [8](#)
- `eph.models`, [9](#)
- `eph.parsers`, [9](#)
- `eph.shortcuts`, [10](#)
- `eph.util`, [12](#)

A

`addparams2url()` (in module *eph.util*), 12
`altaz()` (in module *eph.shortcuts*), 10

B

`BaseMap` (class in *eph.models*), 9

C

`check_csv()` (in module *eph.parsers*), 9
`clean_row()` (in module *eph.util*), 12
`codify_obj()` (in module *eph.horizons*), 7
`codify_site()` (in module *eph.horizons*), 7
`ConfigError`, 6
`ConfigNotFoundError`, 6
`ConfigParserError`, 6
`create_config_file()` (in module *eph.config*), 6

E

`elem()` (in module *eph.shortcuts*), 10
eph (module), 12
eph.cli (module), 6
eph.config (module), 6
eph.exceptions (module), 6
eph.horizons (module), 7
eph.interface (module), 8
eph.models (module), 9
eph.parsers (module), 9
eph.shortcuts (module), 10
eph.util (module), 12
`EphError`, 6

F

`format_time()` (in module *eph.horizons*), 7

G

`get()` (in module *eph.shortcuts*), 10
`get_col_dim()` (in module *eph.horizons*), 7
`get_config_dir()` (in module *eph.config*), 6
`get_config_file()` (in module *eph.config*), 6

`get_data()` (*eph.interface.JplRes* method), 9
`get_default_config_file()` (in module *eph.config*), 6
`get_footer()` (*eph.interface.JplRes* method), 9
`get_header()` (*eph.interface.JplRes* method), 9
`get_sections()` (in module *eph.parsers*), 9
`get_subsections()` (in module *eph.parsers*), 9

H

`humanify()` (in module *eph.horizons*), 7

I

`is_jpl_param()` (in module *eph.horizons*), 7
`is_vector()` (in module *eph.util*), 12

J

`JplBadParamError`, 6
`JplBadReqError`, 6
`JplError`, 6
`JplReq` (class in *eph.interface*), 8
`JplRes` (class in *eph.interface*), 9

M

`main()` (in module *eph.cli*), 6

N

`numberify()` (in module *eph.util*), 12

O

`obs()` (in module *eph.shortcuts*), 11

P

`parse()` (*eph.interface.JplRes* method), 9
`parse()` (in module *eph.parsers*), 9
`parse_cols()` (in module *eph.parsers*), 10
`parse_data()` (in module *eph.parsers*), 10
`parse_meta()` (in module *eph.parsers*), 10
`parse_params()` (in module *eph.parsers*), 10
`parse_row()` (in module *eph.util*), 12

`parse_table()` (in module *eph.util*), 12
`parse_units()` (in module *eph.parsers*), 10
`ParserError`, 6
`path()` (in module *eph.util*), 12
`pos()` (in module *eph.shortcuts*), 11

Q

`query()` (*eph.interface.JplReq* method), 8

R

`radec()` (in module *eph.shortcuts*), 11
`raw()` (*eph.interface.JplRes* method), 9
`read()` (*eph.interface.JplReq* method), 8
`read_config()` (in module *eph.config*), 6

S

`set()` (*eph.models.BaseMap* method), 9

T

`transform()` (in module *eph.horizons*), 7
`transform_key()` (in module *eph.horizons*), 8
`transform_value()` (in module *eph.horizons*), 8
`transpose()` (in module *eph.util*), 12

U

`url()` (*eph.interface.JplReq* method), 9

V

`vec()` (in module *eph.shortcuts*), 11
`vel()` (in module *eph.shortcuts*), 11

W

`wrap()` (in module *eph.util*), 12

Y

`yes_or_no()` (in module *eph.util*), 12