leapseconddata

Release 1.1.0

Jeff Epler

Oct 09, 2022

CONTENTS:

Use the list of known and scheduled leap seconds

For example, to retrieve the UTC-TAI offset on January 1, 2011:

```
>>> import datetime
>>> import leapseconddata
>>> ls = leapseconddata.LeapSecondData.from_standard_source()
>>> when = datetime.datetime(2011, 1, 1, tzinfo=datetime.timezone.utc)
>>> ls.tai_offset(when).total_seconds()
34.0
```

exception leapseconddata.**InvalidHashError**

The file hash could not be verified

class leapseconddata.**LeapSecondData**

Represent the list of known and scheduled leapseconds

Parameters

- **leap_seconds** (*List[LeapSecondInfo]*) – A list of leap seconds
- **valid_until** (*Optional[datetime.datetime]*) – The expiration of the data, if available
- **updated** (*Optional[datetime.datetime]*) – The last update time of the data, if available

static __new__(*cls*, *leap_seconds*, *valid_until=None*, *last_updated=None*)

Create new instance of _LeapSecondData(leap_seconds, valid_until, last_updated)

Parameters

- **leap_seconds** (*List[LeapSecondInfo]*) –
- **valid_until** (*Optional[datetime]*) –
- **last_updated** (*Optional[datetime]*) –

Return type
LeapSecondData

classmethod from_data(*data*, *check_hash=True*)

Retrieve the leap second list from local data

Parameters

- **data** (*Union[bytes, str]*) – Data to parse as a leap second list
- **check_hash** (*bool*) – Whether to check the embedded hash

Return type
LeapSecondData

classmethod from_file(*filename='/usr/share/zoneinfo/leap-seconds.list'*, *check_hash=True*)

Retrieve the leap second list from a local file.

Parameters

- **filename** (*str*) – Local filename to read leap second data from. The default is the standard location for the file on Debian systems.
- **check_hash** (*bool*) – Whether to check the embedded hash

Return type
LeapSecondData

classmethod from_open_file(*open_file*, *check_hash=True*)

Retrieve the leap second list from an open file-like object

Parameters

- **open_file** (*BinaryIO*) – Binary IO object containing the leap second list
- **check_hash** (*bool*) – Whether to check the embedded hash

Return type
LeapSecondData

classmethod from_standard_source(*when=None*, *check_hash=True*)

Get the list of leap seconds from a standard source.

Parameters

- **when** (*Optional[datetime]*) – Check that the data is valid for this moment
- **check_hash** (*bool*) – Whether to check the embedded hash

Return type
LeapSecondData

Using a list of standard sources, including network sources, find a leap-second.list data valid for the given timestamp, or the current time (if unspecified)

classmethod from_url(*url='https://www.ietf.org/timezones/data/leap-seconds.list'*, *check_hash=True*)

Retrieve the leap second list from a local file

Parameters

- **filename** – URL to read leap second data from. The default is maintained by the IETF
- **check_hash** (*bool*) – Whether to check the embedded hash
- **url** (*str*) –

Return type
Optional[LeapSecondData]

is_leap_second(*when*, *check_validity=True*)

Return True if the given timestamp is the leap second.

Parameters

- **when** (*datetime*) – Moment in time to check. If naive, it is assumed to be in UTC.
- **check_validity** (*bool*) – Check whether the database is valid for the given moment

Return type
bool

For a TAI timestamp, it returns True for the leap second (the one that would be shown as :60 in UTC). For a UTC timestamp, it returns True for the :59 second if `fold`, since the :60 second cannot be represented.

tai_offset(*when*, *check_validity=True*)

For a given datetime, return the TAI-UTC offset

Parameters

- **when** (*datetime*) – Moment in time to find offset for

- **check_validity** (*bool*) – Check whether the database is valid for the given moment

Return type
timedelta

For times before the first leap second, a zero offset is returned. For times after the end of the file's validity, an exception is raised unless *check_validity=False* is passed. In this case, it will return the offset of the last list entry.

`tai_to_utc`(*when*, *check_validity=True*)

Convert the given datetime object to UTC

For a leap second, the `fold` property of the returned time is True.

Parameters

- **when** (*datetime*) – Moment in time to convert. If not naive, its `tzinfo` must be *tai*.
- **check_validity** (*bool*) – Check whether the database is valid for the given moment

Return type
datetime

`to_tai`(*when*, *check_validity=True*)

Convert the given datetime object to TAI.

Parameters

- **when** (*datetime*) – Moment in time to convert. If naive, it is assumed to be in UTC.
- **check_validity** (*bool*) – Check whether the database is valid for the given moment

Return type
datetime

Naive timestamps are assumed to be UTC. A TAI timestamp is returned unchanged.

`valid`(*when=None*)

Return True if the data is valid at given datetime (or the current moment, if None is passed)

Parameters
when (*Optional[datetime]*) – Moment to check for validity

Return type
bool

`class` leapseconddata.**LeapSecondInfo**

LeapSecondInfo(start, tai)

`static __new__`(*_cls*, *start*, *tai*)

Create new instance of LeapSecondInfo(start, tai)

`property start`

The UTC timestamp just after the insertion of the leap second.

The leap second is actually the 60th second of the previous minute

`property tai`

The new TAI-UTC offset. Positive numbers indicate that TAI is ahead of UTC

`exception` leapseconddata.**ValidityError**

The leap second information is not valid for the given timestamp

PYTHON MODULE INDEX

INDEX

Symbols

F

I

L

M

S

T

V