
pyiso Documentation

Release 0.3

Anna Schneider

May 23, 2017

1	Introduction	3
1.1	What's an ISO?	3
1.2	What's included	3
2	Installation	5
2.1	Install	5
2.2	Uninstall	5
3	Configuration	7
3.1	Accounts	7
3.2	Logging and debug	7
4	Usage	9
4.1	Clients	9
4.2	Tasks	11
5	Options	13
6	Contributing	15
6.1	For developers	15
6.2	For data users	15
6.3	For project admins	15
6.4	Legal things	16
7	Supporting	17
8	Indices and tables	19

Contents:

Introduction

Pyiso provides Python client libraries for ISO and other power grid data sources. It powers the [WattTime Impact API](#), among other things.

What’s an ISO?

Electricity markets are operated by “balancing authorities,” which manage supply and demand for a given service area. The bigger balancing authorities, called Independent Services Operators and Regional Transmission Organizations (ISOs/RTOs, or simply ISOs), together cover about 2/3 of US electricity consumers.

ISOs are required to provide real-time data about electricity market operations, but choose to do so in a wide variety of unstandardized, inconvenient formats. Some smaller balancing authorities provide data too.

What’s included

Pyiso makes it easier to collect data from ISOs and other balancing authorities by providing a uniform Python interface to each data stream. See the [Usage](#) page for instructions on how to get started.

Specifically, here are the included balancing authorities and their respective data sources:

For European data, you also need to specify a “control area”. The available control areas are:

control area abbrev.	control area country/provider
AL	Albania
AT	Austria
BA	Bosnia and Herzegovina
BE	Belgium
BG	Bulgaria
CH	Switzerland
CY	Cyprus
CZ	Czech Republic
DE(50HzT)	Germany (50 HzT)
DE(Amprion)	Germany (Amprion)
DE(TenneT GER)	Germany (TenneT)
DE(TransnetBW)	Germany (Transnet)
DK	Denmark
EE	Estonia
ES	Spain
Continued on next page	

Table 1.1 – continued from previous page

control area abbrev.	control area country/provider
FI	Finland
FR	France
GR	Greece
HR	Croatia
HU	Hungary
IE	Ireland
IT	Italy
LT	Lithuania
LU	Luxembourg
LV	Latvia
MD	Moldavia
ME	Montenegro
MK	Macedonia
MT	Malta
NIE	UK (NIE)
NL	Netherlands
NO	Norway
National Grid	UK (National Grid)
PL	Poland
PL-CZ	Czech Republic/Poland
PT	Portugal
RO	Romania
RS	Serbia
RU	Russia
RU-KGD	Russia (KGD)
SE	Sweden
SI	Slovenia
SK	Slovakia
TR	Turkey
UA	Ukraine
UA-WEPS	Ukraine (WEPS)

Installation

Install

Pyiso is available on [PyPI](#) and on [GitHub](#).

For users, the easiest way to get pyiso is with pip:

```
pip install pyiso
```

For developers, you can get the source from [GitHub](#) or [PyPI](#), then:

```
cd pyiso
python setup.py install
```

Pyiso depends on pandas so be prepared for a large install.

Windows Users: If you are unable to setup pyiso due to issues with installing or using numpy, a dependent package of pyiso, try installing a precompiled version of numpy found here: <http://www.lfd.uci.edu/~gohlke/pythonlibs/>

Uninstall

To uninstall:

```
pip uninstall pyiso
```

Configuration

Accounts

ISONE and the EU each require a username and password to collect data. You can register for an ISONE account here (<http://www.iso-ne.com/participate/applications-status-changes/access-software-systems#data-feeds>) and an EU ENTSOe account here (<https://transparency.entsoe.eu/>).

Then, set your usernames and passwords as environment variables:

```
export ISONE_USERNAME=myusername1
export ISONE_PASSWORD=mysecret1
export ENTSOe_USERNAME=myusername2
export ENTSOe_PASSWORD=mypassword2
```

All other ISOs allow unauthenticated users to collect data, so no other credentials are needed.

Logging and debug

By default, logging occurs at the INFO level. If you want to change this, you can set the *LOG_LEVEL* environment variable to the integer associated with the desired log level. For instance, ERROR is 40 and DEBUG is 10.

You can also turn on DEBUG level logging by setting the *DEBUG* environment variable to a truthy value. This setting will additionally enable caching during testing, which will significantly speed up the test suite.

Usage

There are two main ways to use `pyiso`: via the client objects, or via celery tasks. The client approach is preferred for scripted data analysis. The task approach enables asynchronous or periodic data collection and is in use at the [WattTime Impact API](#).

Clients

First, create a client using the `client_factory(ba_name)` function. `ba_name` should be taken from this list of abbreviated names for available balancing authorities listed on the [Introduction](#) page. For example:

```
>>> from pyiso import client_factory
>>> isone = client_factory('ISONE')
```

Requests made to external data sources will automatically time out after 20 seconds. To change this value, add a keyword argument in the constructor:

```
>>> isone = client_factory('ISONE', timeout_seconds=60)
```

Each client returned by `client_factory` is derived from `BaseClient` and provides one or more of the following methods (see also [Options](#)):

```
BaseClient.get_generation(latest=False, yesterday=False, start_at=False, end_at=False,
                          **kwargs)
```

Scrape and parse generation fuel mix data.

Parameters

- **latest** (*bool*) – If True, only get the generation mix at the one most recent available time point. Available for all regions.
- **yesterday** (*bool*) – If True, get the generation mix for every time point yesterday. Not available for all regions.
- **start_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be greater than or equal to this value. If using, must provide both `start_at` and `end_at` parameters. Not available for all regions.
- **end_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be less than or equal to this value. If using, must provide both `start_at` and `end_at` parameters. Not available for all regions.

Returns List of dicts, each with keys [ba_name, timestamp, freq, market, fuel_name, gen_MW]. Timestamps are in UTC.

Return type list

BaseClient.**get_load** (latest=False, yesterday=False, start_at=False, end_at=False, **kwargs)
Scrape and parse load data.

Parameters

- **latest** (*bool*) – If True, only get the load at the one most recent available time point. Available for all regions.
- **yesterday** (*bool*) – If True, get the load for every time point yesterday. Not available for all regions.
- **start_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be greater than or equal to this value. If using, must provide both *start_at* and *end_at* parameters. Not available for all regions.
- **end_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be less than or equal to this value. If using, must provide both *start_at* and *end_at* parameters. Not available for all regions.

Returns List of dicts, each with keys [ba_name, timestamp, freq, market, load_MW]. Timestamps are in UTC.

Return type list

BaseClient.**get_trade** (latest=False, yesterday=False, start_at=False, end_at=False, **kwargs)
Scrape and parse import/export data. Value is net export (export - import), can be positive or negative.

Parameters

- **latest** (*bool*) – If True, only get the trade at the one most recent available time point. Available for all regions.
- **yesterday** (*bool*) – If True, get the trade for every time point yesterday. Not available for all regions.
- **start_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be greater than or equal to this value. If using, must provide both *start_at* and *end_at* parameters. Not available for all regions.
- **end_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be less than or equal to this value. If using, must provide both *start_at* and *end_at* parameters. Not available for all regions.

Returns List of dicts, each with keys [ba_name, timestamp, freq, market, net_exp_MW]. Timestamps are in UTC.

Return type list

BaseClient.**get_lmp** (latest=False, yesterday=False, start_at=False, end_at=False, **kwargs)
Scrape and parse location marginal price data. To request a specific LMP node, include kwarg *node_id*.

Parameters

- **latest** (*bool*) – If True, only get LMP at the one most recent available time point. Available for all regions.

- **yesterday** (*bool*) – If True, get LMP for every time point yesterday. Not available for all regions.
- **start_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be greater than or equal to this value. If using, must provide both `start_at` and `end_at` parameters. Not available for all regions.
- **end_at** (*datetime*) – If the datetime is naive, it is assumed to be in the timezone of the Balancing Authority. The timestamp of all returned data points will be less than or equal to this value. If using, must provide both `start_at` and `end_at` parameters. Not available for all regions.

Returns List of dicts, each with keys [`ba_name`, `timestamp`, `freq`, `market`, `lmp`, `lmp_type`]. Timestamps are in UTC.

Return type list

The lists returned by clients are conveniently structured for import into other data structures like `pandas.DataFrame`:

```
>>> import pandas as pd
>>> data = isone.get_generation(latest=True)
>>> df = pd.DataFrame(data)
>>> print df
  ba_name freq fuel_name  gen_MW market  timestamp
0  ISONE  n/a    coal    1170.0  RT5M  2014-03-29 20:40:27+00:00
1  ISONE  n/a    hydro    813.8  RT5M  2014-03-29 20:40:27+00:00
2  ISONE  n/a    natgas  4815.7  RT5M  2014-03-29 20:40:27+00:00
3  ISONE  n/a    nuclear  4618.8  RT5M  2014-03-29 20:40:27+00:00
4  ISONE  n/a    biogas    29.5  RT5M  2014-03-29 20:40:27+00:00
5  ISONE  n/a    refuse   428.6  RT5M  2014-03-29 20:40:27+00:00
6  ISONE  n/a    wind     85.8  RT5M  2014-03-29 20:40:27+00:00
7  ISONE  n/a    biomass  434.3  RT5M  2014-03-29 20:40:27+00:00
```

Happy data analysis!

Tasks

If you have a `celery` environment set up, you can use the tasks provided in the `pyiso.tasks` module. There is one task for each of the client's `get_*` methods that implements a thin wrapper around that method. The call signatures match those of the corresponding client methods, except that the `ba_name` is a required first argument. For example, to get the latest ISONE generation mix data every 10 minutes, add this to your `celerybeat` schedule:

```
CELERYBEAT_SCHEDULE = {
    'get-isone-genmix-latest' : {
        'task': 'pyiso.tasks.get_generation',
        'schedule': crontab(minute='*/10'),
        'args': ['ISONE'],
        'kwargs': {'latest': True},
    }
}
```

In practice, you will want to chain these tasks with something that captures and processes their output.

Options

Not all date range options are available for all methods in all regions. Here's what's available now:

method	latest	start_at and end_at pair	yesterday	forecast ok
BPA.get_generation	yes	yes	no	no
BPA.get_load	yes	yes	no	no
CAISO.get_generation	yes	yes	yes	yes
CAISO.get_load	yes	yes	yes	yes
CAISO.get_trade	yes	yes	yes	yes
CAISO.get_lmp	yes	yes	yes	yes
ERCOT.get_generation	yes	no	no	no
ERCOT.get_lmp	yes	yes	no	yes
ERCOT.get_load	yes	yes	no	yes
EU.get_load	yes	yes	no	yes
ISONE.get_generation	yes	yes	no	no
ISONE.get_lmp	yes	yes	yes	yes
ISONE.get_load	yes	yes	no	yes
MISO.get_generation	yes	yes	no	yes
MISO.get_load	yes	yes	no	yes
MISO.get_trade	no	yes	no	yes
MISO.get_lmp	yes	yes	no	yes
NVEnergy.get_load	yes	yes	no	yes
NYISO.get_generation	yes	yes	no	no
NYISO.get_load	yes	yes	no	yes
NYISO.get_lmp	yes	yes	no	yes
NYISO.get_trade	yes	yes	no	no
PJM.get_generation	yes	no	no	no
PJM.get_load	yes	yes	no	yes
PJM.get_trade	yes	no	no	no
PJM.get_lmp	yes	yes	no	no
SVERI.get_generation	yes	yes	no	no
SVERI.get_load	yes	yes	no	no

Contributing

Right now, `pyiso` only has interfaces for collecting a small subset of the interesting electricity data that the ISOs provide. You can help by adding more! Please create an issue on [github](#) if you have questions about any of this.

For developers

When you're ready to get started coding:

- fork the [repo](#)
- install in development mode: `python setup.py develop`
- run the tests: `python setup.py test` (or `python setup.py test -s tests.test_some_file.TestSomeClass.test_some_method` to run a specific subset of the tests)
- add tests to the `tests` directory and code to the `pyiso` directory, following the conventions that you see in the existing code
- add docs to the `docs/source` directory
- add a note to the changelog in `README.md`
- send a pull request
- sign the CLA at <https://www.clahub.com/agreements/WattTime/pyiso> (see below)

For data users

Found a bug, or know of a data source that you think `pyiso` should include? Please add an issue to [github](#). Ideas of new balancing authorities (anywhere in the world) and of new data streams from ISOs we already support are both very welcome.

For project admins

Before making a release, check that these are true in the master branch of the GitHub repo:

- the changelog in `README.md` includes all changes since the last release
- test coverage is good and the tests pass locally and on Travis

- changes are reflected in the docs in *docs/source*
- the version number is upgraded in *pyiso/__init__.py*

To make a release, run these commands (replacing 0.x.y with the correct version number):

```
git checkout master
git pull origin master
git tag v0.x.y
git push origin master --tags
python setup.py sdist upload
```

Releasing via twine:

```
python setup.py sdist
twine upload dist/pyiso-VERSION.tar.gz
```

Legal things

Because we use pyiso as the base for our other software products, we ask that contributors sign the following Contributor License Agreement. If you have any questions, or concerns, please drop us a line on Github.

You and WattTime, Corp, a california non-profit corporation, hereby accept and agree to the following terms and conditions:

Your “Contributions” means all of your past, present and future contributions of object code, source code and documentation to pyiso however submitted to pyiso, excluding any submissions that are conspicuously marked or otherwise designated in writing by You as “Not a Contribution.”

You hereby grant to the WattTime, Corp a non-exclusive, irrevocable, worldwide, no-charge, transferable copyright license to use, execute, prepare derivative works of, and distribute (internally and externally, in object code and, if included in your Contributions, source code form) your Contributions. Except for the rights granted to the WattTime, Corp in this paragraph, You reserve all right, title and interest in and to your Contributions.

You represent that you are legally entitled to grant the above license. If your employer(s) have rights to intellectual property that you create, you represent that you have received permission to make the Contributions on behalf of that employer, or that your employer has waived such rights for your Contributions to pyiso.

You represent that, except as disclosed in your Contribution submission(s), each of your Contributions is your original creation. You represent that your Contribution submissions(s) included complete details of any license or other restriction (including, but not limited to, related patents and trademarks) associated with any part of your Contributions(including a copy of any applicable license agreement). You agree to notify WattTime, Corp of any facts or circumstances of which you become aware that would make Your representations in the Agreement inaccurate in any respect.

You are not expected to provide support for your Contributions, except to the extent you desire to provide support. You may provide support for free, for a fee, or not at all. Your Contributions are provided as-is, with all faults, defects and errors, and without any warranty of any kind (either express or implied) including, without limitation, any implied warranty of merchantability and fitness for a particular purpose and any warranty of non-infringement.

To get started, sign the Contributor License Agreement.

Supporting

Pyiso is an open source project maintained by [WattTime](#), a nonprofit that develops software standards to reduce power grid pollution and enable new kinds of clean energy choices.

We've spent more than 1000 developer-hours building pyiso, keeping it up-to-date with evolving data sources, and adding features requested by the community. As the foundation of our internal data pipeline, it makes our work easier every day. And we've made it free and open source because we want to make open energy data access a bit easier for other researchers, engineers, and citizens too!

Want to chip in and support pyiso? You or your company can make a tax-deductible donation to WattTime [here](#). Every dollar helps us help you! We also have corporate sponsorship opportunities available; [get in touch](#) if you're interested.

Another great way to support pyiso is to send us a quick [thank-you note](#). Your testimonials help us raise money from other folks, so it really does make a difference. Thanks bunches!

Indices and tables

- *genindex*
- *modindex*
- *search*