WattTime API Client Documentation

Release 0.1

WattTime

September 10, 2015

Contents

This package provides a Python client SDK for the WattTime API (http://api.watttime.org/). See the API site for detailed info about its purpose and usage.

Contents:

CHAPTER 1

Installation

This package is available on GitHub and PyPI.

For users, install using `pip`:

```
pip install watttime_client
```

For developers, you can get the source from GitHub, then:

```
cd watttime-python-client
python setup.py install
```

This depends on pandas, so be prepared for a large install.

CHAPTER 2

Getting a token

The API requires a valid token. To register for a token, sign up for a user account at http://api.watttime.org/accounts/register/. Your API token will be available at http://api.watttime.org/accounts/token/.

The default token permissions aren't sufficient to allow access to the marginal carbon emissions data, which the client library provides. To request an account upgrade, email the WattTime Dev team at <dev@watttime.org> with a quick explanation of what you'll use the marginal carbon data for.

Once your account has been upgraded, the last step is to set the token as an environment variable. This is required for running the test suite, and it's good practice anyway. Type this line in your bash shell, or add it to your .bashrc, replacing the dummy token string with your actual token:

```
export WATTTIME_API_TOKEN=abcdef0123456abcdef0123456
```

CHAPTER 3

Usage

3.1 Set up the token

If you have set your API token as an environment variable (see Getting a token), then you should import it from the environment:

```
>>> import os
>>> mytoken = os.environ.get('WATTTIME_API_TOKEN')
```

3.2 Create a client

Import the client class, and create an instance with your token:

```
>>> from watttime_client.client import WattTimeAPI
>>> client = WattTimeAPI(token=mytoken)
```

You can reuse this client for multiple requests to take advantage of internal caching, which will make your code faster and reduce load on the WattTime server.

3.3 Get marginal carbon data

There are two main ways to get data.

If you want the marginal carbon value at just one time, use `get_impact_at`. This method takes a timezone-aware datetime, the name of a balancing authority, and the name of a market (real-time by default). It returns the numerical value of the marginal carbon emissions due to electricity usage at that place and time, in lb/MWh:

```
>>> from datetime import datetime
>>> import pytz
>>> timestamp = pytz.utc.localize(datetime(2015, 6, 1, 12, 30))
>>> value = client.get_impact_at(timestamp, 'CAISO')
>>> print value
909.2
```

If you want the marginal carbon value at a range of times, use `get_impact_between`. This method takes a pair of timezone-aware datetimes for the start and end times, the interval length between readings (in minutes), the name of a balancing authority, and the name of a market (real-time by default). It returns a pandas Series containing the marginal carbon emissions values for electricity usage at that place and times, in lb/MWh:

```
>>> start_time = pytz.utc.localize(datetime(2015, 6, 1, 12, 30))
>>> end_time = pytz.utc.localize(datetime(2015, 6, 1, 18, 30))
>>> interval_min = 5
>>> data = client.get_impact_between(start_time, end_time, interval_min, 'CAISO')
>>> print data.head()
2015-06-01 12:30:00+00:00    909.2
2015-06-01 12:35:00+00:00    909.2
2015-06-01 12:40:00+00:00    920.5
2015-06-01 12:45:00+00:00    920.5
2015-06-01 12:50:00+00:00    920.5
Freq: 5T, dtype: float64
```

3.4 Analyzing data

Once you have the data in pandas, you can use its full power in your data analysis. Or if you'd rather export it, use any of the pandas I/O tools. For instance, to export to csv:

```
>>> output_file_name = 'awesome_carbon_data.csv'
>>> data.to_csv(output_file_name)
```

CHAPTER 4

Testing

Tests are located in the `tests` directory. To run the test suite, simply run on the command line:

```
./runtests.py
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search