# srpenergy Documentation

*Release 1.3.6*

**Brig Lamoreaux**

**Feb 09, 2022**

# CONTENTS

The `srpenergy` module is an unofficial Python module for interacting with Srp Energy data.

- Development: https://github.com/lamoreauxlab/srpenergy-api-client-python/
- Documentation: https://srpenergy-api-client-python.readthedocs.io/

Srp provides an hourly energy usage report for their customers. The `srpenergy` module fetches the data found via the api.

The data returned from the hourly url `https://myaccount.srpnet.com/myaccountapi/api/usage/hourlydetail?billaccount=<code>&beginDate=<MM-DD-YYYY>&endDate=<MM-DD-YYYY>`


```
{   "hourlyConsumptionList": [],
    "hourlyGenerationList": [],
    "hourlyReceivedList": [],
    "hourlyUsageList":[{
            "date": "2019-10-09T00:00:00",
            "hour": "2019-10-09T00:00:00",
            "onPeakKwh": 0.0,
            "offPeakKwh": 0.0,
            "shoulderKwh": 0.0,
            "superOffPeakKwh": 0.0,
            "totalKwh": 0.4,
            "onPeakCost": 0.0,
            "offPeakCost": 0.0,
            "shoulderCost": 0.0,
            "superOffPeakCost": 0.0,
            "totalCost": 0.08
        }
    ],
    "demandList":[]
}
```


**Note:** Time of use customers do not receive a `totalKwh` or `totalCost` from the api. These values are calculated from `onPeakKwh`, `offPeakKwh`, and the fomula defined by the SRP TOU price plan sheet

EZ3 customers show 0.0 for `totalKwh` and `totalCost`. Those values are split between `onPeak`, `offPeak`, `shoulder`, and `superOffPeak`.

CHAPTER

ONE

# INSTALLING

It is distributed on PyPI and can be installed with pip:

```
pip install srpenergy
```

CHAPTER

# TWO

# USE

```
from datetime import datetime, timedelta
from srpenergy.client import SrpEnergyClient

accountid = 'your account id'
username = 'your username'
password = 'your password'
end_date = datetime.now()
start_date = datetime.now() - timedelta(days=2)

client = SrpEnergyClient(accountid, username, password)
usage = client.usage(start_date, end_date)

date, hour, isodate, kwh, cost = usage[0]
```

For Time of use plans pass in the argument *is_tou*

```
from datetime import datetime, timedelta
from srpenergy.client import SrpEnergyClient

accountid = 'your account id'
username = 'your username'
password = 'your password'
end_date = datetime.now()
start_date = datetime.now() - timedelta(days=2)

client = SrpEnergyClient(accountid, username, password)
usage = client.usage(start_date, end_date, True)

date, hour, isodate, kwh, cost = usage[0]
```

CHAPTER

# THREE

# DEVELOPMENT

## 3.1 Configure Dev Environment

This section will configure your computer to develop, test, and debug the project.

## 3.2 Style Guidelines

This project enforces quite strict PEP8 and PEP257 (Docstring Conventions) compliance on all code submitted.

We use Black for uncompromised code formatting.

Summary of the most relevant points:

- Comments should be full sentences and end with a period.
- Imports should be ordered.
- Constants and the content of lists and dictionaries should be in alphabetical order.
- It is advisable to adjust IDE or editor settings to match those requirements.

## 3.3 Ordering of imports

Instead of order the imports manually, use isort.

```
pip3 install isort
isort .
```

## 3.4 Use new style string formatting

Prefer f-strings over `%` or `str.format`.

```
#New
f"{some_value} {some_other_value}"
# Old, wrong
"{} {}".format("New", "style")
"%s %s" % ("Old", "style")
```

One exception is for logging which uses the percentage formatting. This is to avoid formatting the log message when it is suppressed.

```
_LOGGER.info("Can't connect to the webservice %s at %s", string1, string2)
```

## 3.5 Testing

As it states in the *Style Guidelines* section all code is checked to verify the following:

- All the unit tests pass
- All code passes the checks from the linting tools

Install the test dependencies into your Python environment:

```
pip3 install -r requirements_test.txt
```

Now that you have all test dependencies installed, you can run tests on the project:

```
isort .
codespell  --skip="./.*,*.csv,*.json,*.pyc,./docs/_build/*,./htmlcov/*"
black setup.py srpenergy tests
flake8 setup.py srpenergy tests
pylint setup.py srpenergy tests
pydocstyle setup.py srpenergy tests
rstcheck README.rst
python -m pytest tests
python -m pytest --cov-report term-missing --cov=srpenergy tests
```

## 3.6 Building Docs

Build the documentation locally with

```
cd docs
python -m sphinx -T -b html -d _build/doctrees -D language=en . _build/html
```

## 3.7 Run Git Pre-commit

Run pre-commit hooks on the repository.

```
# Run all hooks
pre-commit run --all-files

# Run a specific hook
pre-commit run hook_id
```

## 3.8 Package and Deploy

After a successful build, packageing and deploying will:

- Bump Version
- Tag version in git
- Create Release in git
- Release to pypi

### 3.8.1 Bump Version

Change the version in the following files:

- srpenergy/__init__.py
- docs/conf.py

### 3.8.2 Tag Version

Commit, tag, and push the new version

```
git commit -m "Bump version"
git tag -a 1.3.1 -m "1.3.1"
git push --tags
```

### 3.8.3 Create Release

- Create a new Release
- Name the Release the same as the tag name
- Auto-generate release notes.

### 3.8.4 Release to pypi

Upgrade to the latest version of setuptools and create package and test

```
python -m pip install --user --upgrade setuptools wheel # Get latest version
python setup.py sdist bdist_wheel
twine check dist/*
```

Upload the package to test first

```
python -m twine upload --repository testpypi dist/*
```

Check that package looks ok. After testing, upload to the main repository

```
python -m twine upload dist/*
```

CHAPTER

FOUR

# DOCUMENTATION

## 4.1 Client Module

Client module.

This module houses the main class used to fetch energy usage.

**class** srpenergy.client.**SrpEnergyClient**(*accountid*, *username*, *password*)

SrpEnergyClient(accountid, username, password).

Client used to fetch srp energy usage.

**Parameters**

**accountid** [string] An srp account id.

**username: string** An srp account username.

**password: string** An srp account password

### Methods

| | |
|---|---|
| **validate()** | Validate user credentials. |
| **usage(startdate, enddate)** | Get the usage for a given date range. |

**usage**(*startdate*, *enddate*, *is_tou=False*)

Get the energy usage for a given date range.

**Parameters**

**startdate** [datetime] the start date

**enddate** [datetime] the end date

**is_tou** [bool] indicate if usage is a time of use plan

**Returns**

**list of tuple** In the form of (datepart, timepart, isotime, kw, cost)

**Raises**

**ValueError** If `startdate` or `enddate` are not datetime, or if `startdate` is greater than `enddate`, or if `startdate` is greater than now.

### Examples

Get the hourly usage for a given day.

```
>>> from srpenergy.client import SrpEnergyClient
>>> accountid = 'your account id'
>>> username = 'your username'
>>> password = 'your password'
>>> client = SrpEnergyClient(accountid, username, password)
>>> start_date = datetime(2018, 9, 19, 0, 0, 0)
>>> end_date = datetime(2018, 9, 19, 23, 0, 0)
>>> usage = client.usage(start_date, end_date)
>>> print(usage)
[
('9/19/2018', '12:00 AM', '2018-09-19T00:00:00-7:00', '1.2', '0.17'),
('9/19/2018', '1:00 AM', '2018-09-19T01:00:00-7:00', '2.1', '0.30'),
('9/19/2018', '2:00 AM', '2018-09-19T02:00:00-7:00', '1.5', '0.23'),
...
('9/19/2018', '9:00 PM', '2018-09-19T21:00:00-7:00', '1.2', '0.19'),
('9/19/2018', '10:00 PM', '2018-09-19T22:00:00-7:00', '1.1', '0.18'),
('9/19/2018', '11:00 PM', '2018-09-19T23:00:00-7:00', '0.4', '0.09')
]
```

**validate**()
Validate user credentials.

**Returns**

**bool**

### Examples

Validate credentials.

```
>>> from srpenergy.client import SrpEnergyClient
>>>
>>> accountid = 'your account id'
>>> username = 'your username'
>>> password = 'your password'
>>> client = SrpEnergyClient(accountid, username, password)
>>>
>>> valid = client.validate()
>>> print(valid)
True
```

srpenergy.client.**get_pretty_date**(*date_part*)
Return a formatted date from an iso date.

srpenergy.client.**get_pretty_time**(*date_part*)
Return a formatted time from an iso date.

srpenergy.client.**get_rate**(*str_usage_time*)
Return the time of use pricing for the given time.

From the SRP website peak times: Winter Nov-Apr (5am-9am, 5pm-9pm) 9.51 peak, 6.91 offpeak Summer May-Oct (2pm-8pm) 20.94 peak, 7.27 offpeak Summer Peak Jul,Aug (2pm-8pm) 24.09, 7.3

Higher on-peak prices are in effect Monday through Friday only during the hours shown. Lower off-peak prices are in effect all other weekday hours, weekends and six observed holidays: New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day and Christmas Day.

see https://srpnet.com/prices/pdfx/April2015/E-26.pdf

## 4.2 Development Documentation

Build module with:

```
$ python setup.py bdist_wheel --universal
```

Publish to pypi test:

```
$ twine upload --repository-url https://test.pypi.org/srpenergy/ dist/*
```

Publish to pypi:

```
$ twine upload dist/*
```

## 4.3 API Documentation

This is an unofficial documentation of the SRP Energy Usage api. Srp provides an hourly energy usage report for their customers.

There are three steps to fetch the data:

1. Log into the site with user credentials.
2. Request an xsrf-token token.
3. Request the usage data.

### 4.3.1 Log into site

Post x-www-form-urlencoded parameters to the endpoint.

**username** The username for the account.

**password** The password for the account.

```
# Post to
https://myaccount.srpnet.com/myaccountapi/api/login/authorize

# Include x-www-form-urlencoded parameters
# username=<username>
# password=<password>
```

### 4.3.2 Fetch Xsrf token

Then fetch the xsrf-token by calling:

```
# Get
https://myaccount.srpnet.com/myaccountapi/api/login/authorize

# Save the result of the xsrf-token cookie
```

### 4.3.3 Call Usage data

Finally call the usage endpoint ad include the following:

**code** billing account number

**str_startdate** start date of billing usage in the format MM-DD-YYYY

**str_enddate** end date of billing usage in the format MM-DD-YYYY

**xsrf-token** the xsrf-token used in the header

```
https://myaccount.srpnet.com/myaccountapi/api/usage/hourlydetail?billaccount="
                + <code>
                + "&beginDate="
                + <str_startdate>
                + "&endDate="
                + str_enddate,

# Headers
# "x-xsrf-token": xsrf_token_value
```

### 4.3.4 Results

```
{   "hourlyConsumptionList": [],
    "hourlyGenerationList": [],
    "hourlyReceivedList": [],
    "hourlyUsageList":[{
            "date": "2019-10-09T00:00:00",
            "hour": "2019-10-09T00:00:00",
            "onPeakKwh": 0.0,
            "offPeakKwh": 0.0,
            "shoulderKwh": 0.0,
            "superOffPeakKwh": 0.0,
            "totalKwh": 0.4,
            "onPeakCost": 0.0,
            "offPeakCost": 0.0,
            "shoulderCost": 0.0,
            "superOffPeakCost": 0.0,
            "totalCost": 0.08
        }
    ],
    "demandList":[]
}
```

**Note:** Time of use customers do not receive a `totalKwh` or `totalCost` from the api. These values are calculated from `onPeakKwh`, `offPeakKwh`, and the fomula defined by the SRP TOU price plan sheet

EZ3 customers show 0.0 for `totalKwh` and `totalCost`. The values are split between `onPeak`, `offPeak`, `shoulder`, and `superOffPeak`.

# PYTHON MODULE INDEX

## S

# INDEX