
BismuthClient Documentation

Release 0.0.1

EggdraSyl Bismuth Foundation

Nov 20, 2018

Contents:

1	BismuthClient	1
1.1	Features	1
1.2	Credits	1
2	Installation	3
2.1	Stable release	3
2.2	From sources	3
3	Usage	5
4	bismuthclient	7
4.1	bismuthclient package	7
5	Contributing	11
5.1	Types of Contributions	11
5.2	Get Started!	12
5.3	Pull Request Guidelines	13
5.4	Tips	13
5.5	Deploying	13
6	Credits	15
6.1	Development Lead	15
6.2	Contributors	15
7	History	17
7.1	0.0.1 (2018-11-17)	17
7.2	0.0.2 (2018-11-20)	17
8	Indices and tables	19
	Python Module Index	21

CHAPTER 1

BismuthClient

Bismuth python library to be used by dApps and clients.

- Free software: MIT license
- Documentation: <https://bismuthclient.readthedocs.io>.

1.1 Features

- TODO

1.2 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

2.1 Stable release

To install BismuthClient, run this command in your terminal:

```
$ pip install bismuthclient
```

This is the preferred method to install BismuthClient, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for BismuthClient can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/EggPool/bismuthclient
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/EggPool/bismuthclient/tarball/master
```

Once you have a copy of the source, you can install it with:

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


CHAPTER 3

Usage

To use BismuthClient in a project:

```
import bismuthclient
```

Import the native API cclient class:

```
from bismuthclient import rpcconnections
```


4.1 bismuthclient package

4.1.1 Submodules

4.1.2 bismuthclient.async_client module

4.1.3 bismuthclient.bismuthapi module

Wrapper around the official Bismuth API from `api.bismuth.live`

WIP

```
bismuthclient.bismuthapi.get_wallet_servers_legacy(light_ip_list="", app_log=None)
```

Use different methods to return the best possible list of wallet servers, sorted ip:port

4.1.4 bismuthclient.bismuthclient module

4.1.5 bismuthclient.bismuthcrypto module

4.1.6 bismuthclient.bismuthformat module

Formatting helpers for Bismuth

```
class bismuthclient.bismuthformat.AmountFormatter(amount)
    Bases: object

    Format a BIS amount to string

    amount

    to_string(decimals=3, leading=0)
```

```
class bismuthclient.bismuthformat.TxFormatter (tx)
    Bases: object
    Formats a Bismuth Transaction
    to_json (for_display=False)
    to_string ()
    tx
```

4.1.7 bismuthclient.bismuthwallet module

4.1.8 bismuthclient.lwbench module

Wallet server benchmarking functions - ported from gui wallet, converted to use the native api class.

```
bismuthclient.lwbench.connectible (ipport)
    Helper: return True if the ip:port can be connected to, without sending any command.
```

```
bismuthclient.lwbench.convert_ip_port (ip, some_port)
    Helper: Get ip and port, but extract port from ip if ip was as ip:port
```

Parameters

- **ip** –
- **some_port** – default port

Returns (ip, port)

```
bismuthclient.lwbench.time_measure (light_ip_list, app_log=None)
    Measure answer time for the servers in light_ip list
```

Parameters

- **light_ip_list** – a list of “ip:port” wallet servers
- **app_log** – an optional logger

Returns A sorted light_ip list

4.1.9 bismuthclient.rpcconnections module

Bismuth default/legacy connection layer. Json over sockets This file is no more compatible with the Bismuth code, it's been converted to a class EggPool 2018

```
class bismuthclient.rpcconnections.Connection (ipport, verbose=False, raw=False)
    Bases: object
    Connection to a Bismuth Node. Handles auto reconnect when needed
    check_connection ()
        Check connection state and reconnect if needed.
    close ()
        Close the socket
    command (command, options=None)
        Sends a command and return it's raw result. options has to be a list. Each item of options will be sent separately. So If you ant to send a list, pass a list of list.
```

- `command_lock`
- `ipport`
- `last_activity`
- `raw`
- `sdef`
- `stats`
- `verbose`

4.1.10 Module contents

Top-level package for BismuthClient.

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at <https://github.com/EggPool/bismuthclient/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

5.1.4 Write Documentation

BismuthClient could always use more documentation, whether as part of the official BismuthClient docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/EggPool/bismuthclient/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *bismuthclient* for local development.

1. Fork the *bismuthclient* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/bismuthclient.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv bismuthclient
$ cd bismuthclient/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 bismuthclient tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/EggPool/bismuthclient/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_bismuthclient
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

6.1 Development Lead

- EggdraSyl from Bismuth Foundation <dev@eggpool.net>

6.2 Contributors

None yet. Why not be the first?

7.1 0.0.1 (2018-11-17)

- First release on PyPI.

7.2 0.0.2 (2018-11-20)

- Alpha release

CHAPTER 8

Indices and tables

- `genindex`
- `modindex`
- `search`

b

`bismuthclient`, 9
`bismuthclient.bismuthapi`, 7
`bismuthclient.bismuthformat`, 7
`bismuthclient.lwbench`, 8
`bismuthclient.rpcconnections`, 8

A

amount (bismuthclient.bismuthformat.AmountFormatter attribute), 7

AmountFormatter (class in bismuthclient.bismuthformat), 7

B

bismuthclient (module), 9

bismuthclient.bismuthapi (module), 7

bismuthclient.bismuthformat (module), 7

bismuthclient.lwbench (module), 8

bismuthclient.rpcconnections (module), 8

C

check_connection() (bismuthclient.rpcconnections.Connection method), 8

close() (bismuthclient.rpcconnections.Connection method), 8

command() (bismuthclient.rpcconnections.Connection method), 8

command_lock (bismuthclient.rpcconnections.Connection attribute), 8

connectible() (in module bismuthclient.lwbench), 8

Connection (class in bismuthclient.rpcconnections), 8

convert_ip_port() (in module bismuthclient.lwbench), 8

G

get_wallet_servers_legacy() (in module bismuthclient.bismuthapi), 7

I

ipport (bismuthclient.rpcconnections.Connection attribute), 9

L

last_activity (bismuthclient.rpcconnections.Connection attribute), 9

R

raw (bismuthclient.rpcconnections.Connection attribute), 9

S

sdef (bismuthclient.rpcconnections.Connection attribute), 9

stats (bismuthclient.rpcconnections.Connection attribute), 9

T

time_measure() (in module bismuthclient.lwbench), 8

to_json() (bismuthclient.bismuthformat.TxFormatter method), 8

to_string() (bismuthclient.bismuthformat.AmountFormatter method), 7

to_string() (bismuthclient.bismuthformat.TxFormatter method), 8

tx (bismuthclient.bismuthformat.TxFormatter attribute), 8

TxFormatter (class in bismuthclient.bismuthformat), 7

V

verbose (bismuthclient.rpcconnections.Connection attribute), 9