Object Blockchain Mapping

Release 0.0.27

Alexander Polishchuk

Jun 23, 2020

CONTENTS:

There are two layers in `OBM` architecture. The top-level is `obm.models`, that provide unified API and should be enough in most cases. The second one is low-level `obm.connectors`, that contains connectors with both non-unified and unified API for each node that supported by `obm`.

Note: This guide uses python built-in async REPL to show asynchronous API features. You can launch it using command `python -m asyncio` (Python 3.8 or higher).

CHAPTER

ONE

MODELS

1.1 Create a model

```
>>> import asyncio
>>> from obm import models
>>> btc = models.Node(
...     name="bitcoin-core",
...     rpc_port=18332,
...     rpc_username="testnet_user",
...     rpc_password="testnet_pass",
... )
>>> eth = models.Node(
...     name="geth",
...     rpc_port=8545,
... )
```

1.2 Create address

```
>>> await btc.create_address()
'2N1Gbn5dqQxxD443Se9moXBaafGLvKCweop'
>>> await eth.create_address()
'0x8a9c181caa4a1273e46a306309e806e2d61fc560'
```

1.3 Send transactions

```
>>> await btc.send_transaction(
...     amount=0.00001,
...     to_address='2NAmne8BsSXWbV5iStkVzL4vW7Z4F6a5o68',
...     subtract_fee_from_amount=True,
... )
{
    "txid": "cc8c9f7a86261fcb00d68b62073c740b8a0e14079d67e44fd726e0de2954c69a",
    "from_address": "2NAmne8BsSXWbV5iStkVzL4vW7Z4F6a5o68",
    "to_address": "2NAmne8BsSXWbV5iStkVzL4vW7Z4F6a5o68",
    "amount": Decimal("0.00000866"),
    "fee": Decimal("0.00000134"),
    "block_number": None,
    "category": "oneself",
```

(continues on next page)

(continued from previous page)

```
    "timestamp": 1588076404,
    "info": {...},
}
>>> await eth.send_transaction(
...     amount=0.00005,
...     from_address='0xe1082e71f1ced0efb0952edd23595e4f76840128',
...     to_address='0xb610de1be67b10c746afec8fe74ad14d97e34146',
...     subtract_fee_from_amount=True,
...     password="abc",
... )
{
    "txid": "0x4831820db0de1aad336c7a083b2504ad0b91eba293e5d7a6fa3bef49f660766c",
    "from_address": "0xe1082e71f1ced0efb0952edd23595e4f76840128",
    "to_address": "0xb610de1be67b10c746afec8fe74ad14d97e34146",
    "amount": Decimal("0.000029"),
    "fee": Decimal("0.000021"),
    "block_number": None,
    "category": "oneself",
    "timestamp": None,
    "info": {...},
}
```

1.4 Fetch transactions

```
>>> await btc.fetch_recent_transactions(limit=1)
[
    {
        "txid": "cc8c9f7a86261fcb00d68b62073c740b8a0e14079d67e44fd726e0de2954c69a",
        "from_address": "2NAmne8BsSXWbV5iStkVzL4vW7Z4F6a5o68",
        "to_address": "2NAmne8BsSXWbV5iStkVzL4vW7Z4F6a5o68",
        "amount": Decimal("0.00000866"),
        "fee": Decimal("0.00000134"),
        "block_number": 1722208,
        "category": "oneself",
        "timestamp": 1588076404,
        "info": {...},
    }
]
>>> await eth.fetch_recent_transactions(limit=1)
[
    {
        "txid": "0x4831820db0de1aad336c7a083b2504ad0b91eba293e5d7a6fa3bef49f660766c",
        "from_address": "0xe1082e71f1ced0efb0952edd23595e4f76840128",
        "to_address": "0xb610de1be67b10c746afec8fe74ad14d97e34146",
        "amount": Decimal("0.000029"),
        "fee": Decimal("0.000021"),
        "block_number": 6394779,
        "category": "oneself",
        "timestamp": None,
        "info": {...},
    }
]
```

CHAPTER

TWO

CONTRIBUTOR'S GUIDE

Welcome and thank you for your interest in contributing to the this open source project. This documentation aims to document how contributors and collaborators should work when using Git, GitHub and the development workflow. This Git workflow is inspired greatly by the QuantConnect Lean Contributors Guide.

2.1 Style Guide

The project was written following Google Python Style Guide and reviewers will be expecting to see code that follow it as well. But you should to use *black* instread of obsolete *yapf*. Please make sure, that your linter (pylint) and formatter (black) are using configs form repo's root.

2.2 Testing

All pull requests must be accompanied by units tests. If it is a new feature, the tests should highlight expected use cases as well as edge cases, if applicable. If it is a bugfix, there should be tests that expose the bug in question.

2.3 Brunching Strategy

The project following GitLab Flow strategy according to which: - `master` is the primary brunch - feature-brunches must branch off from `master` - feature-brunches must be merged back into `master`

2.4 Pull Requests

When you going to develop new feature or make some changes that will change the existing code, please create an issue to suggest changes. In other cases (bugfix, doc improvement and so on) you can just create pull request.

2.4.1 Work In Progress

You can use `WIP` prefix in PR's name if you wish to get immediate feedback, but in this case you should take care of readability of your work-in-progress code. Thus please push changes only when you have a working set of tests and code.

Good lack!

CHAPTER

THREE

AUTHORS

3.1 Maintainers

- Alexander Polishchuk

3.2 Contributors

- Evgeniy Yachmenyov (logo)

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search