uptick

Release 0.1

Fabian Schuh

Oct 12, 2020

Contents

uptick is a tool to interact with the BitShares network using Python 3 and python-bitshares.

- uptick's home is github.com/xeroc/uptick and
- python-bitshares's home is github.com/xeroc/python-bitshares and
- this documentation is available through ReadMyDocs and is hosted on uptick.rocks

CHAPTER 1

uptick.web - Graphical User Interface

(work in progress)

CHAPTER 2

Command Line Tool

The command line tool that is bundled with this package is called `uptick` and helps you

- deal with your funds
- trade
- manage your accounts

in the BitShares network. After installation, you can get the full list of features with:

```
$ uptick --help
```

CHAPTER 3

General

3.1 Installation

3.1.1 Install with *pip*:

```
pip3 install uptick
```

3.1.2 Manual installation:

```
git clone https://github.com/xeroc/uptick
cd uptick
python3 setup.py install --user
```

3.1.3 Upgrade

```
pip3 install uptick --user --upgrade
```

3.2 Contributing to python-bitshares

We welcome your contributions to our project.

3.2.1 Flow

This project makes heavy use of git flow. If you are not familiar with it, then the most important thing for your to understand is that:

pull requests need to be made against the develop branch

3.2.2 How to Contribute

0. Familiarize yourself with *contributing on github <https://guides.github.com/activities/contributing-to-open-source/>*
1. Fork or branch from the master.
2. Create commits following the commit style
3. Start a pull request to the master branch
4. Wait for a @xeroc or another member to review

3.2.3 Issues

Feel free to submit issues and enhancement requests.

3.2.4 Contributing

Please refer to each project's style guidelines and guidelines for submitting patches and additions. In general, we follow the "fork-and-pull" Git workflow.

1. **Fork** the repo on GitHub
2. **Clone** the project to your own machine
3. **Commit** changes to your own branch
4. **Push** your work back up to your fork
5. Submit a **Pull request** so that we can review your changes

NOTE: Be sure to merge the latest from "upstream" before making a pull request!

3.2.5 Copyright and Licensing

This library is open sources under the MIT license. We require your to release your code under that license as well.

3.3 Public API this.uptick.rocks

3.3.1 this.uptick.rocks

The public API node at `this.uptick.rocks` serves as an *experimental endpoint*. It is offered for free to our best efforts.

You may

- use it for prototyping of your tools
- use it for testing

You may not:

- expect it to be reliable

- spam it with unnecessary load

3.3.2 Running your own node

You can run a similar node with rather low efforts assuming you know how to compile the official bitshares daemon

BitShares Daemon

This is the `config.ini` file for the witness_node:

```
rpc-endpoint = 127.0.0.1:28090         # Accepts JSON-HTTP-RPC requests on␣
↪localhost:28090
required-participation = false         # Do not fail if block
                                       # production stops or you are disconnected from
                                       # the p2p network
bucket-size = [15,60,300,3600,86400]   # The buckets (in seconds) for the market trade␣
↪history
history-per-size = 1000                # Amount of buckets to store
max-ops-per-account = 1000             # Max amount of operations to store in the
                                       # database, per account
                                       # (drastically reduces RAM requirements)
partial-operations = true              # Remove old operation history
                                       # objects from RAM
```

This opens up the port `28090` for localhost. Going forward, you can either open up this port directly to the public, or tunnel it through a webserver (such as nginx) to add SSL on top, do load balancing, throttling etc.

Nginx Webserver

`this.uptick.rocks` uses a nginx server to

- provide a readable websocket url
- provide SSL encryption
- perform throttling
- allow load balancing

The configuration would look like this

```
upstream websockets {       # load balancing two nodes
        server 127.0.0.1:5090;
        server 127.0.0.1:5091;
}

server {
    listen 443 ssl;
    server_name this.uptick.rocks;
    root /var/www/html/;

    # Force HTTPS (this may break some websocket clients that try to
    # connect via HTTP)
    if ($scheme != "https") {
            return 301 https://$host$request_uri;
    }
```

(continues on next page)

(continued from previous page)

```
    keepalive_timeout 65;
    keepalive_requests 100000;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;

    ssl_certificate /etc/letsencrypt/live/this.uptick.rocks/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/this.uptick.rocks/privkey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;
    ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
↪AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-
↪AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-
↪SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-
↪ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-
↪SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-
↪AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-
↪SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!
↪EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-
↪DES-CBC3-SHA';
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_stapling on;
    ssl_stapling_verify on;
    add_header Strict-Transport-Security max-age=15768000;

    location ~ ^(/|/ws) {
        limit_req zone=ws burst=5;
        access_log off;
        proxy_pass http://websockets;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_next_upstream     error timeout invalid_header http_500;
        proxy_connect_timeout   2;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

As you can see from the `upstream` block, the node actually uses a load balancing and failover across **two** locally running `witness_node` nodes. This allows to upgrade the code and reply one one while the other takes over the full traffic, and vise versa.

CHAPTER 4

Standalone App

4.1 Full uptick Command List

Swiss army knife for interacting with the BitShares blockchain.

```
$ uptick --help
Usage: uptick [OPTIONS] COMMAND [ARGS]...

Options:
  --debug / --no-debug            Enable/Disable Debugging (no-broadcasting
                                  mode)
  --node TEXT                     Websocket URL for public BitShares API
                                  (default: "wss://this.uptick.rocks/")
  --rpcuser TEXT                  Websocket user if authentication is required
  --rpcpassword TEXT              Websocket password if authentication is
                                  required
  -d, --nobroadcast / --broadcast
                                  Do not broadcast anything
  -x, --unsigned / --signed       Do not try to sign the transaction
  -e, --expires INTEGER           Expiration time in seconds (defaults to 30)
  -v, --verbose INTEGER           Verbosity (0-15)
  --version                       Show version
  --help                          Show this message and exit.

Commands:
  addkey                  Add a private key to the wallet
  allow                   Add a key/account to an account's permission
  api                     Open an local API for trading bots
  approvecommittee        Approve committee member(s)
  approveproposal         Approve a proposal
  approvewitness          Approve witness(es)
  balance                 Show Account balances
  broadcast               Broadcast a json-formatted transaction
  buy                     Buy a specific asset at a certain rate...
```

(continues on next page)

(continued from previous page)

```
  cancel                   Cancel one or multiple orders
  changewalletpassphrase   Change the wallet passphrase
  configuration            Show configuration variables
  delkey                   Delete a private key from the wallet
  disallow                 Remove a key/account from an account's...
  disapprovecommittee      Disapprove committee member(s)
  disapproveproposal       Disapprove a proposal
  disapprovewitness        Disapprove witness(es)
  feeds                    Price Feed Overview
  getkey                   Obtain private key in WIF format
  history                  Show history of an account
  info                     Obtain all kinds of information
  listaccounts             List accounts (for the connected network)
  listkeys                 List all keys (for all networks)
  newaccount               Create a new account
  newfeed                  Publish a price feed! Examples:  uptick...
  openorders               List open orders of an account
  orderbook                Show the orderbook of a particular market
  permissions              Show permissions of an account
  proposals                List proposals
  randomwif                Obtain a random private/public key pair
  sell                     Sell a specific asset at a certain rate...
  set                      Set configuration parameters
  sign                     Sign a json-formatted transaction
  ticker                   Show ticker of a market
  trades                   List trades in a market
  transfer                 Transfer assets
  upgrade                  Upgrade Account
```

4.2 Commonly Used Commands

4.2.1 Adding keys

uptick comes with its own encrypted wallet to which keys need to be added::

```
uptick addkey
```

On first run, you will be asked to provide a new passphrase that you will need to provide every time you want to post on the BitShares network. Using an empty password is not allowed, however, you can used the *UNLOCK* environmental variable if you need to automatically unlock the wallet with uptick::

```
UNLOCK="password" uptick transfer ....
```

4.2.2 List available Keys and accounts

You can list the installed keys using::

```
uptick listkeys
```

This command will give the list of public keys to which the private keys are available.:

```
uptick listaccounts
```

This command tries to resolve the public keys into account names registered on the network (experimental).

4.2.3 Configuration

`uptick` comes with its owne configuration::

```
uptick set default_account <account-name>
```

All configuration variables are provided with `uptick set --help` You can see your local configuration by calling:

```
uptick configuration
```

4.2.4 Transfer Assets

BitShares can be transfered via:

```
uptick transfer receipient 100.000 BTS
```

If `--author` is not provided, the *default* account as defined with `uptick set author` will be taken.

4.2.5 Buy/Sell Assets

You can of course sell your assets in the internal decentralized exchange that is integrated into the BitShares blockchain by using::

```
uptick buy <amount> <asset-to-buy> <price> <asset-to-sell>
uptick sell <amount> <asset-to-sell> <price> <asset-to-buy>
```

4.2.6 Balances

Get an account's balance with:

```
uptick balance <account>
```

If `<account>` is not provided, the *default* account will be taken.

4.2.7 History

You can get an accounts history by using:

```
uptick history <account>
```

Furthermore you can filter by `types` and limit the result by transaction numer. More information can be found by calling `uptick history -h`.

4.2.8 Permissions

Any account permission can be inspected using:

```
uptick permissions [<account>]
```

The take the following form:

```
+------------+----------+-------------------------------------------------------+
| Permission | Threshold |                                           Key/Account |
+------------+----------+-------------------------------------------------------+
|      owner |         2 |                                            fabian (1) |
|            |           | BTS7mgtsF5XPU9tokFpEz2zN9sQ89oAcRfcaSkZLsiqfWMtRDNKkc (1) |
+------------+----------+-------------------------------------------------------+
|     active |         1 | BTS6quoHiVnmiDEXyz4fAsrNd28G6q7qBCitWbZGo4pTfQn8SwkzD (1) |
+------------+----------+-------------------------------------------------------+
|    posting |         1 |                                         streemian (1) |
|            |           | BTS6xpuUdyoRkRJ1GQmrHeNiVC3KGadjrBayo25HaTyBxBCQNwG3j (1) |
|            |           | BTS8aJtoKdTsrRrWg3PB9XsbsCgZbVeDhQS3VUM1jkcXfVSjbv4T8 (1) |
+------------+----------+-------------------------------------------------------+
```

The permissions are either **owner** (full control over the account), **active** (full control, except for changing the owner), and **posting** (for posting and voting). The keys can either be a public key or another account name while the number behind shows the weight of the entry. If the weight is smaller than the threshold, a single signature will not suffice to validate a transaction

4.2.9 Allow/Disallow

Permissions can be changed using::

```
uptick allow --account <account> --weight 1 --permission posting --threshold 1
↪<foreign_account>
uptick disallow --permission <permissions> <foreign_account>
```

More details and the default parameters can be found via::

```
uptick allow --help
uptick disallow --help
```

4.2.10 Info

uptick can read data from the blockchain and present it to the user in tabular form. It can automatically identify:

- block numbers (`1000021`)
- account names (`uptick`)
- assets (`BTS`)
- public keys (`BTSxxxxxxxxxx`)
- general blockchain parameters

The corresponding data can be presented using::

```
uptick info [block_num [account name [pubkey [identifier [asset]]]]]
```

4.3 Custom Applications

Uptick is designed in a way that allows you to build your **own applications** and use the existing infrastructure of pybitshares and uptick easily. This means that you can use so called **decorators** to simplify development of your own application.

4.3.1 Example 1: Cancel all orders in a market

```
from pprint import pprint
from uptick.decorators import unlock, online
from uptick.main import main
from bitshares.market import Market
import click

@main.command()
@click.option("--account", default=None)
@click.argument("market")
@click.pass_context
@online
@unlock
def cancelall(ctx, market, account):
    market = Market(market)
    ctx.bitshares.bundle = True
    market.cancel([
        x["id"] for x in market.accountopenorders(account)
    ], account=account)
    pprint(ctx.bitshares.txbuffer.broadcast())


if __name__ == "__main__":
    main()
```

4.3.2 Example 2: Spread multiple orders evenly in a market:

```
from pprint import pprint
from numpy import linspace
from uptick.decorators import unlock, online
from uptick.main import main
from bitshares.market import Market
import click


@main.command()
@click.option("--account", default=None)
@click.argument("market")
@click.argument("side", type=click.Choice(['buy', 'sell']))
@click.argument("min", type=float)
@click.argument("max", type=float)
@click.argument("num", type=float)
@click.argument("amount", type=float)
@click.pass_context
@online
@unlock
```

(continues on next page)

(continued from previous page)

```
def spread(ctx, market, side, min, max, num, amount, account):
    market = Market(market)
    ctx.bitshares.bundle = True

    if min < max:
        space = linspace(min, max, num)
    else:
        space = linspace(max, min, num)

    func = getattr(market, side)
    for p in space:
        func(p, amount / float(num), account=account)
    pprint(ctx.bitshares.txbuffer.broadcast())
```

4.3.3 Decorators

uptick.decorators.**chain**(*f*)
This decorator allows you to access `ctx.bitshares` which is an instance of BitShares.

uptick.decorators.**configfile**(*f*)
This decorator will parse a configuration file in YAML format and store the dictionary in `ctx.config`

uptick.decorators.**customchain**(***kwargsChain*)
This decorator allows you to access `ctx.bitshares` which is an instance of BitShares. But in contrast to @chain, this is a decorator that expects parameters that are directed right to `BitShares()`.

... code-block::python

@main.command() @click.option("–worker", default=None) @click.pass_context @customchain(foo="bar") @unlock def list(ctx, worker):

print(ctx.obj)

uptick.decorators.**offline**(*f*)
This decorator allows you to access `ctx.bitshares` which is an instance of BitShares with `offline=True`.

uptick.decorators.**unlock**(*f*)
This decorator will unlock the wallet by either asking for a passphrase or taking the environmental variable `UNLOCK`

uptick.decorators.**verbose**(*f*)
Add verbose flags and add logging handlers

Index