BitEx Documentation

Release 2.0.0

Nils Diefenbach

Jul 19, 2018

Contents

Contents:

CHAPTER 1

BitEx Code Reference

BitEx - Crypto-Exchange REST API Framework.

Author: Nils Diefenbach
Email: 23okrs20+github@mykolab.com
Repository at: https://github.com/Crypto-toolbox/bitex/

CHAPTER 2

API Objects

class bitex.api.base.**BaseAPI**(*, *addr*, *key*, *secret*, *version*, *config*)

BaseAPI provides lowest-common-denominator methods used in all API types.

It provides a nonce() method, basic configuration loading and credential validity checking method check_auth_requirements(), which should be extended in subclasses to cover any additional parameters that are necessary.

check_auth_requirements()

Check that neither self.key nor self.secret are None.

If so, this method raises an IncompleteCredentialsError. Otherwise returns None.

Extend this in child classes if you need to check for further required values.

Raise IncompleteCredentialsError

Returns None

load_config(*fname*)

Load configuration from an ini file.

Return it, in case this func needs to be extended.

Parameters **fname** – path to file

Returns configparser.ConfigParser() Obj

static nonce()

Create a Nonce value for signature generation.

Returns Nonce as string

class bitex.api.REST.rest.**RESTAPI**(*addr*, *timeout=None*, *key=None*, *secret=None*, *version=None*, *config=None*)

Generic REST API interface.

Supplies private and public query methods, as well as building blocks to customize the signature generation process.

generate_uri (*endpoint*)
Generate a Unique Resource Identifier (API Version + Endpoint).

Parameters **endpoint** – str, endpoint path (i.e. /market/btcusd)

Returns str, URI

generate_url (*uri*)
Generate a Unique Resource Locator (API Address + URI).

Parameters **uri** – str, URI

Returns str, URL

sign_request_kwargs (*endpoint*, ***kwargs*)
Generate dummy Request Kwarg Signature.

Extend this to implement signing of requests for private API calls. By default, it supplies a default URL using generate_uri and generate_url.

Parameters

- **endpoint** – str, API Endpoint
- **kwargs** – Kwargs meant for requests.Request()

Returns dict, request kwargs

private_query (*method_verb*, *endpoint*, ***request_kwargs*)
Query a private API endpoint requiring signing of the request.

Parameters

- **method_verb** – valid HTTP Verb (GET, PUT, DELETE, etc.)
- **endpoint** – str, API Endpoint
- **request_kwargs** – kwargs for request.Request()

Returns request.Response() object

public_query (*method_verb*, *endpoint*, ***request_kwargs*)
Query a public (i.e. unauthenticated) API endpoint and return the result.

Parameters

- **method_verb** – valid HTTP Verb (GET, PUT, DELETE, etc.)
- **endpoint** – str, API Endpoint
- **request_kwargs** – kwargs for request.Request()

Returns request.Response() object

2.1 Trading Pairs already implemented

Includes the base class for crypto currency pairs and the PairFormatter class.

It also provides convencience wrappers for commonly used Pairs: these can be imported to avoid typos by the user, and can be directly passed to the APIs.

If the pair you want to query isn't present in here, creating such a pair is simple enough - simply initialize the PairFormatter class with the currencies you want:

```
>>> my_pair = PairFormatter('BaseCurrency', 'QuoteCurrency')
```

The object `my_pair` now takes care of all formatting of any exchange, supported by Bitex, you pass it to.

class bitex.pairs.**PairFormatter**(*base*, *quote*)
Container Class which features formatting function for all supported exchanges.

These Formatter functions apply any changes to make a given pair, pased as quote and base currency, compatible with an exchange. This does NOT include an availability check of the pair. It is therefore possible to format a given pair, even though it is not supported by the requested exchange.

format_for(*exchange_name*)
Format the pair for the given exchange.

static kraken_formatter(*base*, *quote*)
Format the currencies for Kraken.

Generally speaking, Kraken prefixes digital currencies with a capital 'X', and fiat Currencies with a capital 'Z'. There exceptions to this rule, unfortunately, which should be handled as well.

static bitstamp_formatter(*base*, *quote*)
Format currencies for Bitstamp.

static bitfinex_formatter(*base*, *quote*)
Format currencies for bitfinex.

Edgecase: DASH This symbol is shortened to `DSH`.

static bittrex_formatter(*base*, *quote*)
Format currencies for Bittrex.

static binance_formatter(*base*, *quote*)
Format currencies for Binance.

static coincheck_formatter(*base*, *quote*)
Format currencies for CoinCheck.

static gdax_formatter(*base*, *quote*)
Format currencies for GDAX.

static itbit_formatter(*base*, *quote*)
Format currencies for ItBit.

Edge case: BTC BTC is denoted as `XBT`.

static okcoin_formatter(*base*, *quote*)
Format currencies for OKCoin.

static ccex_formatter(*base*, *quote*)
Format currencies for C-CEX.

static cryptopia_formatter(*base*, *quote*)
Format currencies for Cryptopia.

static gemini_formatter(*base*, *quote*)
Format currencies for Gemini.

static yunbi_formatter(*base*, *quote*)
Format currencies for Yunbi.

static rocktrading_formatter(*base*, *quote*)
Format currencies for The Rock Trading LTD.

static poloniex_formatter(*base*, *quote*)
Format currencies for Poloniex.

Edge Case: BTC, USDT and XMR in Quote. As theses symbols have their own markets (several currencies are quoted in them), they must be handled accordingly.

static quoine_formatter(*base*, *quote*)
Format currencies for Quoine.

static quadriga_formatter(*base*, *quote*)
Format currencies for QuadrigaCX.

static hitbtc_formatter(*base*, *quote*)
Format currencies for HitBTC.

static vaultoro_formatter(*base*, *quote*)
Format currencies for Vaultoro.

static bter_formatter(*base*, *quote*)
Format currencies for BTer.

static poloniex_unformatter(*pair*)
Unformat the pair for poloniex exchange.

Removes seperator, swapps base and quote.

class bitex.pairs.**BTCUSDFormatter**
BTC/USD PairFormatter object.

class bitex.pairs.**ETHUSDFormatter**
ETH/USD PairFormatter object.

class bitex.pairs.**XMRUSDFormatter**
XMR/USD PairFormatter object.

class bitex.pairs.**ETCUSDFormatter**
ETC/USD PairFormatter object.

class bitex.pairs.**ZECUSDFormatter**
ZEC/USD PairFormatter object.

class bitex.pairs.**DASHUSDFormatter**
DASH/USD PairFormatter object.

class bitex.pairs.**BCHUSDFormatter**
BCH/USD PairFormatter object.

class bitex.pairs.**ETHBTCFormatter**
ETH/BTC PairFormatter object.

class bitex.pairs.**LTCBTCFormatter**
LTC/BTC PairFormatter object.

class bitex.pairs.**XMRBTCFormatter**
XMR/BTC PairFormatter object.

class bitex.pairs.**ETCBTCFormatter**
ETC/BTC PairFormatter object.

class bitex.pairs.**ZECBTCFormatter**
ZEC/BTC PairFormatter object.

class bitex.pairs.**DASHBTCFormatter**
DASH/BTC PairFormatter object.

class bitex.pairs.**BCHBTCFormatter**
BCH/BTC PairFormatter object.

class bitex.pairs.**XRPUSDFormatter**
XRPUSD Pairformatter.

CHAPTER 3

Interface Objects

class bitex.interface.base.**Interface**(*, *name*, *rest_api*)
Base class for Interface objects.

supported_pairs
Return a list of supported currncy pairs.

Returns list

is_supported(*pair*)
Check if the given pair is present in self._supported_pairs.

Input can either be a string or a PairFormatter Obj (or child thereof). If it is the latter two, we'll call the format() method with the Interface's name attribute to acquire proper formatting.

Parameters **pair** – Str, or bitex.pairs.PairFormatter Object

Returns Bool

request(*verb*, *endpoint*, *authenticate=False*, ***req_kwargs*)
Query the API and return its result.

Parameters

- **verb** – HTTP verb (GET, PUT, DELETE, etc)
- **endpoint** – Str
- **authenticate** – Bool, whether to call private_query or public_query method.
- **req_kwargs** – Kwargs to pass to _query / `requests.Request()`

Raise UnsupportedPairError

Returns `requests.Response() Obj`

class bitex.interface.rest.**RESTInterface**(*name*, *rest_api*)
REST Interface base class.

ticker(*pair*, **args*, ***kwargs*)
Return the ticker for the given pair.

Parameters

- **pair** – Str, pair to request data for.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

order_book(*pair*, **args*, ***kwargs*)
Return the order book for the given pair.

Parameters

- **pair** – Str, pair to request data for.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

trades(*pair*, **args*, ***kwargs*)
Return the trades for the given pair.

Parameters

- **pair** – Str, pair to request data for.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

ask(*pair*, *price*, *size*, **args*, ***kwargs*)
Place an ask order.

Parameters

- **pair** – Str, pair to post order for.
- **price** – Float or str, price you'd like to ask.
- **size** – Float or str, amount of currency you'd like to sell.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

bid(*pair*, *price*, *size*, **args*, ***kwargs*)
Place a bid order.

Parameters

- **pair** – Str, pair to post order for.
- **price** – Float or str, price you'd like to bid.
- **size** – Float or str, amount of currency you'd like to buy.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

order_status(*order_id*, **args*, ***kwargs*)
Return the status of an order with the given id.

Parameters

- **order_id** – Order ID of the order you'd like to have a status for.
- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

open_orders(**args*, ***kwargs*)
Return all open orders.

Parameters

- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

cancel_order(**order_ids*, ***kwargs*)
Cancel the order(s) with the given id(s).

Parameters

- **order_ids** – variable amount of order IDs to cancel.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

wallet(**args*, ***kwargs*)
Return the wallet of this account.

Parameters

- **args** – additional arguments.
- **kwargs** – additional kwargs, passed to requests.Requests() as 'param' kwarg.

Returns `requests.Response()` object.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index