BitEx Documentation

Release 1.2.3

Nils Diefenbach

Feb 20, 2021

CONTENTS:

CHAPTER

ONE

BITEX API REFERENCE

1.1 Core Modules

1.1.1 `bitex.session` Module

A customized version of `requests.Session`, tailored to the `bitex-framework` library.

class `bitex.session.`**BitexSession**(*auth: Optional[*bitex.auth.BitexAuth*] = None*)

Custom `requests.Session` object for keep-alive http connections to API endpoints.

It expects a *`BitexAuth`* instance or subclass thereof on instantiation, and assigns it as the default authentication object for any requests made via this class's instance.

Using one of these methods requires an adequate plugin to be installed for *exchange*. If no such plugin is present, an *`bitex.exceptions.MissingPlugin`* exception is raised by *`bitex.request.BitexPreparedRequest`*.

Using the bitex short-hand is not mandatory, but supported. You may as well construct the entire url of an endpoint you'd like to reach manually, and `bitex-framework` will do the right thing.

cancel_order(*exchange: str*, *pair: str*, *method: str = 'DELETE'*, ***kwargs*) → *bitex.response.BitexResponse*

Cancel an order with the given *order_id* for *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **order_id** – The order id of the order you'd like to cancel.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to DELETE.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

deposit(*exchange: str*, *currency: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request the deposit address of the given *currency*'s wallet.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **currency** (*str*) – The currency to withdraw.

- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

property key
Return the Auth's key attribute value.

Return type str

new_order(*exchange: str*, *pair: str*, *method: str = 'POST'*, ***kwargs*) → *bitex.response.BitexResponse*
Create a new order for *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to POST.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

order_status(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*
Request the order status for *order_id* and *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

orderbook(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*
Request order book data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

prepare_request(*request:* bitex.request.BitexRequest) → *bitex.request.BitexPreparedRequest*
Prepare a `BitexPreparedRequest` object for transmission.

This implementation extends `requests.Session.prepare_request` by making a call to `bitex.list_loaded_plugins` and checking if we have any plugins that may provide a custom `BitexPreparedRequest` class.

request(*method*, *url*, *private=False*, *params=None*, *data=None*, *headers=None*, *cookies=None*, *files=None*, *auth=None*, *timeout=None*, *allow_redirects=True*, *proxies=None*, *hooks=None*, *stream=None*, *verify=None*, *cert=None*, *json=None*) → *bitex.response.BitexResponse*

Construct a `BitexRequest`, prepare and send it.

url may either be a URL starting with http/https, or a `bitex-framework` short-hand url in the format of *<exchange>:<instrument>/<data>/<action>*.

property secret

Return the Auth's secret attribute value.

Return type str

ticker(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request ticker data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

trades(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request trade data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

wallet(*exchange: str*, *currency: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request wallet data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **currency** (*str*) – The currency to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

withdraw(*exchange: str*, *currency: str*, *amount: str*, *method: str = 'PUT'*, ***kwargs*) → *bitex.response.BitexResponse*

Request a withdrawal of the given *currency* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **currency** (*str*) – The currency to withdraw.

- **amount** (*str*) – The amount to withdraw.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

1.1.2 `bitex.auth` Module

Basic auth class for `bitex-framework`.

class bitex.auth.**BitexAuth**(*key: str*, *secret: str*)

Authentication Meta Class for API authentication.

Takes care of generating a signature and preparing data to be sent, headers and URLs as required by the exchange this class is subclassed for.

Parameters

- **key** (*str*) – API Key.
- **secret** (*str*) – API Secret.

static decode_body(*request:* bitex.request.BitexPreparedRequest) → Tuple[Tuple[str, List[Any]], ...]

Decode the urlencoded body of the given request and return it.

Some signature algorithms require us to use parameters supplied via the request body. Since the body is already urlencoded using `requests.PreparedRequest.prepare()`, we need to undo its work before returning the request body's contents.

We must accommodate for the case that in some cases the body may be a JSON encoded string. We expect the parsed JSON to be a dictionary of objects.

Parameters **request** (`BitexPreparedRequest`) – The request whose body we should decode.

property key_as_bytes

Return the key encoded as bytes.

static nonce() → str

Create a Nonce value for signature generation.

By default, this is a unix timestamp with millisecond resolution.

converted to a str.

Return type str

property secret_as_bytes

Return the secret encoded as bytes.

1.1.3 bitex.request Module

bitex-framework extension for requests.Request & requests.PreparedRequest classes.

class bitex.request.**BitexPreparedRequest**(*exchange*)
Bitex extension of :cls"*requests.PreparedRequest*.

Implements a checker function for short-hand urls.

static search_url_for_shorthand(*url*) → Optional[Dict[str, Optional[str]]]
Check if the given URL is a bitex short-hand.

If it is, we return the value of re.Match.groupdict(); otherwise we return None instead.

class bitex.request.**BitexRequest**(*private: bool = False*, ***kwargs*)
Bitex extension of :cls"*requests.Request*.

Implements a parser function for exchange names from a given URL.

Additionally re-implements requests.Request.prepare(), replacing the instantiation of the *requests.PreparedRequest* class with an instance of *BitexPreparedRequest*.

parse_target_exchange() → Optional[str]
Check the URL for its scheme and extract an exchange name, if any.

If the url starts with http/https we set BitexRequest.exchange to *None*. Otherwise we store the *exchange* in said attribute.

prepare() → *bitex.request.BitexPreparedRequest*
Construct a *BitexPreparedRequest* for transmission and return it.

> **Note:** Unlike BitexSession.prepare_request(), this method does *not* apply a custom auth class automatically, if no auth object was given.

1.1.4 bitex.response Module

Customized requests.Response class for the bitex-framework framework.

class bitex.response.**BitexResponse**
Custom requests.Response class.

Supplies additional format outputs of the underlying *JSON* data, as returned by json().

key_value_dict() → Dict[str, Union[str, int, float]]
Return the data of the response in a flattened dict.

This provides the data as a dict of key-value pairs, which is ready for consumption by libraries such as pandas:

```
{
    <label>: <value>,
    <label>: <value>,
    ...
}
```

There are certain keys which should be always present (and are encouraged to be implemented by exchange plugin developers):

- *pair*: denotes the crytpo pair a colelction of kv dictionaries belongs to

- ***received*: denotes the timestamp at the time of creation of this Response, specifically, when** the instance's `BitexResponse.__init__()` method was first called.

..admonition::Disclaimer

As these formatter functions are implemented by plugin developers, we cannot fully guarantee that the presented fields above are in fact **always** present. It's your duty to double-check the exchange plugin documentation and/or code to make sure the fields are present.

triples() → List[Tuple[int, str, Union[str, int, float]]]

Return the data of the response in three-column layout.

Data is returned as a list of 3-item tuples:

```
[
    (<timestamp>, <label>, <value>),
    (<timestamp>, <label>, <value>),
    ...
]
```

There are certain rows which should be always present (and are encouraged to be implemented by exchange plugin developers):

- *pair*: denotes the crytpo pair a colelction of triples belongs to
- ***received*: denotes the timestamp at the time of creation of this Response, specifically, when** the instance's `BitexResponse.__init__()` method was first called.

Also note, that the *timestamp* field in the above example **should** be the ts of the data as given by the exchange and is ** should not** be the timestamp of reception (i.e. instantiation of the response instance). The time of reception is always found at the *received* key.

..admonition::Disclaimer

As these formatter functions are implemented by plugin developers, we cannot fully guarantee that the presented fields above are in fact **always** present. It's your duty to double-check the exchange plugin documentation and/or code to make sure the fields are present.

1.1.5 `bitex.adapter` Module

Custom `requests.HTTPAdapter` for `bitex-framework`.

class bitex.adapter.**BitexHTTPAdapter**(*pool_connections=10*, *pool_maxsize=10*, *max_retries=0*, *pool_block=False*)

Custom HTTP Adapter for `Bitex`.

It replaces `requests.Response` as the default response class when building the response, with either an adequate plugin-supplied class or `bitex-framework` 's own default `BitexResponse` class.

build_response(*req:* bitex.request.BitexPreparedRequest, *resp:* *urllib3.response.HTTPResponse*) → *bitex.response.BitexResponse*

Build a `BitexResponse` from the given *req* and *resp*.

The method is largely identical to `HTTPAdapter.build_response()`, and only differs in the class type used when constructing a response.

This class is taken firstly from any valid plugin that supplies an adequate class for the exchange that was queried (as stated in `BitexPreparedRequest.exchange`), or `bitex-framework` 's own default `BitexResponse` class.

Parameters

- **req** (BitexPreparedRequest) – The BitexPreparedRequest used to generate the response.
- **resp** (*HTTPResponse*) – The urllib3 response object.

1.1.6 bitex.constants Module

bitex.**constants = <module 'bitex.constants' from '/home/docs/checkouts/readthedocs.org/use**
Constants used across the :mod`bitex` code base.

1.1.7 bitex.exceptions Module

Custom exceptions raised by the bitex-framework code base.

exception bitex.exceptions.**MissingPlugin**(*plugin_name: str*, **args: list*, ***kwargs: dict*)
A plugin was required to complete the request.

Parameters **plugin_name** (*str*) – The name of the plugin which is missing.

1.2 Plugin System

1.2.1 Hook Specs

1.2.2 Reference Implementation

BitEx is a FOSS library for accessing Crypto-exchange APIs in a convenient way.

It extends requests and implements a standardized set of methods to interact with exchanges, covering the most commonly used functionality:

- Acquiring the order book.
- Acquiring the latest ticker.
- Acquiring trades.
- Placing new orders.
- Cancelling orders.
- Acquiring an order's status.
- Fetching a wallet's value.
- Fetching a wallet's deposit address.
- Withdrawing coins from the wallet.

Additionally, it provides the option to request endpoints which are not one of the above using a shorthand.

Note: The implementation of additional methods and their naming is dependent on the plugin and its author for the exchange you're accessing.

CHAPTER

TWO

DIVING RIGHT IN

A minimal, working example:

```
>>>from bitex import BitexSession, BitexAuth
>>>auth_obj = BitexAuth(key, secret)
>>>session = BitexSession(auth=auth_obj)
>>>session.ticker("exchange_name", "BTCUSD")
<BitexResponse [200 OK]>
```

If you'd like to access a private endpoint of an API, you'll likely need a custom *BitexAuth* class, extending its `BitexAuth.__call__()` method:

```
class BitexAuthSubClass(BitexAuth):
    def __init__(key, secret):
        super(BitexSessionSubclass, self).__init__(auth)

    def __call__(request):
        request.headers = {'SUPER-SECRET': (self.secret_as_bytes + self.key_as_bytes).
↪encode()}
        return request

>>>auth_obj = BitexAuthSubClass(key, secret)
>>>session = BitexSession(auth=auth_obj)
>>>order_options={'price': 100000, 'size': 10, 'type': 'market'}
>>>session.new_order('exchange_name', "BTCUSD", params=order_options)
<BitexResponse [200 OK]>
```

In the example above, we used bitex's set of standardized methods for accessing the change. However, you may also request data using the bitex short-hand notation.

CHAPTER

THREE

MAKING YOUR LIFE EASIER: THE BITEX URL SHORT-HAND NOTATION

The short-hand notation unifies urls and aims to make using and writing plugins easier.

The short-hand looks like this:

```
<exchange>:<instrument>/<endpoint>[/<action>]
```

exchange refers to the exchange you want to request data from. *instrument* is either a single *currency* or a currency *pair*. *endpoint* describes the kind of endpoint you'd like to use:

- *trades*
- *book*
- *ticker*
- *wallet*
- *order*

The latter two endpoint types support *actions*, which are listed below:

- *wallet* actions :
  - *deposit*
  - *withdraw*

  Additionally, a *amount* parameter is always present on the *withdraw* action.
- *order* actions:
  - *new*
  - *status*
  - *cancel*

The previous examples would look as follows, if they used the shorthand instead:

```
>>>auth_obj = BitexAuthSubClass(key, secret)
>>>session = BitexSession(auth=auth_obj)
>>>session.get("SomeExchange:ticker/BTCUSD")
<BitexResponse [200 OK]>
>>>order_options={'price': 100000, 'size': 10, 'type': 'market'}
>>>session.post("SomeExchange:BTCUSD/order/new", params=order_options)
<BitexResponse [200 OK]>
```

As long as a plugin for *SomeExchange* is installed, `bitex-framework` will convert the short-hand to a fully-qualified URL under the hood.

class bitex.**BitexHTTPAdapter**(*pool_connections=10*, *pool_maxsize=10*, *max_retries=0*, *pool_block=False*)

Custom HTTP Adapter for `Bitex`.

It replaces `requests.Response` as the default response class when building the response, with either an adequate plugin-supplied class or `bitex-framework` 's own default `BitexResponse` class.

build_response(*req: bitex.request.BitexPreparedRequest*, *resp: urllib3.response.HTTPResponse*) → *bitex.response.BitexResponse*

Build a `BitexResponse` from the given *req* and *resp*.

The method is largely identical to `HTTPAdapter.build_response()`, and only differs in the class type used when constructing a response.

This class is taken firstly from any valid plugin that supplies an adequate class for the exchange that was queried (as stated in `BitexPreparedRequest.exchange`), or `bitex-framework` 's own default `BitexResponse` class.

Parameters

- **req** (*BitexPreparedRequest*) – The `BitexPreparedRequest` used to generate the response.
- **resp** (*HTTPResponse*) – The urllib3 response object.

class bitex.**BitexSession**(*auth: Optional[bitex.auth.BitexAuth] = None*)

Custom `requests.Session` object for keep-alive http connections to API endpoints.

It expects a *BitexAuth* instance or subclass thereof on instantiation, and assigns it as the default authentication object for any requests made via this class's instance.

Using one of these methods requires an adequate plugin to be installed for *exchange*. If no such plugin is present, an *bitex.exceptions.MissingPlugin* exception is raised by *bitex.request.BitexPreparedRequest*.

Using the bitex short-hand is not mandatory, but supported. You may as well construct the entire url of an endpoint you'd like to reach manually, and `bitex-framework` will do the right thing.

cancel_order(*exchange: str*, *pair: str*, *method: str = 'DELETE'*, ***kwargs*) → *bitex.response.BitexResponse*

Cancel an order with the given *order_id* for *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **order_id** – The order id of the order you'd like to cancel.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to DELETE.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

deposit(*exchange: str*, *currency: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request the deposit address of the given *currency*'s wallet.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.

- **currency** (*str*) – The currency to withdraw.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

property key
Return the Auth's key attribute value.

Return type str

new_order(*exchange: str, pair: str, method: str = 'POST', **kwargs*) → *bitex.response.BitexResponse*
Create a new order for *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to POST.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

order_status(*exchange: str, pair: str, method: str = 'GET', **kwargs*) → *bitex.response.BitexResponse*
Request the order status for *order_id* and *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to place the order for.
- **method** (*str*) – The HTTP method to use when placing the order. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

Return type *BitexResponse*

orderbook(*exchange: str, pair: str, method: str = 'GET', **kwargs*) → *bitex.response.BitexResponse*
Request order book data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

prepare_request(*request:* bitex.request.BitexRequest) → *bitex.request.BitexPreparedRequest*
Prepare a `BitexPreparedRequest` object for transmission.

This implementation extends `requests.Session.prepare_request` by making a call to `bitex.list_loaded_plugins` and checking if we have any plugins that may provide a custom `BitexPreparedRequest` class.

request(*method*, *url*, *private=False*, *params=None*, *data=None*, *headers=None*, *cookies=None*, *files=None*, *auth=None*, *timeout=None*, *allow_redirects=True*, *proxies=None*, *hooks=None*, *stream=None*, *verify=None*, *cert=None*, *json=None*) → *bitex.response.BitexResponse*

Construct a `BitexRequest`, prepare and send it.

url may either be a URL starting with http/https, or a `bitex-framework` short-hand url in the format of *<exchange>:<instrument>/<data>/<action>*.

property secret

Return the Auth's secret attribute value.

Return type str

ticker(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request ticker data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

trades(*exchange: str*, *pair: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request trade data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **pair** (*str*) – The currency pair to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

wallet(*exchange: str*, *currency: str*, *method: str = 'GET'*, ***kwargs*) → *bitex.response.BitexResponse*

Request wallet data for the given *pair* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **currency** (*str*) – The currency to request data for.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

withdraw(*exchange: str*, *currency: str*, *amount: str*, *method: str = 'PUT'*, ***kwargs*) → *bitex.response.BitexResponse*

Request a withdrawal of the given *currency* at the given *exchange*.

Parameters

- **exchange** (*str*) – The exchange you'd like to request data from.
- **currency** (*str*) – The currency to withdraw.

- **amount** (*str*) – The amount to withdraw.
- **method** (*str*) – The HTTP method to use when requesting the data. This defaults to GET.
- **kwargs** (*Any*) – Additional keyword arguments which are passed on to `requests.Session.request()`.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

S

T

W