web3data-py Documentation

Release 0.1.7

Dominik Muhs

Feb 17, 2022

CONTENTS:

CHAPTER

ONE

WEB3DATA-PY PYTHON API

1.1 Obtaining an API Key

Visit Amberdata.io and select the developer plan to get started! Pass your API key to the client instance, either has a hardcoded string, or through an environment variable:

```
from web3data import Web3Data
w3d = Web3Data("<your key>")
```

. . . and start querying!

1.2 Installation

To install web3data-py, run this command in your terminal:

```
$ pip install web3data
```

For alternative ways to install the package, check out the installation instructions

1.3 Usage

```
from web3data import Web3Data

w3d = Web3Data("<your key>")
print(w3d.eth.address.information("0x06012c8cf97bead5deae237070f9587f8e7a266d"))
```

This will print the raw response, such as:

```
{'status': 200,
 'title': 'OK',
 'description': 'Successful request',
 'payload': {'balance': '5296672643815245964',
  'balanceIn': '3.089490543793732271555 1e+22',
  'balanceOut': '3.088960876529350746958 7e+22',
  'addressType': 'contract',
  'changeInPrice': None,
  'contractTypes': ['ERC721'],
  'decimals': '0',
  'name': 'CryptoKitties',
  'numHolders': '84753',
  'numTokens': '1860119',
  'numTransfers': '2723659',
  'symbol': 'CK',
  'totalSupply': '1860119.0000000000000000',
  'totalValueUSD': None,
  'unitValueUSD': None}}
```


1.4 Development

Check out our contribution guidelines to see how to install the development version and run the test suite!

Don't have the time to contribute? Open up an issue and we'll get it fixed! Simply like the project? Tip me some BAT to sponsor development! :)

1.5 Resources

- Free software: MIT license
- Documentation: https://web3data-py.readthedocs.io.

1.6 Credits

The initial version of this package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install web3data-py, run this command in your terminal:

```
$ pip install web3data
```

This is the preferred method to install web3data-py, 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 web3data-py can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dmuhs/web3data-py
```

Or download the tarball:

```
$ curl -OJL https://github.com/dmuhs/web3data-py/tarball/master
```

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

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

CHAPTER

THREE

USAGE

To use web3data-py in a project:

```
from web3data import Web3Data

w3d = Web3Data("<your key>")
print(w3d.eth.address.information("0x06012c8cf97bead5deae237070f9587f8e7a266d"))
```

This will print the raw response, such as:

```
{'status': 200,
 'title': 'OK',
 'description': 'Successful request',
 'payload': {'balance': '5296672643815245964',
  'balanceIn': '3.089490543793732271555 1e+22',
  'balanceOut': '3.088960876529350746958 7e+22',
  'addressType': 'contract',
  'changeInPrice': None,
  'contractTypes': ['ERC721'],
  'decimals': '0',
  'name': 'CryptoKitties',
  'numHolders': '84753',
  'numTokens': '1860119',
  'numTransfers': '2723659',
  'symbol': 'CK',
  'totalSupply': '1860119.000000000000000 0',
  'totalValueUSD': None,
  'unitValueUSD': None}}
```

3.1 Supported Chains and Handlers

Each endpoint of the Amberdata web3 API can be hit for a specified chain. `web3data-py` follows the paradigm set by web3data-js and allows easy switching between chains by providing them as client attributes. Each attribute implements a sub-handler for several kinds of data, such as address-, market-, or transaction-related information.

The methods for each chain are fixed, however some chains might raise an `APIError` if the data is unavailable. For example, token-related queries on Bitcoin will raise an exception, because Bitcoin does not allow for smart-contracts and token implementations on-chain.

```
In [1]: w3d.eth.token.supply_latest("0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2")
Out[1]:
{'status': 200,
```

(continues on next page)

(continued from previous page)

```
 'title': 'OK',
 'description': 'Successful request',
 'payload': {'decimals': '18',
  'circulatingSupply': '985776.2571660122663385',
  'totalBurned': '1014178.1439074671310546',
  'totalMinted': '1999953.40106534372698',
  'totalSupply': '985775.2571578765959254',
  'totalTransfers': '678572'}}
```

And on the other hand:

```
In [1]: w3d.btc.token.supply_latest("0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2")
---------------------------------------------------------------------------
APIError                                  Traceback (most recent call last)
<ipython-input-12-93158fe945ad> in <module>
----> 1 w3d.btc.token.supply_latest("0x9f8f72aa9304c8b593d555f12ef6589cc3a579a2")

~/repos/web3data-py/web3data/handlers/token.py in supply_latest(self, address)
    115         :return: The API response parsed into a dict
    116         """
--> 117         self._check_chain_supported()
    118         return self._token_query(address, "supplies/latest", {})
    119

~/repos/web3data-py/web3data/handlers/base.py in _check_chain_supported(self)
     34     def _check_chain_supported(self):
     35         if self.chain in self.LIMITED:
---> 36             raise APIError(f"This method is not supported for {self.chain}")
     37
     38     @staticmethod

APIError: This method is not supported for Chains.BTC
```

This behaviour aims to notify the developer as early as possible about invalid code and business logic errors that need fixing right away.

Currently, Amberdata supports the following chains, which are implemented as client instance attributes:

- `w3d.aion`
- `w3d.bch`
- `w3d.bsv`
- `w3d.btc`
- `w3d.eth`
- `w3d.eth_rinkeby`
- `w3d.ltc`
- `w3d.xlm`
- `w3d.zec`

Each chain attribute implements the following sub-handlers for specific API queries:

- `address`
- `block`

- `contract`
- `market`
- `signature`
- `token`
- `transaction`

Further information on the implementation details can be found in the package documentation.

CHAPTER

FOUR

WEB3DATA PACKAGE

4.1 Subpackages

4.1.1 web3data.client

This module contains the main API client class.

class web3data.client.**Web3Data**(*api_key: str*)
Bases: object

The Amberdata API client object.

4.1.2 web3data.exceptions

This module contains API-related exceptions.

exception web3data.exceptions.**APIError**
Bases: Exception

An exception denoting generic API errors.

This error is raised when the API returns invalid response data, like invalid JSON.

exception web3data.exceptions.**EmptyResponseError**
Bases: *web3data.exceptions.APIError*

An exception denoting an empty API response.

This error is raised when the API response content is empty, or it contains an empty JSON object.

4.1.3 web3data.chains

This module contains an enum with names of supported chains.

class web3data.chains.**Chains**(*value*)
Bases: enum.Enum

Blockchains supported by the Amberdata API.

BCH = 2

BSV = 3

BTC = 1

ETH = 4

ETH_RINKEBY = 5

LTC = 6

ZEC = 7

4.1.4 web3data.handlers

Subpackages

web3data.handlers.address

This module contains the address subhandler.

class web3data.handlers.address.**AddressHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)

Bases: *web3data.handlers.base.BaseHandler*

The subhandler for address-related queries.

adoption(*address: str*, ***kwargs*) → Dict

Retrieves the historical adoption for the specified address.

Parameters **address** – The address to fetch information for

Key timeFormat The time format to use with the timestamps: milliseconds/ms or iso/iso8611 (str)

Key timeFrame The time frame to return the historical data in (str): by day (1d, 2d, ..., all), by hour (1h, 2h, ..., 72h) or by minute (1m, 2m, ..., 360m)

Key timePeriod The time period (in minutes) to aggregate the historical data (str)

Returns The API response parsed into a dict

balance_historical(*address: str*, ***kwargs*) → Dict

Retrieves the historical (time series) account balances for the specified address.

Parameters **address** – The address to fetch information for

Key blockNumber Filter by account balances at block number (int)

Key startDate Filter by account balances which happened after this date (int)

Key endDate Filter by account balances which happened before this date (int)

Key value Filter by account balances where the balance is equal to this value (int)

Key valueGt Filter by account balances where the balance is greater than this value (int)

Key valueGte Filter by account balances where the balance is greater than or equal to this value (int)

Key valueLt Filter by account balances where the balance is less than this value (int)

Key valueLte Filter by account balances where the balance is less than or equal to this value (int)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

balance_latest(*address: str*, ***kwargs*) → Dict
Retrieves the current account balance for the specified address.

Parameters **address** – The address to fetch information for

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (str)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

balances(*address: str*, ***kwargs*) → Dict
Retrieves the latest account and token balances for the specified address.

Parameters **address** – The address to fetch information for

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Returns The API response parsed into a dict

balances_batch(*addresses: List[str]*, ***kwargs*) → Dict
Retrieves the latest account and token balances for the specified addresses.

This is super useful if you want to get an entire portfolio's summary in a single call. Get totals for ETH & all token amounts with market prices.

Parameters **addresses** – The addresses to fetch information for

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Returns The API response parsed into a dict

information(*address: str*, ***kwargs*) → Dict
Retrieves information about the specified address.

This includes network(s) and blockchain(s) this address exist within.

Parameters **address** – The address to fetch information for

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

internal_messages(*address: str*, ***kwargs*) → Dict
Retrieves internal messages where this address is either the originator or a recipient.

Parameters **address** – The address to fetch information for

Key blockNumber Filter by internal messages contained within this block number (int)

Key from Filter by internal messages for this "from" address (str)

Key to Filter by internal messages for this "to" address (str)

Key transactionHash Filter by internal messages for this transaction (str)

Key startDate Filter by internal messages which happened after this date (int)

Key endDate Filter by internal messages which happened before this date (int)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

logs(*address: str*, ***kwargs*) → Dict
Retrieves the logs for the transactions where this address is either the originator or a recipient.

Parameters **address** – The address to fetch information for

Key blockNumber Filter by logs contained in this block number (int)

Key startDate Filter by logs which happened after this date (int)

Key endDate Filter by logs which happened before this date (int)

Key topic Filter by logs containing this topic (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

metadata(*address: str*, ***kwargs*) → Dict
Retrieves statistics about the specified address: balances, holdings, etc.

Parameters **address** – The address to fetch information for

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Returns The API response parsed into a dict

metrics() → Dict
Get metrics for all addresses that have exist publicly for a given blockchain.

Default metrics are for Ethereum over a 24h period.

Returns The API response parsed into a dict

pending_transactions(*address: str*, ***kwargs*) → Dict
Retrieves pending transactions the specified address is involved in.

Parameters **address** – The address to fetch information for

Key from Filter by transactions for this "from" address. (str)

Key to Filter by transactions for this "to" address (str)

Key startDate Filter by transactions which happened after this date. (int)

Key endDate Filter by transactions which happened before this date. (int)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key page The page number to return. (int)

Key size The number of records per page. (int)

Returns The API response parsed into a dict

token_balances_historical(*address: str*, ***kwargs*) → Dict

Retrieves the historical (time series) token balances for the specified address.

Parameters **address** – The address to fetch information for

Key amount Filters by token balances which value is equal to this amount (int)

Key amountGt Filter by token balances which value is greater than this amount (int)

Key amountGte Filter by token balances which value is greater than or equal to this amount (int)

Key amountLt Filter by token balances which value is less than this amount (int)

Key amountLte Filter by token balances which value is less than or equal to this amount (int)

Key tokenHolder Filter by token balances which are held by this address (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

token_balances_latest(*address: str*, ***kwargs*) → Dict

Retrieves the tokens this address is holding.

Parameters **address** – The address to fetch information for

Key direction The direction by which to sort the tokens (ascending or descending). (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (str)

Key currency The currency of the price information (usd or eth.) - only used in conjunction with includePrice. (str)

Key sortType The metric by which to rank the tokens (amount, name, symbol). (str)

Key page The page number to return. (str)

Key size The number of records per page. (str)

Returns The API response parsed into a dict

token_transfers(*address: str*, ***kwargs*) → Dict

Retrieves all token transfers involving the specified address.

Parameters **address** – The address to fetch information for

Key amount Filter by token transfers which value is equal to this amount. (int)

Key amountGt Filter by token transfers which value is greater than this amount. (int)

Key amountGte Filter by token transfers which value is greater than or equal to this amount. (int)

Key amountLt Filter by token transfers which value is less than this amount. (int)

Key amountLte Filter by token transfers which value is less than or equal to this amount (int)

Key blockNumber Filter by token transfers with this block number. (int)

Key recipientAddress Filter by token transfers which recipient is the specified address. (str)

Key senderAddress Filter by token transfers which sender is the specified address. (str)

Key startDate Filter by token transfers which happened after this date. (int)

Key endDate Filter by token transfers which happened before this date. (int)

Key tokenAddress Filter by token transfers for this token. (str)

Key transactionHash Filter by token transfers for this transaction hash. (str)

Key page The page number to return. (int)

Key size Number of records per page. (int)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

`total`(***kwargs*) → Dict

Retrieves every Ethereum address that has been seen on the network.

Key hash Filter by a specific address (str)

Key blockNumber Filter by addresses first encountered at this block number (int)

Key blockNumberGt Filter by addresses first encountered after this block number, not including this blocknumber (int)

Key blockNumberGte Filter by addresses first encountered after this block number, including this block number (str)

Key blockNumberLt Filter by addresses first encountered before this block number, not including this block number (int)

Key blockNumberLte Filter by addresses first encountered before this block number, including this block number (str)

Key startDate Filter by addresses first encountered after this date (int)

Key endDate Filter by addresses first encountered after before date (int)

Key type Filter by addresses of the specified type (EOA or CONTRACT) (str)

Key transactionHash Filter by addresses first encountered at this transaction hash (str)

Key page The page number to return. (int)

Key size Number of records per page. (int)

Returns The API response parsed into a dict

transactions(*address: str*, ***kwargs*) → Dict

Retrieves the transactions where this address was either the originator or a recipient.

Parameters **address** – The address to fetch information for

Key blockNumber Filter by transactions for this block number. (int)

Key from Filter by transactions for this "from" address. (str)

Key to Filter by transactions for this "to" address (str)

Key startDate Filter by transactions which happened after this date. (date)

Key endDate Filter by transactions which happened before this date. (date)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key includeFunctions Indicates whether or not to include functions (aka internal messages) information for each transaction, if available (falseltrue). (bool)

Key includeLogs Indicates whether or not to include log information for each transaction, if available (falseltrue). (bool)

Key includeTokenTransfers Indicates whether or not to include token transfers information for each transaction, if available (falseltrue). (bool)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key page The page number to return. (int)

Key size The number of records per page. (int)

Returns The API response parsed into a dict

usage(*address: str*, ***kwargs*) → Dict

Retrieves the historical usage for the specified address.

Parameters **address** – The address to fetch information for

Key timeFormat The time format to use with the timestamps: milliseconds/ms or iso/iso8611 (str)

Key timeFrame The time frame to return the historical data in: by day (1d, 2d, . . . , all), by hour (1h, 2h, . . . , 72h) or by minute (1m, 2m, . . . , 360m) (str)

Key timePeriod The time period (in minutes) to aggregate the historical data. (str)

Returns The API response parsed into a dict

web3data.handlers.api

This module contains the main API handler class.

class web3data.handlers.api.**APIHandler**(*api_key: str*, *blockchain_id: str*, *chain: web3data.chains.Chains*)

Bases: object

The API handler object for client requests.

rpc(*method: str*, *params: List[str]*, *ident: int = 1*)

Perform an HTTP POST RPC call on the API.

Consult the docs here for further details on supported commands: https://docs.amberdata.io/reference#rpc-overview

Parameters

- **method** – The RPC method to call
- **params** – Parameters attached to the RPC call
- **ident** – RPC call identifier

web3data.handlers.base

This module contains the API handler's base class.

class web3data.handlers.base.**BaseHandler**(*chain: web3data.chains.Chains*)

Bases: object

The API handler base class.

This class defines the basic methods of performing REST API endpoint queries as well as RPC queries, which are implemented across all handler classes to standardize API requests.

LIMITED = (<Chains.BTC: 1>, <Chains.BCH: 2>, <Chains.BSV: 3>, <Chains.LTC: 6>, <Chains

static raw_query(*base_url: str*, *route: str*, *headers: Dict[str, str]*, *params: Dict[str, str]*) → Union[Dict, str]

Perform an HTTP GET request on an API REST endpoint.

Parameters

- **base_url** – The API base URL (common prefix)
- **route** – The endpoint route after the base (variable suffix)
- **headers** – Headers to attach to the API request
- **params** – Query parameters to attach to the URL

Returns The API response parsed into a dict

web3data.handlers.block

This module contains the address subhandler.

class web3data.handlers.block.**BlockHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)

Bases: *web3data.handlers.base.BaseHandler*

The subhandler for block-related queries.

functions(*block_id: str*, ***kwargs*) → Dict
Retrieves all the functions which were called at the specified block number or hash.

Parameters **block_id** – The block's ID to fetch information for

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Returns The API response parsed into a dict

logs(*block_id: str*, ***kwargs*) → Dict
Retrieves all the logs at the specified block number or hash.

Parameters **block_id** – The block's ID to fetch information for

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key transactionHash Filter by logs for this transaction. (str)

Returns The API response parsed into a dict

metrics_historical(***kwargs*) → Dict
Get metrics for historical confirmed blocks for a given blockchain.

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Key timeFrame time frame to return the historical data in, options: (1m, 5m, 1h, 1d, 1w) (str)

Key startDate Filter by data after this date. (str)

Key endDate Filter by data before this date. (str)

Returns The API response parsed into a dict

metrics_latest(***kwargs*) → Dict
Get metrics for recent confirmed blocks for a given blockchain.

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Returns The API response parsed into a dict

single(*block_id: str*, ***kwargs*) → Dict
Retrieves the block specified by its id (number or hash).

Parameters **block_id** – The block's ID to fetch information for

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key timeFormat The time format to use for the timestamps (milliseconds, ms, iso, iso8611). (str)

Returns The API response parsed into a dict

token_transfers(*block_id: str*, ***kwargs*) → Dict
Retrieves all the token which were transferred at the specified block number.

Parameters **block_id** – The block's ID to fetch information for

Key amount Filter by tokens transfers where the number of tokens is equal to the specified amount. (int)

Key amountGt Filter by token transfers where the number of tokens is more than the specified amount. (int)

Key amountGte Filter by token transfers where the number of tokens is more than or equal to the specified amount. (int)

Key amountLt Filter by token transfers where the number of tokens is less than the specified amount. (int)

Key amountLte Filter by token transfers where the number of tokens is less than or equal to the specified amount. (int)

Key from Filter by token transfers originating from this address. (str)

Key to Filter token transfers received by this address. (str)

Key tokenAddress Filter by token transfers for this token. (str)

Key transactionHash Filter by token transfers for this transaction. (str)

Key includePrice Indicates whether or not to include price data with the results. (bool)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

total(***kwargs*) → Dict
Retrieves all the blocks within the specified range.

Key startNumber The range of blocks to return, inclusive (startNumber and endNumber should be both specified, or both empty) (str)

Key endNumber The end of the range of blocks to return, exclusive (startNumber and endNumber should be both specified, or both empty) (str)

Key size The number of results to return. (int)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Returns The API response parsed into a dict

transactions(*block_id: str*, ***kwargs*) → Dict
Retrieves all the transactions included in a specified block id.

Parameters **block_id** – The block's ID to fetch information for

Key includeFunctions Indicates whether or not to include functions (aka internal messages) information for each transaction, if available. Options: true, false. (bool)

Key includeTokenTransfers Indicates whether or not to include token transfers information for each transaction, if available. Options: true, false. (bool)

Key includeLogs Indicates whether or not to include log information for each transaction, if available. Options: true, false. (bool)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key startDate Filter by transactions executed after this date. Note that the interval can not exceed 1 minute (startDate and endDate should be both specified, or both empty) (int)

Key endDate Filter by transactions executed before this date. Note that the interval can not exceed 1 minute (startDate and endDate should be both specified, or both empty). (int)

Key size The number of results to return. (int)

Returns The API response parsed into a dict

web3data.handlers.contract

This module contains the address subhandler.

class web3data.handlers.contract.**ContractHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)

Bases: *web3data.handlers.base.BaseHandler*

The subhandler for contract-related queries.

audit(*address: str*, ***kwargs*) → Dict

Retrieves the vulnerabilities audit for the specified contract (if available).

The automated security checks are provided by MythX. Check out their stellar service over at https://mythx.io/.

Parameters

- **address** – The address to fetch information for
- **kwargs** – Additional query parameter options

Returns The API response parsed into a dict

details(*address: str*, ***kwargs*) → Dict

Retrieves all the detailed information for the specified contract (ABI, bytecode, sourcecode...).

Parameters

- **address** – The address to fetch information for
- **kwargs** – Additional query parameter options

Returns The API response parsed into a dict

functions(*address: str*, ***kwargs*) → Dict

Retrieves the functions of the specified contract (if available).

If not available on chain, the byte code is decompiled and a list of functions is extracted from it.

Parameters

- **address** – The address to fetch information for
- **kwargs** – Additional query parameter options

Returns The API response parsed into a dict

web3data.handlers.market

This module contains the address subhandler.

class web3data.handlers.market.**MarketHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)

Bases: *web3data.handlers.base.BaseHandler*

The subhandler for market-related queries.

base_wap_latest(*base: str*, ***kwargs*) → Dict

Retrieves the latest VWAP & TWAP price for the specified base.

Parameters **base** – The pair's base

Key quote The currency of the pair. Example: if pair is "eth_usd", then quote is "usd" (str)

Key timeFormat Time format to use for the timestamps (Options: milliseconds, ms, iso, iso8611) (str)

Returns The API response parsed into a dict

exchanges(***kwargs*) → Dict

Retrieves information about supported exchange-pairs.

These types of data are supported: - ticker - ohlc (open-high-low-close) - trade - order_book - order_book_update

Key exchange only return data for the given exchanges (comma separated) (str)

Key pair only return data for the given pairs (comma separated) (str)

Returns The API response parsed into a dict

ohlcv(***kwargs*) → Dict

Retrieves information about supported exchange-pairs for ohlcv.

Key exchange Filter by data for the given exchanges (comma separated). (str)

Returns The API response parsed into a dict

ohlcv_pair_historical(*pair: str*, ***kwargs*) → Dict

Retrieves the historical (time series) open-high-low-close for the specified pair.

Note: This endpoint returns a max of 6 months of historical data. In order to get more than 6 months you must use the startDate & endDate parameters to move the time frame window to get the next n days/months of data.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve OHLCV. Example: exchange=bitfinex,bitstamp (str)

Key startDate Filter by pairs after this date. (int)

Key endDate Filter by pairs before this date. (int)

Key timeInterval Time interval to return the historical data in ("days" | "hours" | "minutes") (str)

Key timeFormat Time format to use for the timestamps ("milliseconds" | "ms" | "iso" | "iso8611") (str)

Returns The API response parsed into a dict

ohlcv_pair_latest (*pair: str*, ***kwargs*) → Dict
Retrieves the latest open-high-low-close for the specified pair.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve OHLCV. Example: exchange=bitfinex,bitstamp (str)

Returns The API response parsed into a dict

order_best_bid_historical (*pair: str*, ***kwargs*) → Dict
Retrieves historical best bid and offer information for the specified pair.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve order book data. Example: exchange=bitfinex,bitstamp (str)

Key timeFormat The timestamp format to use for the timestamps: milliseconds/ms or iso/iso8611. (str)

Key startDate Filter by pairs after this date. (int)

Key endDate Filter by pairs before this date. (int)

Returns The API response parsed into a dict

order_best_bid_latest (*pair: str*, ***kwargs*) → Dict
Retrieves the latest best bid and offer information for the specified pair and exchange.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve order book data. Example: exchange=bitfinex,bitstamp (str)

Key timeFormat The timestamp format to use for the timestamps: milliseconds/ms or iso/iso8611. (str)

Returns The API response parsed into a dict

order_book (*pair: str*, ***kwargs*) → Dict
Retrieves the order book data for the specified pair.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve order book data. Example: exchange=bitfinex,bitstamp (str)

Key timestamp The timestamp at which to return the order book information (closest match, lower or equal to the timestamp specified). (str)

Key timeFormat The timestamp format to use for the timestamps: milliseconds/ms or iso/iso8611. (str)

Key startDate Filter by pairs after this date. Formats: milliseconds, iso, or iso8611 (str)

Key endDate Filter by pairs before this date. Formats: milliseconds, iso, or iso8611. Note: Must be greater than startDate and cannot exceed 10 minutes. (str)

Returns The API response parsed into a dict

order_book_updates (*pair: str*, ***kwargs*) → Dict

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve order book data. Example: exchange=bitfinex,bitstamp (str)

Key timeFormat The timestamp format to use for the timestamps: milliseconds/ms or iso/iso8611. (str)

Key startDate Filter by pairs after this date. (int)

Key endDate Filter by pairs before this date. (int)

Returns The API response parsed into a dict

pairs(***kwargs*) → Dict

Retrieves information about supported exchange-pairs. These types of data are supported:

- ticker
- ohlc (open-high-low-close)
- trade
- order_book
- order_book_update

Key exchange only return data for the given exchanges (comma separated) (str)

Key pair only return data for the given pairs (comma separated) (str)

Returns The API response parsed into a dict

price_pair_historical(*pair: str*, ***kwargs*) → Dict

Retrieves the historical prices for the specified asset.

Parameters **pair** – The asset pair to look up

Key timeFormat Time format to use for the timestamps (Options: milliseconds, ms, iso, iso8611) (str)

Key startDate Filter by prices after this date. (str)

Key endDate Filter by prices before this date. (str)

Key timeInterval Time interval to return the historical data in ("days" | "hours" | "minutes") (str)

Returns The API response parsed into a dict

price_pair_latest(*pair: str*, ***kwargs*) → Dict

Retrieves the latest price for the specified asset.

Parameters **pair** – The asset pair to look up

Key timeFormat Time format to use for the timestamps (Options: milliseconds, ms, iso, iso8611) (str)

Returns The API response parsed into a dict

price_pairs() → Dict

Retrieves the assets for which latest prices are available.

Returns The API response parsed into a dict

rankings(***kwargs*) → Dict

Retrieves the top ranked assets by a specific metric.

Key direction The sort order in which assets are ranked ascending or descending. (str)

Key sortType The metric used to rank the assets. Options: changeInPrice, currentPrice, liquidMarketCap, marketCap, tokenVelocity, tradeVolume, transactionVolume, uniqueAddresses (str)

Key timeInterval The time interval in which to return the historical data days or hours. (str)

Key type The type(s) of assets to include in the rankings: erc20l, erc721, erc777, erc884, erc998. Note: leaving this parameter empty means all tokens will be included. (str)

Key page The page number to return. (int)

Key size The number of records per page. (int)

Returns The API response parsed into a dict

ticker_bid_ask_historical(*pair: str*, ***kwargs*) → Dict

Retrieves the historical ticker, bid/ask/mid/last, for the specified pair.

Note: This endpoint returns a max of 6 months of historical data. In order to get more than 6 months you must use the startDate & endDate parameters to move the time frame window to get the next n days/months of data.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve market tickers. Example: exchange=bitfinex,bitstamp (str)

Key startDate Filter by ticker pairs after this date. (int)

Key endDate Filter by ticker pairs before this date. (int)

Returns The API response parsed into a dict

ticker_bid_ask_latest(*pair: str*, ***kwargs*) → Dict

Retrieves the latest market ticker Bid/Ask/Mid/Last for the specified pair.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve market tickers. Example: exchange=bitfinex,bitstamp (str)

Returns The API response parsed into a dict

ticker_pairs(***kwargs*) → Dict

Retrieves the list of all available market tickers.

Key exchange Only return data for the given exchanges (comma separated). (str)

Returns The API response parsed into a dict

token_price_historical(*address: str*, ***kwargs*) → Dict

Parameters **address** – The token's smart contract address

Key currency The additional currency (other than ETH and USD) for which to return price info. (str)

Key timeFormat The time format to use for the timestamps: milliseconds/ms or iso/iso861. (str)

Key timeInterval The time interval to return the historical data in: by day (d), by hour (h), or by minute (m). (str)

Key startDate Filter by prices after this date. Note that the interval can not exceed 6 months (d), 30 days (h) or 24 hours (m). (int)

Key endDate Filter by prices before this date. Note that the interval can not exceed 6 months (d), 30 days (h) or 24 hours (m). (int)

Returns The API response parsed into a dict

token_price_latest(*address: str*, ***kwargs*) → Dict
Retrieves the latest price (and other market information) for the specified token.

Parameters **address** – The token's smart contract address

Key currency The additional currency (other than ETH and USD) for which to return price info. (str)

Returns The API response parsed into a dict

token_rankings_historical(***kwargs*) → Dict
Retrieves the top ranked tokens by a specific metric, with a lookback window.

Useful for viewing token trends.

Key direction The sort order in which tokens are ranked (ascending or descending). (str)

Key sortType The metric used to rank the tokens (changeInPrice, currentPrice, marketCap, tokenVelocity, transactionVolume & uniqueAddresses). (str)

Key topN Number denominating top ranking tokens to return. Example: If given "5", results with return top 5 token rankings for past timeframe, where there are 5 results per day. (str)

Returns The API response parsed into a dict

token_rankings_latest(***kwargs*) → Dict
Retrieves the top ranked tokens by a specific metric.

Key direction The sort order in which tokens are ranked (ascending or descending). (str)

Key sortType The metric used to rank the tokens (changeInPrice, currentPrice, marketCap, tokenVelocity, transactionVolume & uniqueAddresses). (str)

Key timeInterval The time interval to return the historical data in: by day (days) or by hour (hours). (str)

Key type The type(s) of tokens to include in the rankings (erc20, erc721, erc777, erc884, erc998) (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

trade_pairs_historical(*pair: str*, ***kwargs*) → Dict
Retrieves the historical (time series) trade data for the specified pair.

Note: This endpoint returns a max of 6 months of historical data. In order to get more than 6 months you must use the startDate & endDate parameters to move the time frame window to get the next n days/months of data.

Parameters **pair** – The asset pair to look up

Key exchange The exchange(s) for which to retrieve market trades. (str)

Key startDate Filter by trades after this date. (int)

Key endDate Filter by trades before this date. (int)

Returns The API response parsed into a dict

trades (***kwargs*) → Dict
Retrieves the list of all available market trade data sets.

Key exchange Only return data for the given exchanges (comma separated). (str)

Returns The API response parsed into a dict

uniswap_liquidity (*pair: str*, ***kwargs*) → Dict
Retrieves the Uniswap-specific ether and token balance pairs over time.

Note: This endpoint returns a max of 6 months of historical data. In order to get more than 6 months you must use the startDate & endDate parameters to move the time frame window to get the next n days/months of data.

Parameters **pair** – The asset pair to look up

Key timeFormat The timestamp format to use for the timestamps: milliseconds/ms or iso/iso8611. (str)

Key startDate Filter to liquidity changes after this date (int)

Key endDate Filter to liquidity changes before this date (int)

Returns The API response parsed into a dict

web3data.handlers.signature

This module contains the address subhandler.

class web3data.handlers.signature.**SignatureHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)
Bases: *web3data.handlers.base.BaseHandler*

The subhandler for signature-related queries.

details (*signature: str*) → Dict
Retrieves detailed information about the specified signature hash.

Parameters **signature** – The signature string to look up

Returns The API response parsed into a dict

web3data.handlers.token

This module contains the address subhandler.

class web3data.handlers.token.**TokenHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)
Bases: *web3data.handlers.base.BaseHandler*

The subhandler for token-related queries.

holders_historical (*address: str*, ***kwargs*) → Dict
Retrieves the historical (time series) token holders for the specified token address.

Parameters **address** – The token's smart contract address

Key holderAddresses A comma separated list of addresses for which the historical holdings are to be retrieved. (str)

Key timeFormat The time format to use for the timestamps: milliseconds/ms or iso/iso861. (str)

Key timeFrame The time frame to return the historical data in: by day (1d, 2d, ..., all), by hour (1h, 2h, ..., 72h), by minute (1m, 2m, ..., 360m) or by tick (1t, 10t, ..., 1000t). (str)

Key includePrice Indicates whether or not to include price data with the results. (bool)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

holders_latest (*address: str*, ***kwargs*) → Dict

Retrieves the token holders for the specified address.

Parameters **address** – The token's smart contract address

Key numTokens Filter by token holders who own the specified amount of tokens. (int)

Key numTokensGt Filter by token holders who own more than the specified amount of tokens (int)

Key numTokensGte Filter by token holders who own more than or equal to the specified amount of tokens (int)

Key numTokensLt Fitler by token holders who own less than the specified amount of tokens (int)

Key numTokensLte Filter by token holders who own less than or equal to the specified amount of tokens (int)

Key tokenAddress Filter by token holders for this token (mandatory) (str)

Key timestampGt Filter by token holders who started holding the token after the specified date (int)

Key timestampGte Filter by token holders who started holding the token after or equal to the specified date (int)

Key timestampLt Filter by token holders who started holding the token before the specified date (int)

Key timestampLte Filter by token holders who started holding the token before or equal to the specified date (int)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key page The page number to return. (int)

Key size Number of records per page (int)

Returns The API response parsed into a dict

supply_historical (*address: str*, ***kwargs*) → Dict

Retrieves the historical token supplies (and derivatives) for the specified address.

Parameters

- **address** – The token's smart contract address
- **kwargs** – Additional query parameter options

Key timeFormat The time format to use for the timestamps: milliseconds/ms or iso/iso861. (str)

Key timeInterval The time interval to return the historical data in: by day (days) or by hour (hours). (str)

Key startDate Filter by token prices after this date - note that the interval can not exceed 6 months (d), or 30 days (h). (int)

Key endDate Filter by token prices before this date - note that the interval can not exceed 6 months (d), or 30 days (h). (int)

Returns The API response parsed into a dict

supply_latest(*address: str*) → Dict
Retrieves the latest token supplies (and derivatives) for the specified address.

Parameters **address** – The token's smart contract address

Returns The API response parsed into a dict

transfers(*address: str*, ***kwargs*) → Dict
Retrieves all token transfers involving the specified address.

Parameters **address** – The token's smart contract address

Key amount Filter by token transfers which value is equal to this amount. (int)

Key amountGt Filter by token transfers which value is greater than this amount. (int)

Key amountGte Filter by token transfers which value is greater than or equal to this amount. (int)

Key amountLt Filter by token transfers which value is less than this amount. (int)

Key amountLte Filter by token transfers which value is less than or equal to this amount (int)

Key blockNumber Filter by token transfers with this block number. (int)

Key recipientAddress Filter by token transfers which recipient is the specified address. (str)

Key senderAddress Filter by token transfers which sender is the specified address. (str)

Key startDate Filter by token transfers which happened after this date. (int)

Key endDate Filter by token transfers which happened before this date. (int)

Key tokenAddress Filter by token transfers for this token. (str)

Key transactionHash Filter by token transfers for this transaction hash. (str)

Key page The page number to return. (int)

Key size Number of records per page. (int)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information. Options: usd, btc. Only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

velocity(*address: str*, ***kwargs*) → Dict
Retrieves the historical velocity for the specified address.

Parameters **address** – The token's smart contract address

Key timeFormat The time format to use with the timestamps: milliseconds/ms or iso/iso8611 (str)

Key timeFrame The time frame to return the historical data in: by day (1d, 2d, ..., all), by hour (1h, 2h, ..., 72h) or by minute (1m, 2m, ..., 360m) (str)

Returns The API response parsed into a dict

volume(*address: str*, ***kwargs*) → Dict

Retrieves the historical number of transfers for the specified address.

Parameters **address** – The token's smart contract address

Key timeFormat The time format to use with the timestamps: milliseconds/ms or iso/iso8611 (str)

Key timeFrame The time frame to return the historical data in: by day (1d, 2d, ..., all), by hour (1h, 2h, ..., 72h) or by minute (1m, 2m, ..., 360m) (str)

Returns The API response parsed into a dict

web3data.handlers.transaction

This module contains the address subhandler.

class web3data.handlers.transaction.**TransactionHandler**(*initial_headers: Dict[str, str]*, *chain:* web3data.chains.Chains)

Bases: *web3data.handlers.base.BaseHandler*

The subhandler for transaction-related queries.

find(***kwargs*) → Dict

Retrieves all transactions matching the specified filters.

Key status Filter by the status of the transactions to retrieve (all, completed, failed, pending). (str)

Key startDate Filter by transactions executed after this date. Note that the interval can not exceed 1 minute (startDate and endDate should be both specified, or both empty) (int)

Key endDate Filter by transactions executed before this date. Note that the interval can not exceed 1 minute (startDate and endDate should be both specified, or both empty). (int)

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Key size The number of results to return. (int)

Key includeFunctions Indicates whether or not to include log information for each transaction, if available (false|true) (bool)

Key includeLogs Indicates whether or not to include price information (false|true) (bool)

Key includeTokenTransfers Indicates whether or not to include token transfers information for each transaction, if available (false|true) (bool)

Returns The API response parsed into a dict

gas_percentiles(***kwargs*) → Dict
Retrieves the latest gas price percentiles for the transactions.

Key numBlocks Number of past blocks on which to base the percentiles. (int)

Returns The API response parsed into a dict

gas_predictions() → Dict
Retrieves the latest gas predictions for the transactions.

Returns The API response parsed into a dict

information(*tx_hash: str*, ***kwargs*) → Dict
Retrieves the transaction information for the specified hash.

Parameters **tx_hash** – The transaction hash to fetch information for

Key validationMethod The validation method to be added to the response: none, basic, full. Default: none. (str)

Key includePrice Indicates whether or not to include price data with the results. Options: true, false. (bool)

Key currency The currency of the price information (usd or btc.) - only used in conjunction with includePrice. (str)

Returns The API response parsed into a dict

metrics() → Dict
Get metrics for recent confirmed transactions for a given blockchain.

Returns The API response parsed into a dict

token_transfers(*tx_hash: str*) → Dict
Retrieves the token transfers that took place in the specified transaction.

Parameters **tx_hash** – The transaction hash to fetch information for

Returns The API response parsed into a dict

volume(***kwargs*) → Dict
Retrieves the historical (time series) volume of transactions.

Parameters **kwargs** – Additional query parameter options

Key timeFormat The time format to use for the timestamps: milliseconds/ms or iso/iso861. (str)

Key timeFrame The time frame to return the historical data in: by day (1d, 2d, ..., all), by hour (1h, 2h, ..., 72h) or by minute (1m, 2m, ..., 360m) (str)

Returns The API response parsed into a dict

web3data.handlers.websocket

This module implements the websocket handler.

class web3data.handlers.websocket.**WebsocketHandler**(*api_key: str*, *blockchain_id: str*, *url: str = None*)

Bases: object

The subhandler for websocket-related queries.

on_close(*ws*)
A user-defined handler for websocket close events.

Parameters **ws** – The websocket client instance

on_error(*ws*, *error*)
A user-defined handler for websocket errors.

Parameters

- **ws** – The websocket client instance
- **error** – The error message

on_open(*ws*)
A user-defined handler for websocket open events.

Parameters **ws** – The websocket client instance

register(*params: Union[Iterable[str], str]*, *callback=None*)
Register a new event to listen for and its callback.

This will subscribe to the given event identifiers and execute the provided callback function once the specified event is coming in. Please note that the listening and callback-handling routine only starts once the websocket client is running.

The callback function should take two parameters: `ws` and `message`. The first parameter is the websocket client instance, which allows the user to update the client context. The latter argiment is the message, deserialized from the JSON object received by the websocket server.

Parameters

- **params** – The event to subscribe to
- **callback** – The callback function to execute

run(***kwargs*)
Run the websocket listening loop.

This is a blocking endless loop that will send the subscription events to the websocket server, handle the responses, and then distribute the incoming messages across the registered callbacks.

Any keyword arguments passed to this method are passed on to the websocket client's `run_forever` method. Please consult the project's documentation for more details: https://pypi.org/project/websocket_client/

Parameters **kwargs** – Additional arguments to pass to the websocket client

unregister(*external_id*)
Unregister a subscription from the websocket server.

Given an external ID (i.e. the subscription ID), this will remove the subscription data including locally stored payloads and identifiers. It will also trigger an unsubscribe message for the subscription being sent to the websocket server.

Parameters **external_id** – The subscription ID to remove

Module contents

This package defines the sub-handlers and the main one.

4.2 Module contents

Top-level package for web3data-py.

CHAPTER

FIVE

CONTRIBUTING

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/dmuhs/web3data-py/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

web3data-py could always use more documentation, whether as part of the official web3data-py 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/dmuhs/web3data-py/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 *web3data* for local development.

1. Fork the *web3data* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/web3data-py.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 web3data-py
$ cd -py/
$ 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:

```
$ flake8 web3data tests
$ python setup.py test or pytest
$ make test
```

   To get flake8, 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 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/dmuhs/web3data-py/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 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.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Dominik Muhs <dmuhs@protonmail.ch>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.7 (2021-02-10)

- Add deprecation warning for Uniswap liquidity endpoint

7.2 0.1.6 (2021-01-25)

- Add CSV format switch support
- Various dependency upgrades

7.3 0.1.5 (2020-05-22)

- Include missing files in sdist
- Update pip to 20.1.1
- Update bumpversion to 0.6.0
- Update flake8 to 3.8.1
- Update pytest to 5.4.2
- Update requests-mock to 1.8.0

7.4 0.1.4 (2020-04-28)

- add support for RPC endpoints
- add support for websocket endpoints
- add examples for rpc and websocket calls
- update coverage to 5.1
- update sphinx to 3.0.3

7.5 0.1.3 (2020-03-16)

- update coverage from 4.5.4 to 5.0.3
- update flake8 from 3.7.8 to 3.7.9
- update pip from 19.2.3 to 20.0.2
- update pytest from 4.6.5 to 5.4.1
- update pytest-runner from 5.1 to 5.2
- update sphinx from 1.8.5 to 2.4.4
- update twine from 1.14.0 to 3.1.1
- update watchdog from 0.9.0 to 0.10.2
- update wheel from 0.33.6 to 0.34.2

7.6 0.1.1 + 0.1.2 (2020-03-15)

Add minor documentation, markup, and package publishing fixes

7.7 0.1.0 (2020-03-15)

First release on PyPI

7.8 0.0.1 (2020-03-13)

First implementation for the Amberdata developer challenge

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

U

V

W

Z