
python-binance Documentation

Release 0.2.0

Sam McHardy

Jan 26, 2018

Contents

1	Features	3
2	Quick Start	5
3	Donate	7
4	Other Exchanges	9
4.1	Contents	9
4.2	Index	60
	Python Module Index	61

This is an unofficial Python wrapper for the [Binance exchange REST API v1/3](#). I am in no way affiliated with Binance, use at your own risk.

If you came here looking for the [Binance exchange](#) to purchase cryptocurrencies, then [go here](#). If you want to automate interactions with Binance stick around.

Source code <https://github.com/sammchardy/python-binance>

Documentation <https://python-binance.readthedocs.io/en/latest/>

Binance API Telegram https://t.me/binance_api_english

Blog with examples <https://sammchardy.github.io>

Make sure you update often and check the [Changelog](#) for new features and bug fixes.

CHAPTER 1

Features

- Implementation of all General, Market Data and Account endpoints.
- Simple handling of authentication
- No need to generate timestamps yourself, the wrapper does it for you
- Response exception handling
- Websocket handling with reconnection and multiplexed connections
- Symbol Depth Cache
- Historical Kline/Candle fetching function
- Withdraw functionality
- Deposit addresses

CHAPTER 2

Quick Start

Register an account with Binance.

Generate an API Key and assign relevant permissions.

```
pip install python-binance
```

```
from binance.client import Client
client = Client(api_key, api_secret)

# get market depth
depth = client.get_order_book(symbol='BNBBTC')

# place a test market buy order, to place an actual order use the create_order_
↪function
order = client.create_test_order(
    symbol='BNBBTC',
    side=Client.SIDE_BUY,
    type=Client.ORDER_TYPE_MARKET,
    quantity=100)

# get all symbol prices
prices = client.get_all_tickers()

# withdraw 100 ETH
# check docs for assumptions around withdrawals
from binance.exceptions import BinanceAPIException, BinanceWithdrawException
try:
    result = client.withdraw(
        asset='ETH',
        address='<eth_address>',
        amount=100)
except BinanceAPIException as e:
    print(e)
except BinanceWithdrawException as e:
    print(e)
```

```
else:
    print("Success")

# fetch list of withdrawals
withdraws = client.get_withdraw_history()

# fetch list of ETH withdrawals
eth_withdraws = client.get_withdraw_history(asset='ETH')

# get a deposit address for BTC
address = client.get_deposit_address(asset='BTC')

# start aggregated trade websocket for BNBBTC
def process_message(msg):
    print("message type: {}".format(msg['e']))
    print(msg)
    # do something

from binance.websockets import BinanceSocketManager
bm = BinanceSocketManager(client)
bm.start_aggtrade_socket('BNBBTC', process_message)
bm.start()

# get historical kline data from any date range

# fetch 1 minute klines for the last day up until now
klines = client.get_historical_klines("BNBBTC", Client.KLINE_INTERVAL_1MINUTE, "1 day_
↪ago UTC")

# fetch 30 minute klines for the last month of 2017
klines = client.get_historical_klines("ETHBTC", Client.KLINE_INTERVAL_30MINUTE, "1_
↪Dec, 2017", "1 Jan, 2018")

# fetch weekly klines since it listed
klines = client.get_historical_klines("NEOBTC", KLINE_INTERVAL_1WEEK, "1 Jan, 2017")
```

For more check out the documentation.

CHAPTER 3

Donate

If this library helped you out feel free to donate.

- ETH: 0xD7a7fDdCfA687073d7cC93E9E51829a727f9fE70
- LTC: LPC5vw9ajR1YndE1hYVeo3kJ9LdHjcRCUZ
- NEO: AVJB4ZgN7VgSUtArCt94y7ZYT6d5NDfpBo
- BTC: 1Dknp6L6oRZrHDECRedihPzx2sSfmvEBys

CHAPTER 4

Other Exchanges

If you use [Quoinex](#) or [Qryptos](#) check out my [python-quoine](#) library.

If you use [Kucoin](#) check out my [python-kucoin](#) library.

If you use [IDEX](#) check out my [python-idex](#) library.

4.1 Contents

4.1.1 Getting Started

Installation

`python-binance` is available on [PYPI](#). Install with `pip`:

```
pip install python-binance
```

Windows

If you see errors building Twisted indication Microsoft Visual C++ is required you may need to install the Visual C++ Build Tools refer to the [Python Wiki on Windows Compilers](#) for your relevant version.

Register on Binance

Firstly [register](#) an account with [Binance](#).

Generate an API Key

To use signed account methods you are required to [create an API Key](#).

Initialise the client

Pass your API Key and Secret

```
from binance.client import Client
client = Client(api_key, api_secret)
```

Making API Calls

Every method supports the passing of arbitrary parameters via keyword matching those in the 'Binance API documentation' <<https://github.com/binance-exchange/binance-official-api-docs>>_. These keyword arguments will be sent directly to the relevant endpoint.

Each API method returns a dictionary of the JSON response as per the [Binance API documentation](#). The docstring of each method in the code references the endpoint it implements.

The Binance API documentation references a *timestamp* parameter, this is generated for you where required.

Some methods have a *recvWindow* parameter for [timing security](#), see [Binance documentation](#).

API Endpoints are rate limited by Binance at 20 requests per second, ask them if you require more.

API Rate Limit

Check the `get_exchange_info()` call for up to date rate limits.

At the current time Binance rate limits are:

- 1200 requests per minute
- 10 orders per second
- 100,000 orders per 24hrs

Some calls have a higher weight than others especially if a call returns information about all symbols. Read the '[official Binance documentation](#)' <<https://github.com/binance-exchange/binance-official-api-docs>>_ for specific information.

Requests Settings

python-binance uses the [requests](#) library.

You can set custom requests parameters for all API calls when creating the client.

```
client = Client("api-key", "api-secret", {"verify": False, "timeout": 20})
```

You may also pass custom requests parameters through any API call to override default settings or the above settings specify new ones like the example below.

```
# this would result in verify: False and timeout: 5 for the get_all_orders call
client = Client("api-key", "api-secret", {"verify": False, "timeout": 20})
client.get_all_orders(symbol='BNBBTC', requests_params={'timeout': 5})
```

Check out the [requests documentation](#) for all options.

Proxy Settings

You can use the Requests Settings method above

```
proxies = {
    'http': 'http://10.10.1.10:3128',
    'https': 'http://10.10.1.10:1080'
}

# in the Client instantiation
client = Client("api-key", "api-secret", {'proxies': proxies})

# or on an individual call
client.get_all_orders(symbol='BNBBTC', requests_params={'proxies': proxies})
```

Or set an environment variable for your proxy if required to work across all requests.

An example for Linux environments from the [requests Proxies documentation](#) is as follows.

```
$ export HTTP_PROXY="http://10.10.1.10:3128"
$ export HTTPS_PROXY="http://10.10.1.10:1080"
```

For Windows environments

```
C:\>set HTTP_PROXY=http://10.10.1.10:3128
C:\>set HTTPS_PROXY=http://10.10.1.10:1080
```

4.1.2 Enumerated Types

Binance defines Enumerated Types for Order Types, Order Side, Time in Force, Order response and Kline intervals these are found on *binance.client.Client*.

```
SYMBOL_TYPE_SPOT = 'SPOT'

ORDER_STATUS_NEW = 'NEW'
ORDER_STATUS_PARTIALLY_FILLED = 'PARTIALLY_FILLED'
ORDER_STATUS_FILLED = 'FILLED'
ORDER_STATUS_CANCELED = 'CANCELED'
ORDER_STATUS_PENDING_CANCEL = 'PENDING_CANCEL'
ORDER_STATUS_REJECTED = 'REJECTED'
ORDER_STATUS_EXPIRED = 'EXPIRED'

KLINE_INTERVAL_1MINUTE = '1m'
KLINE_INTERVAL_2MINUTE = '3m'
KLINE_INTERVAL_5MINUTE = '5m'
KLINE_INTERVAL_15MINUTE = '15m'
KLINE_INTERVAL_30MINUTE = '30m'
KLINE_INTERVAL_1HOUR = '1h'
KLINE_INTERVAL_2HOUR = '2h'
KLINE_INTERVAL_4HOUR = '4h'
KLINE_INTERVAL_6HOUR = '6h'
KLINE_INTERVAL_8HOUR = '8h'
KLINE_INTERVAL_12HOUR = '12h'
KLINE_INTERVAL_1DAY = '1d'
KLINE_INTERVAL_3DAY = '3d'
KLINE_INTERVAL_1WEEK = '1w'
```

```
KLINE_INTERVAL_1MONTH = '1M'

SIDE_BUY = 'BUY'
SIDE_SELL = 'SELL'

ORDER_TYPE_LIMIT = 'LIMIT'
ORDER_TYPE_MARKET = 'MARKET'
ORDER_TYPE_STOP_LOSS = 'STOP_LOSS'
ORDER_TYPE_STOP_LOSS_LIMIT = 'STOP_LOSS_LIMIT'
ORDER_TYPE_TAKE_PROFIT = 'TAKE_PROFIT'
ORDER_TYPE_TAKE_PROFIT_LIMIT = 'TAKE_PROFIT_LIMIT'
ORDER_TYPE_LIMIT_MAKER = 'LIMIT_MAKER'

TIME_IN_FORCE_GTC = 'GTC'
TIME_IN_FORCE_IOC = 'IOC'
TIME_IN_FORCE_FOK = 'FOK'

ORDER_RESP_TYPE_ACK = 'ACK'
ORDER_RESP_TYPE_RESULT = 'RESULT'
ORDER_RESP_TYPE_FULL = 'FULL'
```

For Websocket Depth these are found on *binance.websockets.BinanceSocketManager*

```
WEBSOCKET_DEPTH_5 = '5'
WEBSOCKET_DEPTH_10 = '10'
WEBSOCKET_DEPTH_20 = '20'
```

To use in your code reference either `binance.client.Client` or `binance.websockets.BinanceSocketManager`

```
from binance.client import Client
from binance.websockets import BinanceSocketManager

side = Client.SIDE_BUY
```

4.1.3 General Endpoints

Ping the server

```
client.ping()
```

Get the server time

```
time_res = client.get_server_time()
```

Get Exchange Info

```
info = client.get_exchange_info()
```

Get Symbol Info

Get the exchange info for a particular symbol


```
info = client.get_symbol_info('BNBBTC')
```

Get Current Products

This call is deprecated, use the above Exchange Info call

```
products = client.get_products()
```

4.1.4 Market Data Endpoints

Get Market Depth

```
depth = client.get_order_book(symbol='BNBBTC')
```

Get Recent Trades

```
trades = client.get_recent_trades(symbol='BNBBTC')
```

Get Historical Trades

```
trades = client.get_historical_trades(symbol='BNBBTC')
```

Get Aggregate Trades

```
trades = client.get_aggregate_trades(symbol='BNBBTC')
```

Get Kline/Candlesticks

```
candles = client.get_klines(symbol='BNBBTC', interval=Client.KLINE_INTERVAL_30MINUTE)
```

Get Historical Kline/Candlesticks

Fetch klines for any date range and interval

```
# fetch 1 minute klines for the last day up until now
klines = client.get_historical_klines("BNBBTC", Client.KLINE_INTERVAL_1MINUTE, "1 day_
↳ ago UTC")

# fetch 30 minute klines for the last month of 2017
klines = client.get_historical_klines("ETHBTC", Client.KLINE_INTERVAL_30MINUTE, "1_
↳ Dec, 2017", "1 Jan, 2018")

# fetch weekly klines since it listed
klines = client.get_historical_klines("NEOBTCT", KLINE_INTERVAL_1WEEK, "1 Jan, 2017")
```

Get 24hr Ticker

```
tickers = client.get_ticker()
```

Get All Prices

Get last price for all markets.

```
prices = client.get_all_tickers()
```

Get Orderbook Tickers

Get first bid and ask entry in the order book for all markets.

```
tickers = client.get_orderbook_tickers()
```

4.1.5 Account Endpoints

Orders

Order Validation

Binance has a number of rules around symbol pair orders with validation on minimum price, quantity and total order value.

Read more about their specifics in the [Filters](#) section of the official API.

It can be helpful to format the output using the following snippet

```
amount = 0.000234234
precision = 5
amt_str = "{:0.0{}}f".format(amount, precision)
```

Fetch all orders

```
orders = client.get_all_orders(symbol='BNBBTC', limit=10)
```

Place an order

Place an order

Use the `create_order` function to have full control over creating an order .. code:: python

```
from binance.enums import *
order = client.create_order(
    symbol='BNBBTC', side=SIDE_BUY, type=ORDER_TYPE_LIMIT, timeIn-
    Force=TIME_IN_FORCE_GTC, quantity=100, price='0.00001')
```

Place a limit order

Use the helper functions to easily place a limit buy or sell order

```
order = client.order_limit_buy(
    symbol='BNBBTC',
    quantity=100,
    price='0.00001')

order = client.order_limit_sell(
    symbol='BNBBTC',
    quantity=100,
    price='0.00001')
```

Place a market order

Use the helper functions to easily place a market buy or sell order

```
order = client.order_market_buy(
    symbol='BNBBTC',
    quantity=100)

order = client.order_market_sell(
    symbol='BNBBTC',
    quantity=100)
```

Place a test order

Creates and validates a new order but does not send it into the exchange.

```
from binance.enums import *
order = client.create_test_order(
    symbol='BNBBTC',
    side=SIDE_BUY,
    type=ORDER_TYPE_LIMIT,
    timeInForce=TIME_IN_FORCE_GTC,
    quantity=100,
    price='0.00001')
```

Check order status

```
order = client.get_order(
    symbol='BNBBTC',
    orderId='orderId')
```

Cancel an order

```
result = client.cancel_order(
    symbol='BNBBTC',
    orderId='orderId')
```

Get all open orders

```
orders = client.get_open_orders(symbol='BNBBTC')
```

Get all orders

```
orders = client.get_all_orders(symbol='BNBBTC')
```

Account

Get account info

```
info = client.get_account()
```

Get asset balance

```
balance = client.get_asset_balance(asset='BTC')
```

Get account status

```
status = client.get_account_status()
```

Get trades

```
trades = client.get_my_trades(symbol='BNBBTC')
```

4.1.6 Websockets

Sockets are handled through a Socket Manager [BinanceSocketManager](#).

Multiple socket connections can be made through the manager.

Only one instance of each socket type will be created, i.e. only one BNBBTC Depth socket can be created and there can be both a BNBBTC Depth and a BNBBTC Trade socket open at once.

When creating socket connections a callback function is passed which receives the messages.

Messages are received as dictionary objects relating to the message formats defined in the [Binance WebSocket API documentation](#).

Websockets are setup to reconnect with a maximum of 5 retries.

Websocket Usage

Create the manager like so, passing the API client.

```
from binance.websockets import BinanceSocketManager
bm = BinanceSocketManager(client)
# start any sockets here, i.e a trade socket
conn_key = bm.start_trade_socket('BNBBTC', process_message)
# then start the socket manager
bm.start()
```

A callback to process messages would take the format

```
def process_message(msg):
    print("message type: {}".format(msg['e']))
    print(msg)
    # do something
```

Websocket Errors

If the websocket is disconnected and is unable to reconnect a message is sent to the callback to indicate this. The format is

```
{
    'e': 'error',
    'm': 'Max reconnect retries reached'
}

# check for it like so
def process_message(msg):
    if msg['e'] == 'error':
        # close and restart the socket
    else:
        # process message normally
```

Multiplex Socket

Create a socket combining multiple streams.

These streams can include the depth, kline, ticker and trade streams but not the user stream which requires extra authentication.

Symbols in socket name must be lowercase i.e `bnbbtc@aggTrade`, `neobtc@ticker`

See the [Binance Websocket Streams API documentation](#) for details on socket names.

```
def process_m_message(msg):
    print("stream: {} data: {}".format(msg['stream'], msg['data']))

# pass a list of stream names
conn_key = bm.start_multiplex_socket(['bnbbtc@aggTrade', 'neobtc@ticker'], process_m_
↪message)
```

Depth Socket

Depth sockets have an optional depth parameter to receive partial book rather than a diff response. By default this the diff response is returned. Valid depth values are 5, 10 and 20 and [defined as enums](#).

```
# depth diff response
diff_key = bm.start_depth_socket('BNBBTC', process_message)

# partial book response
partial_key = bm.start_depth_socket('BNBBTC', process_message, ↵
↪depth=BinanceSocketManager.WEBSOCKET_DEPTH_5)
```

Kline Socket

Kline sockets have an optional interval parameter. By default this is set to 1 minute. Valid interval values are [defined as enums](#).

```
from binance.enums import *
conn_key = bm.start_kline_socket('BNBBTC', process_message, interval=KLINE_INTERVAL_
↪30MINUTE)
```

Aggregated Trade Socket

```
conn_key = bm.start_aggttrade_socket('BNBBTC', process_message)
```

Trade Socket

```
conn_key = bm.start_trade_socket('BNBBTC', process_message)
```

Symbol Ticker Socket

```
conn_key = bm.start_symbol_ticker_socket('BNBBTC', process_message)
```

Ticker Socket

```
conn_key = bm.start_ticker_socket(process_message)
```

User Socket

This watches for 3 different user events

- Account Update Event
- Order Update Event
- Trade Update Event

The Manager handles keeping the socket alive.

```
bm.start_user_socket(process_message)
```

Close a Socket

To close an individual socket call the *stop_socket* function. This takes a *conn_key* parameter which is returned when starting the socket.

```
bm.stop_socket(conn_key)
```

To stop all sockets and end the manager call *close* after doing this a *start* call would be required to connect any new sockets.

```
bm.close()
```

Close and exit program

Websockets utilise a reactor loop from the Twisted library. Using the *close* method above will close the websocket connections but it won't stop the reactor loop so your code may not exit when you expect.

If you do want to exit then use the *stop* method from reactor like below.

```
from twisted.internet import reactor

# program code here

# when you need to exit
reactor.stop()
```

4.1.7 Depth Cache

To follow the depth cache updates for a symbol use the *DepthCacheManager*

Create the manager like so, passing the api client, symbol and an optional callback function.

```
from binance.depthcache import DepthCacheManager
dcm = DepthCacheManager(client, 'BNBBTC', callback=process_depth)
```

The callback function receives the current *DepthCache* object which allows access to a pre-sorted list of bids or asks able to be filtered as required.

Access the symbol value from the *depth_cache* object in case you have multiple caches using the same callback.

By default the depth cache will fetch the order book via REST request every 30 minutes. This duration can be changed by using the *refresh_interval* parameter. To disable the refresh pass 0 or None. The socket connection will stay open receiving updates to be replayed once the full order book is received.

Websocket Errors

If the underlying websocket is disconnected and is unable to reconnect None is returned for the *depth_cache* parameter.

Examples

```
# 1 hour interval refresh
dcm = DepthCacheManager(client, 'BNBBTC', callback=process_depth, refresh_
→interval=60*60)

# disable refreshing
dcm = DepthCacheManager(client, 'BNBBTC', callback=process_depth, refresh_interval=0)
```

```
def process_depth(depth_cache):
    if depth_cache is not None:
        print("symbol {}".format(depth_cache.symbol))
        print("top 5 bids")
        print(depth_cache.get_bids()[ :5])
        print("top 5 asks")
        print(depth_cache.get_asks()[ :5])
    else:
        # depth cache had an error and needs to be restarted
```

At any time the current *DepthCache* object can be retrieved from the *DepthCacheManager*

```
depth_cache = dcm.get_depth_cache()
if depth_cache is not None:
    print("symbol {}".format(depth_cache.symbol))
    print("top 5 bids")
    print(depth_cache.get_bids()[ :5])
    print("top 5 asks")
    print(depth_cache.get_asks()[ :5])
else:
    # depth cache had an error and needs to be restarted
```

To stop the *DepthCacheManager* from returning messages use the *close* method. This will close the internal websocket and this instance of the *DepthCacheManager* will not be able to be used again.

```
dcm.close()
```

4.1.8 Withdraw Endpoints

Place a withdrawal

Make sure you enable Withdrawal permissions for your API Key to use this call.

You must have withdrawn to the address through the website and approved the withdrawal via email before you can withdraw using the API.

Raises a *BinanceWithdrawException* if the withdraw fails.

```
from binance.exceptions import BinanceAPIException, BinanceWithdrawException
try:
    # name parameter will be set to the asset value by the client if not passed
    result = client.withdraw(
        asset='ETH',
        address='<eth_address>',
        amount=100)
except BinanceAPIException as e:
    print(e)
```



```

except BinanceWithdrawException as e:
    print(e)
else:
    print("Success")

# passing a name parameter
result = client.withdraw(
    asset='ETH',
    address='<eth_address>',
    amount=100,
    name='Withdraw')

# if the coin requires a extra tag or name such as XRP or XMR then pass an_
↪ `addressTag` parameter.
result = client.withdraw(
    asset='XRP',
    address='<xrp_address>',
    addressTag='<xrp_address_tag>',
    amount=10000)

```

Fetch deposit history

```

deposits = client.get_deposit_history()
btc_deposits = client.get_deposit_history(asset='BTC')

```

Fetch withdraw history

```

withdraws = client.get_withdraw_history()
btc_withdraws = client.get_withdraw_history(asset='BTC')

```

Get deposit address

```

address = client.get_deposit_address(asset='BTC')

```

4.1.9 Helper Functions

`binance.helpers`
alias of `binance.helpers`

4.1.10 Exceptions

BinanceResponseException

Raised if a non JSON response is returned

BinanceAPIException

On an API call error a `binance.exceptions.BinanceAPIException` will be raised.

The exception provides access to the

- `status_code` - response status code
- `response` - response object
- `code` - Binance error code
- `message` - Binance error message
- `request` - request object if available

```
try:
    client.get_all_orders()
except BinanceAPIException as e:
    print e.status_code
    print e.message
```

BinanceOrderException

When placing an order parameters are validated to check they fit within the [Binance Trading Rules](#).

The following exceptions extend *BinanceOrderException*.

BinanceOrderMinAmountException

Raised if the specified amount isn't a multiple of the trade minimum amount.

BinanceOrderMinPriceException

Raised if the price is lower than the trade minimum price.

BinanceOrderTotalPriceException

Raised if the total is lower than the trade minimum total.

BinanceOrderUnknownSymbolException

Raised if the symbol is not recognised.

BinanceOrderInactiveSymbolException

Raised if the symbol is inactive.

BinanceWithdrawException

Raised if the withdraw fails.

4.1.11 FAQ

Q: Why do I get “Timestamp for this request is not valid”

A: This occurs in 2 different cases.

The timestamp sent is outside of the serverTime - recvWindow value The timestamp sent is more than 1000ms ahead of the server time

Check that your system time is in sync. See [this issue](#) for some sample code to check the difference between your local time and the Binance server time.

Q: Why do I get “Signature for this request is not valid”

A1: One of your parameters may not be in the correct format.

Check recvWindow is an integer and not a string.

A2: You may need to regenerate your API Key and Secret

A3: You may be attempting to access the API from a Chinese IP address, these are now restricted by Binance.

Q: Twisted won't install using pip on Windows

A: If you see errors building Twisted indication Microsoft Visual C++ is required you may need to install the Visual C++ Build Tools refer to the [Python Wiki on Windows Compilers](#) for your relevant version.

4.1.12 Changelog

v0.6.2 - 2018-01-12

Fixes

- fixed handling Binance errors that aren't JSON objects

v0.6.1 - 2018-01-10

Fixes

- added missing dateparser dependency to setup.py
- documentation fixes

v0.6.0 - 2018-01-09

New version because why not.

Added

- get_historical_klines function to fetch klines for any date range
- ability to override requests parameters globally
- error on websocket disconnect
- example related to blog post

Fixes

- documentation fixes

v0.5.17 - 2018-01-08

Added

- check for name parameter in withdraw, set to asset parameter if not passed

Update

- Windows install error documentation

Removed

- reference to disable_validation in documentation

v0.5.16 - 2018-01-06

Added

- addressTag documentation to withdraw function
- documentation about requests proxy environment variables

Update

- FAQ for signature error with solution to regenerate API key
- change create_order to create_test_order in example

Fixed

- reference to BinanceAPIException in documentation

v0.5.15 - 2018-01-03

Fixed

- removed all references to WEBSOCKET_DEPTH_1 enum

v0.5.14 - 2018-01-02

Added

- Wait for depth cache socket to start
- check for sequential depth cache messages

Updated

- documentation around depth websocket and diff and partial responses

Removed

- Removed unused WEBSOCKET_DEPTH_1 enum
- removed unused libraries and imports

v0.5.13 - 2018-01-01

Fixed

- Signature invalid error

v0.5.12 - 2017-12-29

Added

- get_asset_balance helper function to fetch an individual asset's balance

Fixed

- added timeout to requests call to prevent hanging
- changed variable type to str for price parameter when creating an order
- documentation fixes

v0.5.11 - 2017-12-28

Added

- refresh interval parameter to depth cache to keep it fresh, set default at 30 minutes

Fixed

- watch depth cache socket before fetching order book to replay any messages

v0.5.10 - 2017-12-28

Updated

- updated dependencies certifi and cryptography to help resolve signature error

v0.5.9 - 2017-12-26

Fixed

- fixed websocket reconnecting, was no distinction between manual close or network error

v0.5.8 - 2017-12-25

Changed

- change symbol parameter to optional for get_open_orders function
- added listenKey parameter to stream_close function

Added

- get_account_status function that was missed

v0.5.7 - 2017-12-24

Changed

- change depth cache callback parameter to optional

Added

- note about stopping Twisted reactor loop to exit program

v0.5.6 - 2017-12-20

Added

- `get_symbol_info` function to simplify getting info about a particular symbol

v0.5.5 - 2017-12-19

Changed

- Increased default limit for order book on depth cache from 10 to 500

v0.5.4 - 2017-12-14

Added

- `symbol` property made public on `DepthCache` class

Changed

- Enums now also accessible from `binance.client.Client` and `binance.websockets.BinanceSocketManager`

v0.5.3 - 2017-12-09

Changed

- User stream refresh timeout from 50 minutes to 30 minutes
- User stream socket listen key change check simplified

v0.5.2 - 2017-12-08

Added

- `start_multiplex_socket` function to `BinanceSocketManager` to create multiplexed streams

v0.5.1 - 2017-12-06

Added

- `Close` method for `DepthCacheManager`

Fixes

- Fixed modifying array error message when closing the `BinanceSocketManager`

v0.5.0 - 2017-12-05

Updating to match new API documentation

Added

- Recent trades endpoint
- Historical trades endpoint
- Order response type option

- Check for invalid user stream listen key in socket to keep connected

Fixes

- Fixed exchange info endpoint as it was renamed slightly

v0.4.3 - 2017-12-04

Fixes

- Fixed stopping sockets where they were reconnecting
- Fixed websockets unable to be restarted after close
- Exception in parsing non-JSON websocket message

v0.4.2 - 2017-11-30

Removed

- Removed websocket update time as 0ms option is not available

v0.4.1 - 2017-11-24

Added

- Reconnecting websockets, automatic retry on disconnect

v0.4.0 - 2017-11-19

Added

- Get deposit address endpoint
- Upgraded withdraw endpoints to v3
- New exchange info endpoint with rate limits and full symbol info

Removed

- Order validation to return at a later date

v0.3.8 - 2017-11-17

Fixes

- Fix order validation for market orders
- WEBSOCKET_DEPTH_20 value, 20 instead of 5
- General tidy up

v0.3.7 - 2017-11-16

Fixes

- Fix multiple depth caches sharing a cache by initialising bid and ask objects each time

v0.3.6 - 2017-11-15

Fixes

- check if Reactor is already running

v0.3.5 - 2017-11-06

Added

- support for BNB market

Fixes

- fixed error if new market type is created that we don't know about

v0.3.4 - 2017-10-31

Added

- depth parameter to depth socket
- interval parameter to kline socket
- update time parameter for compatible sockets
- new enums for socket depth and update time values
- better websocket documentation

Changed

- Depth Cache Manager uses 0ms socket update time
- connection key returned when creating socket, this key is then used to stop it

Fixes

- General fixes

v0.3.3 - 2017-10-31

Fixes

- Fixes for broken tests

v0.3.2 - 2017-10-30

Added

- More test coverage of requests

Fixes

- Order quantity validation fix

v0.3.1 - 2017-10-29

Added

- Withdraw exception handler with translation of obscure error

Fixes

- Validation fixes

v0.3.0 - 2017-10-29

Added

- Withdraw endpoints
- Order helper functions

v0.2.0 - 2017-10-27

Added

- Symbol Depth Cache

v0.1.6 - 2017-10-25

Changes

- Upgrade to v3 signed endpoints
- Update function documentation

v0.1.5 - 2017-09-12

Changes

- Added get_all_tickers call
- Added get_orderbook_tickers call
- Added some FAQs

Fixes

- Fix error in enum value

v0.1.4 - 2017-09-06

Changes

- Added parameter to disable client side order validation

v0.1.3 - 2017-08-26

Changes

- Updated documentation

Fixes

- Small bugfix

v0.1.2 - 2017-08-25

Added

- Travis.CI and Coveralls support

Changes

- Validation for pairs using public endpoint

v0.1.1 - 2017-08-17

Added

- Validation for HSR/BTC pair

v0.1.0 - 2017-08-16

Websocket release

Added

- Websocket manager
- Order parameter validation
- Order and Symbol enums
- API Endpoints for Data Streams

v0.0.2 - 2017-08-14

Initial version

Added

- General, Market Data and Account endpoints

4.1.13 Binance API

client module

```
class binance.client.Client (api_key, api_secret, requests_params=None)
    Bases: object

    API_URL = 'https://api.binance.com/api'

    KLINE_INTERVAL_12HOUR = '12h'
```

```
KLINE_INTERVAL_15MINUTE = '15m'
KLINE_INTERVAL_1DAY = '1d'
KLINE_INTERVAL_1HOUR = '1h'
KLINE_INTERVAL_1MINUTE = '1m'
KLINE_INTERVAL_1MONTH = '1M'
KLINE_INTERVAL_1WEEK = '1w'
KLINE_INTERVAL_2HOUR = '2h'
KLINE_INTERVAL_30MINUTE = '30m'
KLINE_INTERVAL_3DAY = '3d'
KLINE_INTERVAL_3MINUTE = '3m'
KLINE_INTERVAL_4HOUR = '4h'
KLINE_INTERVAL_5MINUTE = '5m'
KLINE_INTERVAL_6HOUR = '6h'
KLINE_INTERVAL_8HOUR = '8h'
ORDER_RESP_TYPE_ACK = 'ACK'
ORDER_RESP_TYPE_FULL = 'FULL'
ORDER_RESP_TYPE_RESULT = 'RESULT'
ORDER_STATUS_CANCELED = 'CANCELED'
ORDER_STATUS_EXPIRED = 'EXPIRED'
ORDER_STATUS_FILLED = 'FILLED'
ORDER_STATUS_NEW = 'NEW'
ORDER_STATUS_PARTIALLY_FILLED = 'PARTIALLY_FILLED'
ORDER_STATUS_PENDING_CANCEL = 'PENDING_CANCEL'
ORDER_STATUS_REJECTED = 'REJECTED'
ORDER_TYPE_LIMIT = 'LIMIT'
ORDER_TYPE_LIMIT_MAKER = 'LIMIT_MAKER'
ORDER_TYPE_MARKET = 'MARKET'
ORDER_TYPE_STOP_LOSS = 'STOP_LOSS'
ORDER_TYPE_STOP_LOSS_LIMIT = 'STOP_LOSS_LIMIT'
ORDER_TYPE_TAKE_PROFIT = 'TAKE_PROFIT'
ORDER_TYPE_TAKE_PROFIT_LIMIT = 'TAKE_PROFIT_LIMIT'
PRIVATE_API_VERSION = 'v3'
PUBLIC_API_VERSION = 'v1'
SIDE_BUY = 'BUY'
SIDE_SELL = 'SELL'
SYMBOL_TYPE_SPOT = 'SPOT'
```

```
TIME_IN_FORCE_FOK = 'FOK'
TIME_IN_FORCE_GTC = 'GTC'
TIME_IN_FORCE_IOC = 'IOC'
WEBSITE_URL = 'https://www.binance.com'
WITHDRAW_API_URL = 'https://api.binance.com/wapi'
WITHDRAW_API_VERSION = 'v3'

__init__(api_key, api_secret, requests_params=None)
    Binance API Client constructor
```

Parameters

- **api_key** (*str.*) – Api Key
- **api_secret** (*str.*) – Api Secret
- **requests_params** (*dict.*) – optional - Dictionary of requests params to use for all calls

cancel_order (***params*)

Cancel an active order. Either orderId or origClientId must be sent.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#cancel-order-trade>

Parameters

- **symbol** (*str*) – required
- **orderId** (*int*) – The unique order id
- **origClientId** (*str*) – optional
- **newClientId** (*str*) – Used to uniquely identify this cancel. Automatically generated by default.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "symbol": "LTCBTC",
  "origClientId": "myOrder1",
  "orderId": 1,
  "clientId": "cancelMyOrder1"
}
```

Raises BinanceResponseException, BinanceAPIException

create_order (***params*)

Send in a new order

Any order with an icebergQty MUST have timeInForce set to GTC.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#new-order-trade>

Parameters

- **symbol** (*str*) – required
- **side** (*enum*) – required

- **type** (*enum*) – required
- **timeInForce** (*enum*) – required if limit order
- **quantity** (*decimal*) – required
- **price** (*str*) – required
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **icebergQty** (*decimal*) – Used with LIMIT, STOP_LOSS_LIMIT, and TAKE_PROFIT_LIMIT to create an iceberg order.
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

Response ACK:

```
{
  "symbol": "LTCBTC",
  "orderId": 1,
  "clientOrderId": "myOrder1" # Will be newClientOrderId
  "transactTime": 1499827319559
}
```

Response RESULT:

```
{
  "symbol": "BTCUSDT",
  "orderId": 28,
  "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
  "transactTime": 1507725176595,
  "price": "0.00000000",
  "origQty": "10.00000000",
  "executedQty": "10.00000000",
  "status": "FILLED",
  "timeInForce": "GTC",
  "type": "MARKET",
  "side": "SELL"
}
```

Response FULL:

```
{
  "symbol": "BTCUSDT",
  "orderId": 28,
  "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
  "transactTime": 1507725176595,
  "price": "0.00000000",
  "origQty": "10.00000000",
  "executedQty": "10.00000000",
  "status": "FILLED",
  "timeInForce": "GTC",
  "type": "MARKET",
  "side": "SELL",
  "fills": [
    {

```

```
        "price": "4000.00000000",
        "qty": "1.00000000",
        "commission": "4.00000000",
        "commissionAsset": "USDT"
    },
    {
        "price": "3999.00000000",
        "qty": "5.00000000",
        "commission": "19.99500000",
        "commissionAsset": "USDT"
    },
    {
        "price": "3998.00000000",
        "qty": "2.00000000",
        "commission": "7.99600000",
        "commissionAsset": "USDT"
    },
    {
        "price": "3997.00000000",
        "qty": "1.00000000",
        "commission": "3.99700000",
        "commissionAsset": "USDT"
    },
    {
        "price": "3995.00000000",
        "qty": "1.00000000",
        "commission": "3.99500000",
        "commissionAsset": "USDT"
    }
]
```

Raises `BinanceResponseException`, `BinanceAPIException`, `BinanceOrderException`, `BinanceOrderMinAmountException`, `BinanceOrderMinPriceException`, `BinanceOrderMinTotalException`, `BinanceOrderUnknownSymbolException`, `BinanceOrderInactiveSymbolException`

create_test_order (**params)

Test new order creation and signature/recvWindow long. Creates and validates a new order but does not send it into the matching engine.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#test-new-order-trade>

Parameters

- **symbol** (*str*) – required
- **side** (*enum*) – required
- **type** (*enum*) – required
- **timeInForce** (*enum*) – required if limit order
- **quantity** (*decimal*) – required
- **price** (*str*) – required
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.

- **icebergQty** (*decimal*) – Used with iceberg orders
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – The number of milliseconds the request is valid for

Returns API response

```
{ }
```

Raises BinanceResponseException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

get_account (***params*)

Get current account information.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#account-information-user_data

Parameters **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "makerCommission": 15,
  "takerCommission": 15,
  "buyerCommission": 0,
  "sellerCommission": 0,
  "canTrade": true,
  "canWithdraw": true,
  "canDeposit": true,
  "balances": [
    {
      "asset": "BTC",
      "free": "4723846.89208129",
      "locked": "0.00000000"
    },
    {
      "asset": "LTC",
      "free": "4763368.68006011",
      "locked": "0.00000000"
    }
  ]
}
```

Raises BinanceResponseException, BinanceAPIException

get_account_status (***params*)

Get account status detail.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/wapi-api.md#account-status-user_data

Parameters **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "msg": "Order failed:Low Order fill rate! Will be reactivated after 5_
↪minutes.",
  "success": true,
  "objs": [
    "5"
  ]
}
```

Raises `BinanceWithdrawException`

get_aggregate_trades (***params*)

Get compressed, aggregate trades. Trades that fill at the time, from the same order, with the same price will have the quantity aggregated.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#compressedaggregate-trades-list>

Parameters

- **symbol** (*str*) – required
- **fromId** (*str*) – ID to get aggregate trades from INCLUSIVE.
- **startTime** (*int*) – Timestamp in ms to get aggregate trades from INCLUSIVE.
- **endTime** (*int*) – Timestamp in ms to get aggregate trades until INCLUSIVE.
- **limit** (*int*) – Default 500; max 500.

Returns API response

```
[
  {
    "a": 26129,          # Aggregate tradeId
    "p": "0.01633102",  # Price
    "q": "4.70443515",  # Quantity
    "f": 27781,         # First tradeId
    "l": 27781,         # Last tradeId
    "T": 1498793709153, # Timestamp
    "m": true,          # Was the buyer the maker?
    "M": true           # Was the trade the best price match?
  }
]
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_all_orders (***params*)

Get all account orders; active, canceled, or filled.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#all-orders-user-data>

Parameters

- **symbol** (*str*) – required
- **orderId** (*int*) – The unique order id
- **limit** (*int*) – Default 500; max 500.

- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
[
  {
    "symbol": "LTCBTC",
    "orderId": 1,
    "clientOrderId": "myOrder1",
    "price": "0.1",
    "origQty": "1.0",
    "executedQty": "0.0",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "stopPrice": "0.0",
    "icebergQty": "0.0",
    "time": 1499827319559
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_all_tickers ()

Latest price for all symbols.

<https://www.binance.com/restapipub.html#symbols-price-ticker>

Returns List of market tickers

```
[
  {
    "symbol": "LTCBTC",
    "price": "4.00000200"
  },
  {
    "symbol": "ETHBTC",
    "price": "0.07946600"
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_asset_balance (*asset*, ***params*)

Get current asset balance.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#account-information-user_data

Parameters

- **asset** (*str*) – required
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns dictionary or None if not found

```
{
  "asset": "BTC",
  "free": "4723846.89208129",
  "locked": "0.00000000"
}
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_deposit_address (**params)

Fetch a deposit address for a symbol

<https://www.binance.com/restapipub.html>

Parameters

- **asset** (*str*) – required
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "address": "0x6915f16f8791d0a1cc2bf47c13a6b2a92000504b",
  "success": true,
  "addressTag": "1231212",
  "asset": "BNB"
}
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_deposit_history (**params)

Fetch deposit history.

<https://www.binance.com/restapipub.html>

Parameters

- **asset** (*str*) – optional
- **startTime** (*long*) – optional
- **endTime** (*long*) – optional
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "depositList": [
    {
      "insertTime": 1508198532000,
      "amount": 0.04670582,
      "asset": "ETH",
      "status": 1
    }
  ],
  "success": true
}
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_exchange_info()

Return rate limits and list of symbols

Returns list - List of product dictionaries

```
{
  "timezone": "UTC",
  "serverTime": 1508631584636,
  "rateLimits": [
    {
      "rateLimitType": "REQUESTS",
      "interval": "MINUTE",
      "limit": 1200
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "SECOND",
      "limit": 10
    },
    {
      "rateLimitType": "ORDERS",
      "interval": "DAY",
      "limit": 100000
    }
  ],
  "exchangeFilters": [],
  "symbols": [
    {
      "symbol": "ETHBTC",
      "status": "TRADING",
      "baseAsset": "ETH",
      "baseAssetPrecision": 8,
      "quoteAsset": "BTC",
      "quotePrecision": 8,
      "orderTypes": ["LIMIT", "MARKET"],
      "icebergAllowed": false,
      "filters": [
        {
          "filterType": "PRICE_FILTER",
          "minPrice": "0.00000100",
          "maxPrice": "100000.00000000",
          "tickSize": "0.00000100"
        }, {
          "filterType": "LOT_SIZE",
          "minQty": "0.00100000",
          "maxQty": "100000.00000000",
          "stepSize": "0.00100000"
        }, {
          "filterType": "MIN_NOTIONAL",
          "minNotional": "0.00100000"
        }
      ]
    }
  ]
}
```

Raises BinanceResponseException, BinanceAPIException

get_historical_klines (*symbol*, *interval*, *start_str*, *end_str=None*)

Get Historical Klines from Binance

See dateparse docs for valid start and end string formats <http://dateparser.readthedocs.io/en/latest/>

If using offset strings for dates add “UTC” to date string e.g. “now UTC”, “11 hours ago UTC”

Parameters

- **symbol** (*str*) – Name of symbol pair e.g BNBBTC
- **interval** (*str*) – Biannce Kline interval
- **start_str** (*str*) – Start date string in UTC format
- **end_str** (*str*) – optional - end date string in UTC format

Returns list of OHLCV values

get_historical_trades (***params*)

Get older trades.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#recent-trades-list>

Parameters

- **symbol** (*str*) – required
- **limit** (*int*) – Default 500; max 500.
- **fromId** (*str*) – TradeId to fetch from. Default gets most recent trades.

Returns API response

```
[
  {
    "id": 28457,
    "price": "4.00000100",
    "qty": "12.00000000",
    "time": 1499865549590,
    "isBuyerMaker": true,
    "isBestMatch": true
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_klines (***params*)

Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#klinecandlestick-data>

Parameters

- **symbol** (*str*) – required
- **interval** (*enum*) –
–
- **limit** (*int*) –
– Default 500; max 500.

- **startTime** (*int*) –
- **endTime** (*int*) –

Returns API response

```
[
  [
    1499040000000,      # Open time
    "0.01634790",       # Open
    "0.80000000",       # High
    "0.01575800",       # Low
    "0.01577100",       # Close
    "148976.11427815",  # Volume
    1499644799999,      # Close time
    "2434.19055334",    # Quote asset volume
    308,                # Number of trades
    "1756.87402397",    # Taker buy base asset volume
    "28.46694368",      # Taker buy quote asset volume
    "17928899.62484339" # Can be ignored
  ]
]
```

Raises BinanceResponseException, BinanceAPIException

get_my_trades (**params*)

Get trades for a specific symbol.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#account-trade-list-user_data

Parameters

- **symbol** (*str*) – required
- **limit** (*int*) – Default 500; max 500.
- **fromId** (*int*) – TradeId to fetch from. Default gets most recent trades.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
[
  {
    "id": 28457,
    "price": "4.00000100",
    "qty": "12.00000000",
    "commission": "10.10000000",
    "commissionAsset": "BNB",
    "time": 1499865549590,
    "isBuyer": true,
    "isMaker": false,
    "isBestMatch": true
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_open_orders (**params)

Get all open orders on a symbol.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#current-open-orders-user_data

Parameters

- **symbol** (*str*) – optional
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
[
  {
    "symbol": "LTCBTC",
    "orderId": 1,
    "clientOrderId": "myOrder1",
    "price": "0.1",
    "origQty": "1.0",
    "executedQty": "0.0",
    "status": "NEW",
    "timeInForce": "GTC",
    "type": "LIMIT",
    "side": "BUY",
    "stopPrice": "0.0",
    "icebergQty": "0.0",
    "time": 1499827319559
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_order (**params)

Check an order's status. Either orderId or origClientId must be sent.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#query-order-user_data

Parameters

- **symbol** (*str*) – required
- **orderId** (*int*) – The unique order id
- **origClientId** (*str*) – optional
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "symbol": "LTCBTC",
  "orderId": 1,
  "clientOrderId": "myOrder1",
  "price": "0.1",
  "origQty": "1.0",
  "executedQty": "0.0",
  "status": "NEW",
  "timeInForce": "GTC",
  "type": "LIMIT",
```

```

    "side": "BUY",
    "stopPrice": "0.0",
    "icebergQty": "0.0",
    "time": 1499827319559
}

```

Raises BinanceResponseException, BinanceAPIException

get_order_book (**params)

Get the Order Book for the market

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#order-book>

Parameters

- **symbol** (*str*) – required
- **limit** (*int*) – Default 100; max 100

Returns API response

```

{
    "lastUpdateId": 1027024,
    "bids": [
        [
            "4.00000000",      # PRICE
            "431.00000000",   # QTY
            []                 # Can be ignored
        ]
    ],
    "asks": [
        [
            "4.00000200",
            "12.00000000",
            []
        ]
    ]
}

```

Raises BinanceResponseException, BinanceAPIException

get_orderbook_ticker (**params)

Latest price for a symbol or symbols.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#symbol-order-book-ticker>

Parameters **symbol** (*str*) –

Returns API response

```

{
    "symbol": "LTCBTC",
    "bidPrice": "4.00000000",
    "bidQty": "431.00000000",
    "askPrice": "4.00000200",
    "askQty": "9.00000000"
}

```

OR

```
[
  {
    "symbol": "LTCBTC",
    "bidPrice": "4.00000000",
    "bidQty": "431.00000000",
    "askPrice": "4.00000200",
    "askQty": "9.00000000"
  },
  {
    "symbol": "ETHBTC",
    "bidPrice": "0.07946700",
    "bidQty": "9.00000000",
    "askPrice": "0.00000000",
    "askQty": "1000.00000000"
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_orderbook_tickers()

Best price/qty on the order book for all symbols.

<https://www.binance.com/restapipub.html#symbols-order-book-ticker>

Returns List of order book market entries

```
[
  {
    "symbol": "LTCBTC",
    "bidPrice": "4.00000000",
    "bidQty": "431.00000000",
    "askPrice": "4.00000200",
    "askQty": "9.00000000"
  },
  {
    "symbol": "ETHBTC",
    "bidPrice": "0.07946700",
    "bidQty": "9.00000000",
    "askPrice": "0.00000000",
    "askQty": "1000.00000000"
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_products()

Return list of products currently listed on Binance

Use get_exchange_info() call instead

Returns list - List of product dictionaries

Raises BinanceResponseException, BinanceAPIException

get_recent_trades(params)**

Get recent trades (up to last 500).

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#recent-trades-list>

Parameters

- **symbol** (*str*) – required
- **limit** (*int*) – Default 500; max 500.

Returns API response

```
[
  {
    "id": 28457,
    "price": "4.00000100",
    "qty": "12.00000000",
    "time": 1499865549590,
    "isBuyerMaker": true,
    "isBestMatch": true
  }
]
```

Raises BinanceResponseException, BinanceAPIException

get_server_time()

Test connectivity to the Rest API and get the current server time.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#check-server-time>

Returns Current server time

```
{
  "serverTime": 1499827319559
}
```

Raises BinanceResponseException, BinanceAPIException

get_symbol_info(symbol)

Return information about a symbol

Parameters **symbol** (*str*) – required e.g BNBBTC

Returns Dict if found, None if not

```
{
  "symbol": "ETHBTC",
  "status": "TRADING",
  "baseAsset": "ETH",
  "baseAssetPrecision": 8,
  "quoteAsset": "BTC",
  "quotePrecision": 8,
  "orderTypes": ["LIMIT", "MARKET"],
  "icebergAllowed": false,
  "filters": [
    {
      "filterType": "PRICE_FILTER",
      "minPrice": "0.00000100",
      "maxPrice": "100000.00000000",
      "tickSize": "0.00000100"
    }
  ]
}
```

```
    }, {
        "filterType": "LOT_SIZE",
        "minQty": "0.00100000",
        "maxQty": "100000.00000000",
        "stepSize": "0.00100000"
    }, {
        "filterType": "MIN_NOTIONAL",
        "minNotional": "0.00100000"
    }
]
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_symbol_ticker (**params)

Latest price for a symbol or symbols.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#24hr-ticker-price-change-statistics>

Parameters `symbol` (*str*) –

Returns API response

```
{
    "symbol": "LTCBTC",
    "price": "4.00000200"
}
```

OR

```
[
    {
        "symbol": "LTCBTC",
        "price": "4.00000200"
    },
    {
        "symbol": "ETHBTC",
        "price": "0.07946600"
    }
]
```

Raises `BinanceResponseException`, `BinanceAPIException`

get_ticker (**params)

24 hour price change statistics.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#24hr-ticker-price-change-statistics>

Parameters `symbol` (*str*) –

Returns API response

```
{
    "priceChange": "-94.99999800",
    "priceChangePercent": "-95.960",
    "weightedAvgPrice": "0.29628482",
```

```

    "prevClosePrice": "0.10002000",
    "lastPrice": "4.00000200",
    "bidPrice": "4.00000000",
    "askPrice": "4.00000200",
    "openPrice": "99.00000000",
    "highPrice": "100.00000000",
    "lowPrice": "0.10000000",
    "volume": "8913.30000000",
    "openTime": 1499783499040,
    "closeTime": 1499869899040,
    "fristId": 28385,      # First tradeId
    "lastId": 28460,      # Last tradeId
    "count": 76           # Trade count
}

```

OR

```

[
  {
    "priceChange": "-94.99999800",
    "priceChangePercent": "-95.960",
    "weightedAvgPrice": "0.29628482",
    "prevClosePrice": "0.10002000",
    "lastPrice": "4.00000200",
    "bidPrice": "4.00000000",
    "askPrice": "4.00000200",
    "openPrice": "99.00000000",
    "highPrice": "100.00000000",
    "lowPrice": "0.10000000",
    "volume": "8913.30000000",
    "openTime": 1499783499040,
    "closeTime": 1499869899040,
    "fristId": 28385,      # First tradeId
    "lastId": 28460,      # Last tradeId
    "count": 76           # Trade count
  }
]

```

Raises `BinanceResponseException`, `BinanceAPIException`

get_withdraw_history (***params*)

Fetch withdraw history.

<https://www.binance.com/restapipub.html>

Parameters

- **asset** (*str*) – optional
- **startTime** (*long*) – optional
- **endTime** (*long*) – optional
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```

{
  "withdrawList": [
    {

```

```
        "amount": 1,
        "address": "0x6915f16f8791d0a1cc2bf47c13a6b2a92000504b",
        "asset": "ETH",
        "applyTime": 1508198532000
        "status": 4
    },
    {
        "amount": 0.005,
        "address": "0x6915f16f8791d0a1cc2bf47c13a6b2a92000504b",
        "txId":
↪ "0x80aaabed54bdab3f6de5868f89929a2371ad21d666f20f7393d1a3389fad95a1",
        "asset": "ETH",
        "applyTime": 1508198532000,
        "status": 4
    }
],
"success": true
}
```

Raises BinanceResponseException, BinanceAPIException

order_limit (*timeInForce*='GTC', ***params*)

Send in a new limit order

Any order with an icebergQty MUST have timeInForce set to GTC.

Parameters

- **symbol** (*str*) – required
- **side** (*enum*) – required
- **quantity** (*decimal*) – required
- **price** (*str*) – required
- **timeInForce** (*enum*) – default Good till cancelled
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **icebergQty** (*decimal*) – Used with LIMIT, STOP_LOSS_LIMIT, and TAKE_PROFIT_LIMIT to create an iceberg order.
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises BinanceResponseException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

order_limit_buy (*timeInForce*='GTC', ***params*)

Send in a new limit buy order

Any order with an icebergQty MUST have timeInForce set to GTC.

Parameters

- **symbol** (*str*) – required
- **quantity** (*decimal*) – required
- **price** (*str*) – required
- **timeInForce** (*enum*) – default Good till cancelled
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **stopPrice** (*decimal*) – Used with stop orders
- **icebergQty** (*decimal*) – Used with iceberg orders
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises BinanceResponseException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

order_limit_sell (*timeInForce*=*'GTC'*, ***params*)

Send in a new limit sell order

Parameters

- **symbol** (*str*) – required
- **quantity** (*decimal*) – required
- **price** (*str*) – required
- **timeInForce** (*enum*) – default Good till cancelled
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **stopPrice** (*decimal*) – Used with stop orders
- **icebergQty** (*decimal*) – Used with iceberg orders
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises BinanceResponseException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

order_market (***params*)

Send in a new market order

Parameters

- **symbol** (*str*) – required

- **side** (*enum*) – required
- **quantity** (*decimal*) – required
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises `BinanceResponseException`, `BinanceAPIException`, `BinanceOrderException`, `BinanceOrderMinAmountException`, `BinanceOrderMinPriceException`, `BinanceOrderMinTotalException`, `BinanceOrderUnknownSymbolException`, `BinanceOrderInactiveSymbolException`

order_market_buy (***params*)

Send in a new market buy order

Parameters

- **symbol** (*str*) – required
- **quantity** (*decimal*) – required
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises `BinanceResponseException`, `BinanceAPIException`, `BinanceOrderException`, `BinanceOrderMinAmountException`, `BinanceOrderMinPriceException`, `BinanceOrderMinTotalException`, `BinanceOrderUnknownSymbolException`, `BinanceOrderInactiveSymbolException`

order_market_sell (***params*)

Send in a new market sell order

Parameters

- **symbol** (*str*) – required
- **quantity** (*decimal*) – required
- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
- **newOrderRespType** (*enum*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

See order endpoint for full response options

Raises `BinanceResponseException`, `BinanceAPIException`, `BinanceOrderException`, `BinanceOrderMinAmountException`, `BinanceOrderMinPriceException`, `BinanceOrderMinTotalException`, `BinanceOrderUnknownSymbolException`, `BinanceOrderInactiveSymbolException`

ping()

Test connectivity to the Rest API.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#test-connectivity>

Returns Empty array

```
{ }
```

Raises `BinanceResponseException`, `BinanceAPIException`

stream_close(listenKey)

Close out a user data stream.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#close-user-data-stream-user_stream

Parameters `listenKey` (*str*) – required

Returns API response

```
{ }
```

Raises `BinanceResponseException`, `BinanceAPIException`

stream_get_listen_key()

Start a new user data stream and return the listen key If a stream already exists it should return the same key. If the stream becomes invalid a new key is returned.

Can be used to keep the user stream alive.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#start-user-data-stream-user_stream

Returns API response

```
{
  "listenKey":
  → "pqia91ma19a5s61cv6a81va65sdf19v8a65a1a5s61cv6a81va65sdf19v8a65a1"
}
```

Raises `BinanceResponseException`, `BinanceAPIException`

stream_keepalive(listenKey)

PING a user data stream to prevent a time out.

https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md#keepalive-user-data-stream-user_stream

Parameters `listenKey` (*str*) – required

Returns API response

```
{ }
```

Raises `BinanceResponseException`, `BinanceAPIException`

withdraw (***params*)

Submit a withdraw request.

<https://www.binance.com/restapipub.html>

Assumptions:

- You must have Withdraw permissions enabled on your API key
- You must have withdrawn to the address specified through the website and approved the transaction via email

Parameters

- **asset** (*str*) – required
- **amount** (*decimal*) – required
- **name** (*str*) – optional - Description of the address, default asset value passed will be used
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

Returns API response

```
{
  "msg": "success",
  "success": true,
  "id": "7213fea8e94b4a5593d507237e5a555b"
}
```

Raises `BinanceResponseException`, `BinanceAPIException`, `BinanceWithdrawException`

depthcache module

class `binance.depthcache.DepthCache` (*symbol*)

Bases: `object`

__init__ (*symbol*)

Intialise the DepthCache

Parameters **symbol** (*string*) – Symbol to create depth cache for

add_ask (*ask*)

Add an ask to the cache

Parameters **ask** –

Returns

add_bid (*bid*)

Add a bid to the cache

Parameters **bid** –

Returns

get_asks ()

Get the current asks

Returns list of asks with price and quantity as floats

```
[
  [
    0.0001955, # Price
    57.0'      # Quantity
  ],
  [
    0.00019699,
    778.0
  ],
  [
    0.000197,
    64.0
  ],
  [
    0.00019709,
    1130.0
  ],
  [
    0.0001971,
    385.0
  ]
]
```

get_bids()

Get the current bids

Returns list of bids with price and quantity as floats

```
[
  [
    0.0001946, # Price
    45.0       # Quantity
  ],
  [
    0.00019459,
    2384.0
  ],
  [
    0.00019158,
    5219.0
  ],
  [
    0.00019157,
    1180.0
  ],
  [
    0.00019082,
    287.0
  ]
]
```

static sort_depth (vals, reverse=False)

Sort bids or asks by price

class binance.depthcache.**DepthCacheManager** (*client, symbol, callback=None, re-
fresh_interval=1800*)

Bases: object

`__init__ (client, symbol, callback=None, refresh_interval=1800)`

Initialise the DepthCacheManager

Parameters

- **client** (*binance.Client*) – Binance API client
- **symbol** (*string*) – Symbol to create depth cache for
- **callback** (*function*) – Optional function to receive depth cache updates
- **refresh_interval** (*int*) – Optional number of seconds between cache refresh, use 0 or None to disable

`close ()`

Close the open socket for this manager

Returns

`get_depth_cache ()`

Get the current depth cache

Returns DepthCache object

exceptions module

exception `binance.exceptions.BinanceAPIException (response)`

Bases: `exceptions.Exception`

LISTENKEY_NOT_EXIST = '-1125'

`__init__ (response)`

exception `binance.exceptions.BinanceOrderException (code, message)`

Bases: `exceptions.Exception`

`__init__ (code, message)`

exception `binance.exceptions.BinanceOrderInactiveSymbolException (value)`

Bases: `binance.exceptions.BinanceOrderException`

`__init__ (value)`

exception `binance.exceptions.BinanceOrderMinAmountException (value)`

Bases: `binance.exceptions.BinanceOrderException`

`__init__ (value)`

exception `binance.exceptions.BinanceOrderMinPriceException (value)`

Bases: `binance.exceptions.BinanceOrderException`

`__init__ (value)`

exception `binance.exceptions.BinanceOrderMinTotalException (value)`

Bases: `binance.exceptions.BinanceOrderException`

`__init__ (value)`

exception `binance.exceptions.BinanceOrderUnknownSymbolException (value)`

Bases: `binance.exceptions.BinanceOrderException`

`__init__ (value)`

exception `binance.exceptions.BinanceRequestException (message)`

Bases: `exceptions.Exception`

```
__init__(message)
```

```
exception binance.exceptions.BinanceWithdrawException(message)
```

```
Bases: exceptions.Exception
```

```
__init__(message)
```

helpers module

```
binance.helpers.date_to_milliseconds(date_str)
```

Convert UTC date to milliseconds

If using offset strings add “UTC” to date string e.g. “now UTC”, “11 hours ago UTC”

See dateparse docs for formats <http://dateparser.readthedocs.io/en/latest/>

Parameters `date_str` (*str*) – date in readable format, i.e. “January 01, 2018”, “11 hours ago UTC”, “now UTC”

```
binance.helpers.interval_to_milliseconds(interval)
```

Convert a Binance interval string to milliseconds

Parameters `interval` (*str*) – Binance interval string 1m, 3m, 5m, 15m, 30m, 1h, 2h, 4h, 6h, 8h, 12h, 1d, 3d, 1w

Returns None if unit not one of m, h, d or w None if string not in correct format int value of interval in milliseconds

websockets module

```
class binance.websockets.BinanceClientFactory(*args, **kwargs)
```

Bases: autobahn.twisted.websocket.WebSocketClientFactory, *binance.websockets.BinanceReconnectingClientFactory*

```
clientConnectionFailed(connector, reason)
```

```
clientConnectionLost(connector, reason)
```

```
protocol
```

alias of *BinanceClientProtocol*

```
class binance.websockets.BinanceClientProtocol
```

Bases: autobahn.twisted.websocket.WebSocketClientProtocol

```
onConnect(response)
```

```
onMessage(payload, isBinary)
```

```
class binance.websockets.BinanceReconnectingClientFactory
```

Bases: twisted.internet.protocol.ReconnectingClientFactory

```
initialDelay = 0.1
```

```
maxDelay = 10
```

```
maxRetries = 5
```

```
class binance.websockets.BinanceSocketManager(client)
```

Bases: threading.Thread

```
STREAM_URL = 'wss://stream.binance.com:9443/'
```

```
WEBSOCKET_DEPTH_10 = '10'
```

```
WEBSOCKET_DEPTH_20 = '20'
```

```
WEBSOCKET_DEPTH_5 = '5'
```

```
__init__(client)
```

Initialise the BinanceSocketManager

Parameters `client` (*binance.Client*) – Binance API client

```
close()
```

Close all connections

```
run()
```

```
start_aggtrade_socket(symbol, callback)
```

Start a websocket for symbol trade data

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#aggregate-trade-streams>

Parameters

- **symbol** (*str*) – required
- **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format

```
{
    "e": "aggTrade",           # event type
    "E": 1499405254326,       # event time
    "s": "ETHBTC",            # symbol
    "a": 70232,                # aggregated tradeid
    "p": "0.10281118",        # price
    "q": "8.15632997",        # quantity
    "f": 77489,                # first breakdown trade id
    "l": 77489,                # last breakdown trade id
    "T": 1499405254324,       # trade time
    "m": false,                # whether buyer is a maker
    "M": true                  # can be ignored
}
```

```
start_depth_socket(symbol, callback, depth=None)
```

Start a websocket for symbol market depth returning either a diff or a partial book

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#partial-book-depth-streams>

Parameters

- **symbol** (*str*) – required
- **callback** (*function*) – callback function to handle messages
- **depth** (*enum*) – optional Number of depth entries to return, default None. If passed returns a partial book instead of a diff

Returns connection key string if successful, False otherwise

Partial Message Format

```
{
  "lastUpdateId": 160,  # Last update ID
  "bids": [             # Bids to be updated
    [
      "0.0024",         # price level to be updated
      "10",             # quantity
      []                # ignore
    ]
  ],
  "asks": [             # Asks to be updated
    [
      "0.0026",         # price level to be updated
      "100",            # quantity
      []                # ignore
    ]
  ]
}
```

Diff Message Format

```
{
  "e": "depthUpdate", # Event type
  "E": 123456789,     # Event time
  "s": "BNBBTC",      # Symbol
  "U": 157,           # First update ID in event
  "u": 160,           # Final update ID in event
  "b": [              # Bids to be updated
    [
      "0.0024",         # price level to be updated
      "10",             # quantity
      []                # ignore
    ]
  ],
  "a": [              # Asks to be updated
    [
      "0.0026",         # price level to be updated
      "100",            # quantity
      []                # ignore
    ]
  ]
}
```

start_kline_socket (*symbol*, *callback*, *interval*='1m')

Start a websocket for symbol kline data

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#klinecandlestick-streams>

Parameters

- **symbol** (*str*) – required
- **callback** (*function*) – callback function to handle messages
- **interval** (*enum*) – Kline interval, default KLINE_INTERVAL_1MINUTE

Returns connection key string if successful, False otherwise

Message Format

```

{
    "e": "kline",                # event type
    "E": 1499404907056,          # event time
    "s": "ETHBTC",               # symbol
    "k": {
        "t": 1499404860000,      # start time of this bar
        "T": 1499404919999,      # end time of this bar
        "s": "ETHBTC",           # symbol
        "i": "1m",               # interval
        "f": 77462,              # first trade id
        "L": 77465,              # last trade id
        "o": "0.10278577",        # open
        "c": "0.10278645",        # close
        "h": "0.10278712",        # high
        "l": "0.10278518",        # low
        "v": "17.47929838",       # volume
        "n": 4,                  # number of trades
        "x": false,              # whether this bar is
    }                             # final
    "q": "1.79662878",           # quote volume
    "V": "2.34879839",           # volume of active buy
    "Q": "0.24142166",           # quote volume of active buy
    "B": "13279784.01349473"    # can be ignored
}

```

start_multiplex_socket (*streams*, *callback*)

Start a multiplexed socket using a list of socket names. User stream sockets can not be included.

Symbols in socket name must be lowercase i.e `bnbbtc@aggTrade`, `neobtc@ticker`

Combined stream events are wrapped as follows: `{"stream": "<streamName>", "data": <rawPayload>}`

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md>

Parameters

- **streams** (*list*) – list of stream names in lower case
- **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

start_symbol_ticker_socket (*symbol*, *callback*)

Start a websocket for a symbol's ticker data

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#individual-symbol-ticker-streams>

Parameters

- **symbol** (*str*) – required
- **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format

```

{
    "e": "24hrTicker",          # Event type

```

```

    "E": 123456789,      # Event time
    "s": "BNBBTC",      # Symbol
    "p": "0.0015",      # Price change
    "P": "250.00",      # Price change percent
    "w": "0.0018",      # Weighted average price
    "x": "0.0009",      # Previous day's close price
    "c": "0.0025",      # Current day's close price
    "Q": "10",          # Close trade's quantity
    "b": "0.0024",      # Best bid price
    "B": "10",          # Bid bid quantity
    "a": "0.0026",      # Best ask price
    "A": "100",         # Best ask quantity
    "o": "0.0010",      # Open price
    "h": "0.0025",      # High price
    "l": "0.0010",      # Low price
    "v": "10000",       # Total traded base asset volume
    "q": "18",          # Total traded quote asset volume
    "O": 0,             # Statistics open time
    "C": 86400000,      # Statistics close time
    "F": 0,             # First trade ID
    "L": 18150,         # Last trade Id
    "n": 18151          # Total number of trades
}

```

start_ticker_socket (*callback*)

Start a websocket for all ticker data

By default all markets are included in an array.

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#all-market-tickers-stream>

Parameters **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format

```

[
  {
    'F': 278610,
    'o': '0.07393000',
    's': 'BCCBTC',
    'C': 1509622420916,
    'b': '0.07800800',
    'l': '0.07160300',
    'h': '0.08199900',
    'L': 287722,
    'P': '6.694',
    'Q': '0.10000000',
    'q': '1202.67106335',
    'p': '0.00494900',
    'O': 1509536020916,
    'a': '0.07887800',
    'n': 9113,
    'B': '1.00000000',
    'c': '0.07887900',
    'x': '0.07399600',
    'w': '0.07639068',
    'A': '2.41900000',
  }
]

```

```
        'v': '15743.68900000'
    }
]
```

start_trade_socket (*symbol*, *callback*)

Start a websocket for symbol trade data

<https://github.com/binance-exchange/binance-official-api-docs/blob/master/web-socket-streams.md#trade-streams>

Parameters

- **symbol** (*str*) – required
- **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format

```
{
    "e": "trade",      # Event type
    "E": 123456789,    # Event time
    "s": "BNBBTC",     # Symbol
    "t": 12345,        # Trade ID
    "p": "0.001",      # Price
    "q": "100",        # Quantity
    "b": 88,           # Buyer order Id
    "a": 50,           # Seller order Id
    "T": 123456785,    # Trade time
    "m": true,         # Is the buyer the market maker?
    "M": true          # Ignore.
}
```

start_user_socket (*callback*)

Start a websocket for user data

<https://www.binance.com/restapipub.html#user-wss-endpoint>

Parameters **callback** (*function*) – callback function to handle messages

Returns connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

stop_socket (*conn_key*)

Stop a websocket given the connection key

Parameters **conn_key** (*string*) – Socket connection key

Returns connection key string if successful, False otherwise

4.2 Index

- [genindex](#)

b

- `binance.client`, [30](#)
- `binance.depthcache`, [52](#)
- `binance.exceptions`, [54](#)
- `binance.helpers`, [55](#)
- `binance.websockets`, [55](#)

Symbols

[__init__\(\)](#) (binance.client.Client method), 32
[__init__\(\)](#) (binance.depthcache.DepthCache method), 52
[__init__\(\)](#) (binance.depthcache.DepthCacheManager method), 53
[__init__\(\)](#) (binance.exceptions.BinanceAPIException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderInactiveSymbolException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderMinAmountException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderMinPriceException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderMinTotalException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceOrderUnknownSymbolException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceRequestException method), 54
[__init__\(\)](#) (binance.exceptions.BinanceWithdrawException method), 55
[__init__\(\)](#) (binance.websockets.BinanceSocketManager method), 56
[BinanceClientProtocol](#) (class in binance.websockets), 55
[BinanceOrderException](#), 54
[BinanceOrderInactiveSymbolException](#), 54
[BinanceOrderMinAmountException](#), 54
[BinanceOrderMinPriceException](#), 54
[BinanceOrderMinTotalException](#), 54
[BinanceOrderUnknownSymbolException](#), 54
[BinanceReconnectingClientFactory](#) (class in binance.websockets), 55
[BinanceRequestException](#), 54
[BinanceSocketManager](#) (class in binance.websockets), 55
[BinanceWithdrawException](#), 55

C

[cancel_order\(\)](#) (binance.client.Client method), 32
[Client](#) (class in binance.client), 30
[clientConnectionFailed\(\)](#) (binance.websockets.BinanceClientFactory method), 55
[clientConnectionLost\(\)](#) (binance.websockets.BinanceClientFactory method), 55
[close\(\)](#) (binance.depthcache.DepthCacheManager method), 54
[close\(\)](#) (binance.websockets.BinanceSocketManager method), 56
[create_order\(\)](#) (binance.client.Client method), 32
[create_test_order\(\)](#) (binance.client.Client method), 34

D

[date_to_milliseconds\(\)](#) (in module binance.helpers), 55
[DepthCache](#) (class in binance.depthcache), 52
[DepthCacheManager](#) (class in binance.depthcache), 53

G

[get_account\(\)](#) (binance.client.Client method), 35
[get_account_status\(\)](#) (binance.client.Client method), 35
[get_aggregate_trades\(\)](#) (binance.client.Client method), 36
[get_all_orders\(\)](#) (binance.client.Client method), 36

A

[add_ask\(\)](#) (binance.depthcache.DepthCache method), 52
[add_bid\(\)](#) (binance.depthcache.DepthCache method), 52
[API_URL](#) (binance.client.Client attribute), 30

B

[binance.client](#) (module), 30
[binance.depthcache](#) (module), 52
[binance.exceptions](#) (module), 54
[binance.helpers](#) (module), 55
[binance.websockets](#) (module), 55
[BinanceAPIException](#), 54
[BinanceClientFactory](#) (class in binance.websockets), 55

[get_all_tickers\(\)](#) (binance.client.Client method), [37](#)
[get_asks\(\)](#) (binance.depthcache.DepthCache method), [52](#)
[get_asset_balance\(\)](#) (binance.client.Client method), [37](#)
[get_bids\(\)](#) (binance.depthcache.DepthCache method), [53](#)
[get_deposit_address\(\)](#) (binance.client.Client method), [38](#)
[get_deposit_history\(\)](#) (binance.client.Client method), [38](#)
[get_depth_cache\(\)](#) (binance.depthcache.DepthCacheManager method), [54](#)
[get_exchange_info\(\)](#) (binance.client.Client method), [39](#)
[get_historical_klines\(\)](#) (binance.client.Client method), [39](#)
[get_historical_trades\(\)](#) (binance.client.Client method), [40](#)
[get_klines\(\)](#) (binance.client.Client method), [40](#)
[get_my_trades\(\)](#) (binance.client.Client method), [41](#)
[get_open_orders\(\)](#) (binance.client.Client method), [41](#)
[get_order\(\)](#) (binance.client.Client method), [42](#)
[get_order_book\(\)](#) (binance.client.Client method), [43](#)
[get_orderbook_ticker\(\)](#) (binance.client.Client method), [43](#)
[get_orderbook_tickers\(\)](#) (binance.client.Client method), [44](#)
[get_products\(\)](#) (binance.client.Client method), [44](#)
[get_recent_trades\(\)](#) (binance.client.Client method), [44](#)
[get_server_time\(\)](#) (binance.client.Client method), [45](#)
[get_symbol_info\(\)](#) (binance.client.Client method), [45](#)
[get_symbol_ticker\(\)](#) (binance.client.Client method), [46](#)
[get_ticker\(\)](#) (binance.client.Client method), [46](#)
[get_withdraw_history\(\)](#) (binance.client.Client method), [47](#)

I

[initialDelay](#) (binance.websockets.BinanceReconnectingClientFactory attribute), [55](#)
[interval_to_milliseconds\(\)](#) (in module binance.helpers), [55](#)

K

[KLINE_INTERVAL_12HOUR](#) (binance.client.Client attribute), [30](#)
[KLINE_INTERVAL_15MINUTE](#) (binance.client.Client attribute), [30](#)
[KLINE_INTERVAL_1DAY](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_1HOUR](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_1MINUTE](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_1MONTH](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_1WEEK](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_2HOUR](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_30MINUTE](#) (binance.client.Client attribute), [31](#)

[KLINE_INTERVAL_3DAY](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_3MINUTE](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_4HOUR](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_5MINUTE](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_6HOUR](#) (binance.client.Client attribute), [31](#)
[KLINE_INTERVAL_8HOUR](#) (binance.client.Client attribute), [31](#)

L

[LISTENKEY_NOT_EXIST](#) (binance.exceptions.BinanceAPIException attribute), [54](#)

M

[maxDelay](#) (binance.websockets.BinanceReconnectingClientFactory attribute), [55](#)
[maxRetries](#) (binance.websockets.BinanceReconnectingClientFactory attribute), [55](#)

O

[onConnect\(\)](#) (binance.websockets.BinanceClientProtocol method), [55](#)
[onMessage\(\)](#) (binance.websockets.BinanceClientProtocol method), [55](#)
[order_limit\(\)](#) (binance.client.Client method), [48](#)
[order_limit_buy\(\)](#) (binance.client.Client method), [48](#)
[order_limit_sell\(\)](#) (binance.client.Client method), [49](#)
[order_market\(\)](#) (binance.client.Client method), [49](#)
[order_market_buy\(\)](#) (binance.client.Client method), [50](#)
[order_market_sell\(\)](#) (binance.client.Client method), [50](#)
[ORDER_RESP_TYPE_ACK](#) (binance.client.Client attribute), [31](#)
[ORDER_RESP_TYPE_FULL](#) (binance.client.Client attribute), [31](#)
[ORDER_RESP_TYPE_RESULT](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_CANCELED](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_EXPIRED](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_FILLED](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_NEW](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_PARTIALLY_FILLED](#) (binance.client.Client attribute), [31](#)
[ORDER_STATUS_PENDING_CANCEL](#) (binance.client.Client attribute), [31](#)

ORDER_STATUS_REJECTED (binance.client.Client attribute), 31

ORDER_TYPE_LIMIT (binance.client.Client attribute), 31

ORDER_TYPE_LIMIT_MAKER (binance.client.Client attribute), 31

ORDER_TYPE_MARKET (binance.client.Client attribute), 31

ORDER_TYPE_STOP_LOSS (binance.client.Client attribute), 31

ORDER_TYPE_STOP_LOSS_LIMIT (binance.client.Client attribute), 31

ORDER_TYPE_TAKE_PROFIT (binance.client.Client attribute), 31

ORDER_TYPE_TAKE_PROFIT_LIMIT (binance.client.Client attribute), 31

P

ping() (binance.client.Client method), 51

PRIVATE_API_VERSION (binance.client.Client attribute), 31

protocol (binance.websockets.BinanceClientFactory attribute), 55

PUBLIC_API_VERSION (binance.client.Client attribute), 31

R

run() (binance.websockets.BinanceSocketManager method), 56

S

SIDE_BUY (binance.client.Client attribute), 31

SIDE_SELL (binance.client.Client attribute), 31

sort_depth() (binance.depthcache.DepthCache static method), 53

start_aggttrade_socket() (binance.websockets.BinanceSocketManager method), 56

start_depth_socket() (binance.websockets.BinanceSocketManager method), 56

start_kline_socket() (binance.websockets.BinanceSocketManager method), 57

start_multiplex_socket() (binance.websockets.BinanceSocketManager method), 58

start_symbol_ticker_socket() (binance.websockets.BinanceSocketManager method), 58

start_ticker_socket() (binance.websockets.BinanceSocketManager method), 59

start_trade_socket() (binance.websockets.BinanceSocketManager method), 60

start_user_socket() (binance.websockets.BinanceSocketManager method), 60

stop_socket() (binance.websockets.BinanceSocketManager method), 60

stream_close() (binance.client.Client method), 51

stream_get_listen_key() (binance.client.Client method), 51

stream_keepalive() (binance.client.Client method), 51

STREAM_URL (binance.websockets.BinanceSocketManager attribute), 55

SYMBOL_TYPE_SPOT (binance.client.Client attribute), 31

T

TIME_IN_FORCE_FOK (binance.client.Client attribute), 31

TIME_IN_FORCE_GTC (binance.client.Client attribute), 32

TIME_IN_FORCE_IOC (binance.client.Client attribute), 32

W

WEBSITE_URL (binance.client.Client attribute), 32

WEBSOCKET_DEPTH_10 (binance.websockets.BinanceSocketManager attribute), 55

WEBSOCKET_DEPTH_20 (binance.websockets.BinanceSocketManager attribute), 55

WEBSOCKET_DEPTH_5 (binance.websockets.BinanceSocketManager attribute), 56

withdraw() (binance.client.Client method), 52

WITHDRAW_API_URL (binance.client.Client attribute), 32

WITHDRAW_API_VERSION (binance.client.Client attribute), 32