# python-binance Documentation

*Release 0.2.0*

**Sam McHardy**

**Sep 30, 2021**

# Contents

Updated 27th Sept 2021

This is an unofficial Python wrapper for the Binance exchange REST API v3. 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.

If you're interested in Binance's new DEX Binance Chain see my python-binance-chain library

**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 including async**  https://sammchardy.github.io

- Async basics for Binance
- Understanding Binance Order Filters

Make sure you update often and check the Changelog for new features and bug fixes.

# Features

- Implementation of all General, Market Data and Account endpoints.
- Asyncio implementation
- Testnet support for Spot, Futures and Vanilla Options
- 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
- Margin Trading
- Futures Trading
- Vanilla Options
- Support other domains (.us, .jp, etc)

# Upgrading to v1.0.0+

The breaking changes include the migration from wapi to sapi endpoints which related to the wallet endpoints detailed in the Binance Docs

The other breaking change is for websocket streams and the Depth Cache Manager which have been converted to use Asynchronous Context Managers. See examples in the Async section below or view the websockets and depth cache docs.

# Quick Start

Register an account with Binance.

Generate an API Key and assign relevant permissions.

If you are using an exchange from the US, Japan or other TLD then make sure pass *tld='us'* when creating the client.

To use the Spot or Vanilla Options Testnet, pass *testnet=True* when creating the client.

```
pip install python-binance
```

```python
from binance import Client, ThreadedWebsocketManager, ThreadedDepthCacheManager
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
try:
    result = client.withdraw(
        asset='ETH',
        address='<eth_address>',
        amount=100)
```

```python
except BinanceAPIException 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(coin='ETH')

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

# 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", Client.KLINE_INTERVAL_1WEEK, "1 Jan,␣
↪2017")

# socket manager using threads
twm = ThreadedWebsocketManager()
twm.start()

# depth cache manager using threads
dcm = ThreadedDepthCacheManager()
dcm.start()

def handle_socket_message(msg):
    print(f"message type: {msg['e']}")
    print(msg)

def handle_dcm_message(depth_cache):
    print(f"symbol {depth_cache.symbol}")
    print("top 5 bids")
    print(depth_cache.get_bids()[:5])
    print("top 5 asks")
    print(depth_cache.get_asks()[:5])
    print("last update time {}".format(depth_cache.update_time))

twm.start_kline_socket(callback=handle_socket_message, symbol='BNBBTC')

dcm.start_depth_cache(callback=handle_dcm_message, symbol='ETHBTC')

# replace with a current options symbol
options_symbol = 'BTC-210430-36000-C'
dcm.start_options_depth_cache(callback=handle_dcm_message, symbol=options_symbol)

# join the threaded managers to the main thread
```

```
twm.join()
dcm.join()
```

For more check out the documentation.

# Async Example

Read Async basics for Binance for more information.

```python
import asyncio
import json

from binance import AsyncClient, DepthCacheManager, BinanceSocketManager

async def main():

    # initialise the client
    client = await AsyncClient.create()

    # run some simple requests
    print(json.dumps(await client.get_exchange_info(), indent=2))

    print(json.dumps(await client.get_symbol_ticker(symbol="BTCUSDT"), indent=2))

    # initialise websocket factory manager
    bsm = BinanceSocketManager(client)

    # create listener using async with
    # this will exit and close the connection after 5 messages
    async with bsm.trade_socket('ETHBTC') as ts:
        for _ in range(5):
            res = await ts.recv()
            print(f'recv {res}')

    # 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", AsyncClient.KLINE_INTERVAL_
→1MINUTE, "1 day ago UTC")

    # use generator to fetch 1 minute klines for the last day up until now
```

```python
    async for kline in await client.get_historical_klines_generator("BNBBTC",
→AsyncClient.KLINE_INTERVAL_1MINUTE, "1 day ago UTC"):
        print(kline)

    # 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", Client.KLINE_INTERVAL_1WEEK, "1
→Jan, 2017")

    # setup an async context the Depth Cache and exit after 5 messages
    async with DepthCacheManager(client, symbol='ETHBTC') as dcm_socket:
        for _ in range(5):
            depth_cache = await dcm_socket.recv()
            print(f"symbol {depth_cache.symbol} updated:{depth_cache.update_time}")
            print("Top 5 asks:")
            print(depth_cache.get_asks()[:5])
            print("Top 5 bids:")
            print(depth_cache.get_bids()[:5])

    # Vanilla options Depth Cache works the same, update the symbol to a current one
    options_symbol = 'BTC-210430-36000-C'
    async with OptionsDepthCacheManager(client, symbol=options_symbol) as dcm_socket:
        for _ in range(5):
            depth_cache = await dcm_socket.recv()
            count += 1
            print(f"symbol {depth_cache.symbol} updated:{depth_cache.update_time}")
            print("Top 5 asks:")
            print(depth_cache.get_asks()[:5])
            print("Top 5 bids:")
            print(depth_cache.get_bids()[:5])

    await client.close_connection()

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

# CHAPTER 5

# Donate

If this library helped you out feel free to donate.

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

# Other Exchanges

If you use Binance Chain check out my python-binance-chain library.

If you use Kucoin check out my python-kucoin library.

If you use IDEX check out my python-idex library.

## 6.1 Contents

### 6.1.1 Getting Started

#### Installation

`python-binance` is available on PYPI. Install with `pip`:

```
pip install python-binance
```

#### 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)
```

or for Asynchronous client

```
async def main():

    # initialise the client
    client = await AsyncClient.create(api_key, api_secret)

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

## Using the Spot, Futures or Vanilla Options Testnet

Binance offers a Spot, Futures and Vanilla Options Testnet, to test interacting with the exchange.

To enable this set the *testnet* parameter passed to the Client to True.

The testnet parameter will also be used by any websocket streams when the client is passed to the BinanceSocketManager.

```
client = Client(api_key, api_secret, testnet=True)
```

or for Asynchronous client

```
client = await AsyncClient.create(api_key, api_secret, testnet=True)
```

## Using a different TLD

If you are interacting with a regional version of Binance which has a different TLD such as *.us* or '.jp' then you will need to pass this when creating the client, see examples below.

This tld will also be used by any websocket streams when the client is passed to the BinanceSocketManager.

```
client = Client(api_key, api_secret, tld='us')
```

or for Asynchronous client

```
client = await AsyncClient.create(api_key, api_secret, tld='us')
```

## Making API Calls

Every method supports the passing of arbitrary parameters via keyword matching those in the Binance API documentation. 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.

### Async API Calls

aiohttp is used to handle asyncio REST requests.

Each function available in the normal client is available in the AsyncClient class.

The only difference is to run within an asyncio event loop and await the function like below.

```python
import asyncio
from binance import AsyncClient

async def main():
    client = await AsyncClient.create()

    # fetch exchange info
    res = await client.get_exchange_info()
    print(json.dumps(res, indent=2))

    await client.close_connection()

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Read Async basics for Binance for more information about asynchronous patterns.

### 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 for specific information.

On each request Binance returns *X-MBX-USED-WEIGHT-(intervalNum)(intervalLetter)* and *X-MBX-ORDER-COUNT-(intervalNum)* headers.

Here are examples to access these

Asynchronous example

```python
import asyncio
from binance import AsyncClient

api_key = '<api_key>'
api_secret = '<api_secret>'

async def main():
    client = await AsyncClient.create(api_key, api_secret)

    res = await client.get_exchange_info()
    print(client.response.headers)
```

(continues on next page)

```python
    await client.close_connection()

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Synchronous example

```python
from binance import Client

api_key = '<api_key>'
api_secret = '<api_secret>'

def main():
    client = Client(api_key, api_secret)

    res = client.get_exchange_info()
    print(client.response.headers)

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

## Requests Settings

*python-binance* uses the requests library.

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

```python
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 settingsspecify new ones like the example below.

```python
# 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

```python
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
```

## 6.1.2 Binance Constants

Binance requires specific string constants 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_3MINUTE = '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'
```

(continues on next page)

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

# For accessing the data returned by Client.aggregate_trades().
AGG_ID             = 'a'
AGG_PRICE          = 'p'
AGG_QUANTITY       = 'q'
AGG_FIRST_TRADE_ID = 'f'
AGG_LAST_TRADE_ID  = 'l'
AGG_TIME           = 'T'
AGG_BUYER_MAKES    = 'm'
AGG_BEST_MATCH     = 'M'
```

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
```

### 6.1.3 General Endpoints

**Ping the server**

```
client.ping()
```

**Get the server time**

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

**Get system status**

```
status = client.get_system_status()
```

Returns

```
{
    "status": 0,       # 0: normal1system maintenance
    "msg": "normal"    # normal or System maintenance.
}
```

### 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 All Coins Info

Get information of coins (available for deposit and withdraw) for user

```
info = client.get_all_tickers()
```

### Get Get Daily Account Snapshot

Get daily account snapshot of specific type. Valid types: SPOT/MARGIN/FUTURES.

```
info = client.get_account_snapshot(type='SPOT')
```

### Get Current Products

This call is deprecated, use the above Exchange Info call

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

## 6.1.4 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')
```

### Aggregate Trade Iterator

Iterate over aggregate trades for a symbol from a given date or a given order id.

```
agg_trades = client.aggregate_trade_iter(symbol='ETHBTC', start_str='30 minutes ago
→UTC')

# iterate over the trade iterator
for trade in agg_trades:
    print(trade)
    # do something with the trade data

# convert the iterator to a list
# note: generators can only be iterated over once so we need to call it again
agg_trades = client.aggregate_trade_iter(symbol='ETHBTC', '30 minutes ago UTC')
agg_trade_list = list(agg_trades)

# example using last_id value
agg_trades = client.aggregate_trade_iter(symbol='ETHBTC', last_id=23380478)
agg_trade_list = list(agg_trades)
```

### 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("NEOBTC", Client.KLINE_INTERVAL_1WEEK, "1 Jan,
→2017")
```

### Get Historical Kline/Candlesticks using a generator

Fetch klines using a generator

```
for kline in client.get_historical_klines_generator("BNBBTC", Client.KLINE_INTERVAL_
↪1MINUTE, "1 day ago UTC")
    print(kline)
    # do something with the kline
```

### Get average price for a symbol

```
avg_price = client.get_avg_price(symbol='BNBBTC')
```

### 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()
```

## 6.1.5 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.

Read Understanding Binance Order Filters for more information about price and quantity filters on Binance.

It can be helpful to format the output using formatting

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

Or if you have the tickSize or stepSize then use the helper to round to step size

```python
from binance.helpers import round_step_size

amount = 0.000234234
tick_size = 0.00001
rounded_amount = round_step_size(amount, tick_size)
```

## Fetch all orders

```python
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

```python
from binance.enums import *
order = client.create_order(
    symbol='BNBBTC',
    side=SIDE_BUY,
    type=ORDER_TYPE_LIMIT,
    timeInForce=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

```python
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

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

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

### Place an OCO order

Use the *create_oco_order* function to have full control over creating an OCO order

```python
from binance.enums import *
order = client.create_oco_order(
    symbol='BNBBTC',
    side=SIDE_SELL,
    stopLimitTimeInForce=TIME_IN_FORCE_GTC,
    quantity=100,
    stopPrice='0.00001',
    price='0.00002')
```

### Place a test order

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

```python
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

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

### Cancel an order

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

### Get all open orders

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

### Get all orders

```python
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 account API trading status

```
status = client.get_account_api_trading_status()
```

### Get trades

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

### Get trade fees

```
# get fees for all symbols
fees = client.get_trade_fee()

# get fee for one symbol
fees = client.get_trade_fee(symbol='BNBBTC')
```

### Get asset details

```
details = client.get_asset_details()
```

### Get dust log

```
log = client.get_dust_log()
```

### Transfer dust

```
transfer = client.transfer_dust(asset='BNZ')
```

### Get Asset Dividend History

```
history = client.get_asset_dividend_history()
```

### Disable Fast Withdraw Switch

```
client.disable_fast_withdraw_switch()
```

### Enable Fast Withdraw Switch

```
client.enable_fast_withdraw_switch()
```

## 6.1.6 Sub Account Endpoints

### Get Sub Account list

```
accounts = client.get_sub_account_list()
```

### Get Sub Account Transfer History

```
history = client.get_sub_account_transfer_history(fromEmail='blah@gmail.com', toEmail=
↪'foo@gmail.com')
```

### Get Sub Account Assets

```
assets = client.get_sub_account_assets(email='blah@gmail.com')
```

## 6.1.7 Margin Trading Endpoints

---

**Note: Cross-margin vs isolated margin trading**

Binance offers both *cross-margin* trading (where all margin is in one account) and *isolated margin* trading (where each pair is a separate margin account). Make sure you are interacting with the right one.

Some of the API endpoints apply to the cross-margin or isolated margin accounts only. Other endpoints, such as the trade execution endpoints, are used for the cross-margin account trades by default, but you can use your isolated margin accounts by using the `isIsolated` or `isolatedSymbol` parameters. See the documentation below.

---

### Market Data

### Get cross-margin asset info

```
info = client.get_margin_asset(asset='BNB')
```

### Get cross-margin symbol info

```
info = client.get_margin_symbol(symbol='BTCUSDT')
```

### Get isolated margin symbol info

```
info = client.get_isolated_margin_symbol(symbol='BTCUSDT')
```

### Get all isolated margin symbols

```
info = client.get_all_isolated_margin_symbols()
```

### Get margin price index

```
info = client.get_margin_price_index(symbol='BTCUSDT')
```

### Orders

### Cross-margin vs isolated margin orders

By default, these trade execution endpoints will create an order using the *cross-margin* account.

To use the *isolated margin* account for the `symbol` you have specified, simply add the `isIsolated='TRUE'` parameter to the API calls below in this 'Orders' section.

### 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 margin_orders

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

### Place a margin order

Use the *create_margin_order* function to have full control over creating an order

```
from binance.enums import *
order = client.create_margin_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_margin_order(
    symbol='BNBBTC',
    orderId='orderId')
```

### Cancel a margin order

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

### Get all open margin orders

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

For isolated margin, add the `isIsolated='TRUE'` parameter.

### Get all margin orders

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

For isolated margin, add the `isIsolated='TRUE'` parameter.

### Account

### Get cross-margin account info

```
info = client.get_margin_account()
```

### Create isolated margin account

```
account = client.create_isolated_margin_account(base='BTC', quote='ETH')
```

### Get isolated margin account info

```
info = client.get_isolated_margin_account()
```

### Transfer spot to cross-margin account

```
transaction = client.transfer_spot_to_margin(asset='BTC', amount='1.1')
```

### Transfer cross-margin account to spot

```
transaction = client.transfer_margin_to_spot(asset='BTC', amount='1.1')
```

### Transfer spot to isolated margin account

```
transaction = client.transfer_spot_to_isolated_margin(asset='BTC',
                                            symbol='ETHBTC', amount='1.1')
```

### Transfer isolated margin account to spot

```
transaction = client.transfer_isolated_margin_to_spot(asset='BTC',
                                            symbol='ETHBTC', amount='1.1')
```

### Get max transfer amount

```
details = client.get_max_margin_transfer(asset='BTC')
```

This max transfer is for the cross-margin account by default. For isolated margin records, add the `isolatedSymbol=symbol_name` parameter.

### Trades

### Get all margin trades

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

For isolated margin trades, add the `isIsolated='TRUE'` parameter.

### Loans

### Create loan

```
transaction = client.create_margin_loan(asset='BTC', amount='1.1')
```

This for the cross-margin account by default. For isolated margin, add the `isIsolated='TRUE'` and the `symbol=symbol_name` parameters.

### Repay loan

```
transaction = client.repay_margin_loan(asset='BTC', amount='1.1')
```

This for the cross-margin account by default. For isolated margin, add the `isIsolated='TRUE'` and the `symbol=symbol_name` parameters.

### Get loan details

```
details = client.get_margin_loan_details(asset='BTC', txId='100001')
```

This for the cross-margin account by default. For isolated margin records, add the `isolatedSymbol=symbol_name` parameter.

### Get repay details

```
details = client.get_margin_repay_details(asset='BTC', txId='100001')
```

This for the cross-margin account by default. For isolated margin records, add the `isolatedSymbol=symbol_name` parameter.

### Get max loan amount

```
details = client.get_max_margin_loan(asset='BTC')
```

The max loan is for the cross-margin account by default. For isolated margin records, add the `isolatedSymbol=symbol_name` parameter.

## 6.1.8 Websockets

There are 2 ways to interact with websockets.

with ThreadedWebsocketManager or BinanceSocketManager.

ThreadedWebsocketManager does not require asyncio programming, while BinanceSocketManager does.

ThreadedWebsocketManager function begin with *start_*, e.g *start_ticker_socket* while BinanceSocketManager is simply *ticker_socket*.

Multiple socket connections can be made through either 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.

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 with an exponential backoff strategy.

### ThreadedWebsocketManager Websocket Usage

Starting sockets on the ThreadedWebsocketManager requires a callback parameter, similar to the old implementations of websockets on python-binance.

ThreadedWebsocketManager takes similar parameters to the Client class as it creates an AsyncClient internally.

For authenticated streams *api_key* and *api_stream* are required.

As these use threads *start()* is required to be called before starting any sockets.

To keep the ThreadedWebsocketManager running, use *join()* to join it to the main thread.

```python
import time

from binance import ThreadedWebsocketManager

api_key = '<api_key>'
api_secret = '<api_secret>'

def main():

    symbol = 'BNBBTC'

    twm = ThreadedWebsocketManager(api_key=api_key, api_secret=api_secret)
    # start is required to initialise its internal loop
    twm.start()

    def handle_socket_message(msg):
        print(f"message type: {msg['e']}")
        print(msg)

    twm.start_kline_socket(callback=handle_socket_message, symbol=symbol)

    # multiple sockets can be started
    twm.start_depth_socket(callback=handle_socket_message, symbol=symbol)

    # or a multiplex socket can be started like this
    # see Binance docs for stream names
```

```python
    streams = ['bnbbtc@miniTicker', 'bnbbtc@bookTicker']
    twm.start_multiplex_socket(callback=handle_socket_message, streams=streams)

    twm.join()


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

### Stop Individual Stream

When starting a stream, a name for that stream will be returned. This can be used to stop that individual stream.

```python
from binance import ThreadedWebsocketManager

symbol = 'BNBBTC'

twm = ThreadedWebsocketManager()
# start is required to initialise its internal loop
twm.start()

def handle_socket_message(msg):
    print(f"message type: {msg['e']}")
    print(msg)

    twm.start_kline_socket(callback=handle_socket_message, symbol=symbol)
depth_stream_name = twm.start_depth_socket(callback=handle_socket_message,
→symbol=symbol)

# some time later

twm.stop_socket(depth_stream_name)
```

### Stop All Streams

```python
from binance import ThreadedWebsocketManager

twm = ThreadedWebsocketManager()
# start is required to initialise its internal loop
twm.start()

def handle_socket_message(msg):
    print(f"message type: {msg['e']}")
    print(msg)

depth_stream_name = twm.start_depth_socket(callback=handle_socket_message,
→symbol=symbol)

twm.stop()
```

Attempting to start a stream after *stop* is called will not work.

### BinanceSocketManager Websocket Usage

Create the manager like so, passing an AsyncClient.

---

```python
import asyncio
from binance import AsyncClient, BinanceSocketManager


async def main():
    client = await AsyncClient.create()
    bm = BinanceSocketManager(client)
    # start any sockets here, i.e a trade socket
    ts = bm.trade_socket('BNBBTC')
    # then start receiving messages
    async with ts as tscm:
        while True:
            res = await tscm.recv()
            print(res)

    await client.close_connection()

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Set a custom timeout for the websocket connections

```python
# set a timeout of 60 seconds
bm = BinanceSocketManager(client, user_timeout=60)
```

Manually enter and exit the Asynchronous context manager

```python
ts = bm.trade_socket('BNBBTC')
# enter the context manager
await ts.__aenter__()
# receive a message
msg = await ts.recv()
print(msg)
# exit the context manager
await ts.__aexit__(None, None, None)
```

## Using a different TLD

The ThreadedWebsocketManager can take the tld when created if required.

```python
from binance.streams import ThreadedWebsocketManager

twm = ThreadedWebsocketManager(tld='us')
```

The BinanceSocketManager uses the same tld value as the AsyncClient that is passed in. To use the 'us' tld we can do this.

```python
from binance import AsyncClient, BinanceSocketManager


async def x():
    client = await AsyncClient.create(tld='us')
    bm = BinanceSocketManager(client)

    # start a socket...
```

(continues on next page)

```
    await client.close_connection()
```

## 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.

```
# pass a list of stream names
ms = bm.multiplex_socket(['bnbbtc@aggTrade', 'neobtc@ticker'])
```

## 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
ds = bm.depth_socket('BNBBTC')

# partial book response
ds = bm.depth_socket('BNBBTC', 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 *
ks = bm.kline_socket('BNBBTC', interval=KLINE_INTERVAL_30MINUTE)
```

### Aggregated Trade Socket

```
ats = bm.aggtrade_socket('BNBBTC')
```

### Trade Socket

```
ts = bm.trade_socket('BNBBTC')
```

### Symbol Ticker Socket

```
sts = bm.symbol_ticker_socket('BNBBTC')
```

### Ticker Socket

```
ts = bm.ticker_socket(process_message)
```

### Mini Ticker Socket

```
# by default updates every second
mts = bm.miniticker_socket()

# this socket can take an update interval parameter
# set as 5000 to receive updates every 5 seconds
mts = bm.miniticker_socket(5000)
```

### 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.

There are separate sockets for Spot, Cross-margin and separate Isolated margin accounts.

### Spot trading

```
bm.user_socket()
```

## Cross-margin

```
bm.margin_socket()
```

## Isolated margin

```
bm.isolated_margin_socket(symbol)
```

### 6.1.9 Depth Cache

To follow the depth cache updates for a symbol there are 2 options similar to websockets.

Use the DepthCacheManager (or OptionsDepthCacheManager for vanilla options) or use the ThreadedDepthCache-Manager if you don't want to interact with asyncio.

#### ThreadedDepthCacheManager Websocket Usage

Starting sockets on the ThreadedDepthCacheManager requires a callback parameter, similar to old implementations of depth cache on python-binance pre v1

ThreadedDepthCacheManager takes similar parameters to the Client class as it creates an AsyncClient internally.

As these use threads *start()* is required to be called before starting any depth cache streams.

To keep the ThreadedDepthCacheManager running using *join()* to join it to the main thread.

```python
from binance import ThreadedDepthCacheManager

def main():

    dcm = ThreadedDepthCacheManager()
    # start is required to initialise its internal loop
    dcm.start()

    def handle_depth_cache(depth_cache):
        print(f"symbol {depth_cache.symbol}")
        print("top 5 bids")
        print(depth_cache.get_bids()[:5])
        print("top 5 asks")
        print(depth_cache.get_asks()[:5])
        print("last update time {}".format(depth_cache.update_time))

    dcm_name = dcm.start_depth_cache(handle_depth_cache, symbol='BNBBTC')

    # multiple depth caches can be started
    dcm_name = dcm.start_depth_cache(handle_depth_cache, symbol='ETHBTC')

    dcm.join()
```

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

### Stop Individual Depth Cache

When starting a stream, a name for that stream will be returned. This can be used to stop that individual stream

```python
from binance import ThreadedDepthCacheManager

symbol = 'BNBBTC'

dcm = ThreadedDepthCacheManager()
dcm.start()

def handle_depth_cache(depth_cache):
    print(f"message type: {msg['e']}")
    print(msg)

dcm_name = dcm.start_depth_cache(handle_depth_cache, symbol='BNBBTC')

# some time later

dcm.stop_socket(dcm_name)
```

### Stop All Depth Cache streams

```python
from binance import ThreadedDepthCacheManager

symbol = 'BNBBTC'

dcm = ThreadedDepthCacheManager()
dcm.start()

def handle_depth_cache(depth_cache):
    print(f"message type: {msg['e']}")
    print(msg)

dcm_name = dcm.start_depth_cache(handle_depth_cache, symbol='BNBBTC')

# some time later

dcm.stop()
```

Attempting to start a stream after *stop* is called will not work.

### DepthCacheManager or OptionsDepthCacheManager Usage

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

```python
import asyncio

from binance import AsyncClient, DepthCacheManager


async def main():
```

```python
    client = await AsyncClient.create()
    dcm = DepthCacheManager(client, 'BNBBTC')

    async with dcm as dcm_socket:
        while True:
            depth_cache = await dcm_socket.recv()
            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])
            print("last update time {}".format(depth_cache.update_time))

if __name__ == "__main__":

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

The *DepthCacheManager* returns an Asynchronous Context Manager which can be used with *async for* or by interacting with the *__aenter__* and *__aexit__* functions

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.

### Share a Socket Manager

Here dcm1 and dcm2 share the same instance of BinanceSocketManager

```python
from binance.websockets import BinanceSocketManager
from binance.depthcache import DepthCacheManager
bm = BinanceSocketManager(client)
dcm1 = DepthCacheManager(client, 'BNBBTC', bm=bm)
dcm2 = DepthCacheManager(client, 'ETHBTC', bm=bm)
```

### Websocket Errors

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

### Examples

```python
# 1 hour interval refresh
dcm = DepthCacheManager(client, 'BNBBTC', refresh_interval=60*60)

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

```python
async with dcm as dcm_socket:
    while True:
        depth_cache = await dcm_socket.recv()
        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])
        print("last update time {}".format(depth_cache.update_time))
```

To use the magic *__aenter__* and *__aexit__* functions to use this class without the *async with*

```
dcm = DepthCacheManager(client, 'BNBBTC')

await dcm.__aenter__()
depth_cache = await dcm.recv()
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])
print("last update time {}".format(depth_cache.update_time))

# exit the context manager
await dcm.__aexit__(None, None, None)
```

## 6.1.10 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.

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

# passing a name parameter
result = client.withdraw(
    coin='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(
    coin='XRP',
    address='<xrp_address>',
```

```
    addressTag='<xrp_address_tag>',
    amount=10000)
```

### Fetch deposit history

```
deposits = client.get_deposit_history()
btc_deposits = client.get_deposit_history(coin='BTC')
```

### Fetch withdraw history

```
withdraws = client.get_withdraw_history()
btc_withdraws = client.get_withdraw_history(coin='BTC')
```

### Get deposit address

```
address = client.get_deposit_address(coin='BTC')
```

## 6.1.11 Helper Functions

binance.**helpers**
    alias of *binance.helpers*

## 6.1.12 Exceptions

### BinanceRequestException

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
```

### 6.1.13 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.

### 6.1.14 Changelog

#### v1.0.15 - 2021-09-27

**Added**

- Enable/disable margin account for symbol endpoints
- Top trader long/short positions endpoint
- Global long/short ratio endpoint

**Fixed**

- fix websockets to 9.1
- websocket reconnect updates
- fix futures kline sockets

#### v1.0.14 - 2021-09-08

**Fixed**

- websocket reconnecting

#### v1.0.13 - 2021-09-08

**Added**

- Futures Depth Cache Manager
- Futures kline websocket stream
- Coin Futures User websocket stream
- New Margin endpoints
- Margin OCO order endpoints

- Fiat endpoints
- C2C endpoints
- Account API permissions endpoint

**Fixed**

- changed *asset* to *coin* in withdraw endpoint

### v1.0.12 - 2021-06-03

**Added**

- coin futures batch order function

**Fixed**

- threaded websockets on python3.9
- filter out None params in request kwargs
- deconflict streams with same name on different websocket urls
- reduce close timeout on websocket close to short time to reduce waiting

### v1.0.10 - 2021-05-13

**Added**

- futures multi-asset margin mode endpoints
- optional symbol param to get_all_tickers

**Fixed**

- start_multiplex_socket remove lower case filter on stream names

### v1.0.9 - 2021-05-12

**Fixed**

- start_book_ticker_socket and start_multiplex_socket to call correct async function

### v1.0.8 - 2021-05-11

**Added**

- old style websocket and depth cache managers as option without interacting with asyncio

**Fixed**

- fixed issue with get_historical_klines in Client
- remove print debug line

### v1.0.7

**Fixed**

- remove version param from get_sub_account_assets

---

### v1.0.6

**Fixed**

- fix time for authenticated stream keepalive

### v1.0.5

**Fixed**

- Restored access to last response on client

### v1.0.4

**Added**

- Futures Testnet support
- Kline type for fetching historical klines

**Fixed**

- Spot Testnet websocket URL

### v1.0.3

**Added**

- Spot Testnet support

### v1.0.2

**Added**

- start of typing to client and websockets

**Fixed**

- end_str, limit, spot params in kline fetching
- drop None values in params passed

**Updated**

- more examples in docs

### v1.0.1

**Fixed**

- restored params for Client and AsyncClient classes

## v1.0.0

**Added**

- Async support for all REST endpoints
- USD-M and Coin-M Futures websocket streams
- Websockets use same tld as Client
- convert type option for DepthCache

**Breaking Changes**

- Supports only py3.6+
- All wapi calls changed to sapi
- Websockets have changed to use Asynchronous context managers

**Fixed**

- get_historical_klines params

## v0.7.11

**Added** - Vanilla Options REST endpoints - Vanilla Options websockets - Futures order type enums

**Updated**

- websocket keep-alive functions for different socket types
- dependencies

**Fixed**

- change to User-Agent to avoid connection issues

## v0.7.5.dev

**Changed** - Stock json lib to ujson (https://github.com/sammchardy/python-binance/pull/383)

## v0.7.5 - 2020-02-06

**Added**

- Futures REST endpoints
- Lending REST endpoints
- OCO Orders function *create_oco_order*, *order_oco_buy*, *order_oco_sell*
- Average Price function *get_avg_price*
- Support for other domains (.us, .jp, etc)

**Updated**

- dependencies

**Fixed**

- websocket keepalive callback not found

---

### v0.7.4 - 2019-09-22

**Added**

- symbol book ticker websocket streams
- margin websocket stream

**Updated**

- can call Client without any params
- make response a property of the Client class so you can access response properties after a request

**Fixed**

- issue with None value params causing errors

### v0.7.3 - 2019-08-12

**Added**

- sub account endpoints
- dust transfer endpoint
- asset divident history endpoint

**Removed**

- deprecated withdraw fee endpoint

### v0.7.2 - 2019-08-01

**Added**

- margin trading endpoints

**Fixed**

- depth cache clearing bug

### v0.7.1 - 2019-01-23

**Added**

- limit param to DepthCacheManager
- limit param to get_historical_klines
- update_time to DepthCache class

**Updated**

- test coverage

**Fixed**

- super init in Websocket class
- removal of request params from signature
- empty set issue in aggregate_trade_iter

### v0.7.0 - 2018-08-08

**Added**

- get_asset_details endpoint
- get_dust_log endpoint
- get_trade_fee endpoint
- ability for multiple DepthCacheManagers to share a BinanceSocketManager
- get_historial_klines_generator function
- custom socket timeout param for BinanceSocketManager

**Updated**

- general dependency version
- removed support for python3.3

**Fixed**

- add a super init on BinanceClientProtocol

### v0.6.9 - 2018-04-27

**Added**

- timestamp in milliseconds to *get_historical_klines* function
- timestamp in milliseconds to *aggregate_trade_iter* function

**Fixed**

- Don't close user stream listen key on socket close

### v0.6.8 - 2018-03-29

**Added**

- *get_withdraw_fee* function

**Fixed**

- Remove unused LISTENKEY_NOT_EXISTS
- Optimise the historical klines function to reduce requests
- Issue with end_time in aggregate trade iterator

### v0.6.7 - 2018-03-14

**Fixed**

- Issue with *get_historical_klines* when response had exactly 500 results
- Changed BinanceResponseException to BinanceRequestException
- Set default code value in BinanceApiException properly

### v0.6.6 - 2018-02-17

**Fixed**

- User stream websocket keep alive strategy updated

### v0.6.5 - 2018-02-13

**Fixed**

- *get_historical_klines* response for month interval

### v0.6.4 - 2018-02-09

**Added**

- system status endpoint *get_system_status*

### v0.6.3 - 2018-01-29

**Added**

- mini ticker socket function *start_miniticker_socket*
- aggregate trade iterator *aggregate_trade_iter*

**Fixes**

- clean up *interval_to_milliseconds* logic
- general doc and file cleanups

### 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

## 6.1.15 Binance API

### client module

**class** binance.client.**AsyncClient**(*api_key: Optional[str] = None, api_secret: Optional[str] = None, requests_params: Dict[str, str] = None, tld: str = 'com', testnet: bool = False, loop=None*)

    Bases: *binance.client.BaseClient*

    **__init__**(*api_key: Optional[str] = None, api_secret: Optional[str] = None, requests_params: Dict[str, str] = None, tld: str = 'com', testnet: bool = False, loop=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

- **testnet** (`bool`) – Use testnet environment - only available for vanilla options at the moment

    **aggregate_trade_iter**(*symbol*, *start_str=None*, *last_id=None*)

        Iterate over aggregate trade data from (start_time or last_id) to the end of the history so far.

        If start_time is specified, start with the first trade after start_time. Meant to initialise a local cache of trade data.

        If last_id is specified, start with the trade after it. This is meant for updating a pre-existing local trade data cache.

        Only allows start_str or last_id—not both. Not guaranteed to work right if you're running more than one of these simultaneously. You will probably hit your rate limit.

        See dateparser 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`) – Symbol string e.g. ETHBTC

- **start_str** – Start date string in UTC format or timestamp in milliseconds. The iterator will

        return the first trade occurring later than this time. :type start_str: str|int :param last_id: aggregate trade ID of the last known aggregate trade. Not a regular trade ID. See https://binance-docs.github.io/apidocs/spot/en/#compressed-aggregate-trades-list

        **Returns** an iterator of JSON objects, one per trade. The format of

        each object is identical to Client.aggregate_trades().

    **cancel_margin_oco_order**(*\*\*params*)

    **cancel_margin_order**(*\*\*params*)

    **cancel_order**(*\*\*params*)

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

        https://binance-docs.github.io/apidocs/spot/en/#cancel-order-trade

**Parameters**

- **symbol** (*str*) – required
- **orderId** (*int*) – The unique order id
- **origClientOrderId** (*str*) – optional
- **newClientOrderId** (*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",
    "origClientOrderId": "myOrder1",
    "orderId": 1,
    "clientOrderId": "cancelMyOrder1"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**change_fixed_activity_to_daily_position**(*\*\*params*)

**close_connection**()

**classmethod create**(*api_key: Optional[str] = None, api_secret: Optional[str] = None, requests_params: Dict[str, str] = None, tld: str = 'com', testnet: bool = False, loop=None*)

**create_isolated_margin_account**(*\*\*params*)

**create_margin_loan**(*\*\*params*)

**create_margin_oco_order**(*\*\*params*)

**create_margin_order**(*\*\*params*)

**create_oco_order**(*\*\*params*)
    Send in a new OCO order

https://binance-docs.github.io/apidocs/spot/en/#new-oco-trade

**Parameters**

- **symbol** (*str*) – required
- **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.
- **side** (*str*) – required
- **quantity** (*decimal*) – required
- **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.
- **price** (*str*) – required
- **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.
- **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated if not sent.

- **stopPrice** (`str`) – required

- **stopLimitPrice** (`str`) – If provided, stopLimitTimeInForce is required.

- **stopIcebergQty** (`decimal`) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.

- **stopLimitTimeInForce** (`str`) – Valid values are GTC/FOK/IOC.

- **newOrderRespType** (`str`) – 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:

```
{
}
```

Response RESULT:

```
{
}
```

Response FULL:

```
{
}
```

**Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_order** (*\*\*params*)
Send in a new order

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

https://binance-docs.github.io/apidocs/spot/en/#new-order-trade

**Parameters**

- **symbol** (`str`) – required

- **side** (`str`) – required

- **type** (`str`) – required

- **timeInForce** (`str`) – required if limit order

- **quantity** (`decimal`) – required

- **quoteOrderQty** (`decimal`) – amount the user wants to spend (when buying) or receive (when selling) of the quote asset, applicable to MARKET orders

- **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** (*str*) – 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",
    "cummulativeQuoteQty": "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",
    "cummulativeQuoteQty": "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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOr-
> derMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalExcep-
> tion, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_sub_account_futures_transfer**(*\*\*params*)

**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://binance-docs.github.io/apidocs/spot/en/#test-new-order-trade

> **Parameters**
> - **symbol** (*str*) – required
> - **side** (*str*) – required
> - **type** (*str*) – required
> - **timeInForce** (*str*) – 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** (*str*) – Set the response JSON. ACK, RESULT, or FULL; de-
>   fault: RESULT.
> - **recvWindow** (*int*) – The number of milliseconds the request is valid for

> **Returns** API response

```
{}
```

> > **Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**disable_fast_withdraw_switch**(*\*\*params*)

**disable_isolated_margin_account**(*\*\*params*)

**enable_fast_withdraw_switch**(*\*\*params*)

**enable_isolated_margin_account**(*\*\*params*)

**enable_subaccount_futures**(*\*\*params*)

**enable_subaccount_margin**(*\*\*params*)

**futures_account**(*\*\*params*)

**futures_account_balance**(*\*\*params*)

**futures_account_trades**(*\*\*params*)

**futures_account_transfer**(*\*\*params*)

**futures_adl_quantile_estimate**(*\*\*params*)

**futures_aggregate_trades**(*\*\*params*)

**futures_cancel_all_open_orders**(*\*\*params*)

**futures_cancel_order**(*\*\*params*)

**futures_cancel_orders**(*\*\*params*)

**futures_change_leverage**(*\*\*params*)

**futures_change_margin_type**(*\*\*params*)

**futures_change_multi_assets_mode**(*multiAssetsMargin: bool*)

**futures_change_position_margin**(*\*\*params*)

**futures_change_position_mode**(*\*\*params*)

**futures_coin_account**(*\*\*params*)

**futures_coin_account_balance**(*\*\*params*)

**futures_coin_account_trades**(*\*\*params*)

**futures_coin_aggregate_trades**(*\*\*params*)

**futures_coin_cancel_all_open_orders**(*\*\*params*)

**futures_coin_cancel_order**(*\*\*params*)

**futures_coin_cancel_orders**(*\*\*params*)

**futures_coin_change_leverage**(*\*\*params*)

**futures_coin_change_margin_type**(*\*\*params*)

**futures_coin_change_position_margin**(*\*\*params*)

**futures_coin_change_position_mode**(*\*\*params*)

**futures_coin_continous_klines**(*\*\*params*)

**futures_coin_create_order**(*\*\*params*)

**futures_coin_exchange_info**()

**futures_coin_funding_rate**(*\*\*params*)

**futures_coin_get_all_orders**(*\*\*params*)

**futures_coin_get_open_orders**(*\*\*params*)

**futures_coin_get_order**(*\*\*params*)

**futures_coin_get_position_mode**(*\*\*params*)

**futures_coin_historical_trades**(*\*\*params*)

**futures_coin_income_history**(*\*\*params*)

**futures_coin_index_price_klines**(*\*\*params*)

**futures_coin_klines**(*\*\*params*)

**futures_coin_leverage_bracket**(*\*\*params*)

**futures_coin_liquidation_orders**(*\*\*params*)

**futures_coin_mark_price**(*\*\*params*)

**futures_coin_mark_price_klines**(*\*\*params*)

**futures_coin_open_interest**(*\*\*params*)

**futures_coin_open_interest_hist**(*\*\*params*)

**futures_coin_order_book**(*\*\*params*)

**futures_coin_orderbook_ticker**(*\*\*params*)

**futures_coin_ping**()

**futures_coin_place_batch_order**(*\*\*params*)

**futures_coin_position_information**(*\*\*params*)

**futures_coin_position_margin_history**(*\*\*params*)

**futures_coin_recent_trades**(*\*\*params*)

**futures_coin_stream_close**(*listenKey*)

**futures_coin_stream_get_listen_key**()

**futures_coin_stream_keepalive**(*listenKey*)

**futures_coin_symbol_ticker**(*\*\*params*)

**futures_coin_ticker**(*\*\*params*)

**futures_coin_time**()

**futures_continous_klines**(*\*\*params*)

**futures_create_order**(*\*\*params*)

**futures_exchange_info**()

**futures_funding_rate**(*\*\*params*)

**futures_get_all_orders**(*\*\*params*)

**futures_get_multi_assets_mode**()

**futures_get_open_orders**(*\*\*params*)

**futures_get_order**(*\*\*params*)

**futures_get_position_mode**(*\*\*params*)

**futures_global_longshort_ratio**(*\*\*params*)

**futures_historical_klines**(*symbol*, *interval*, *start_str*, *end_str=None*, *limit=500*)

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

**futures_historical_trades**(*\*\*params*)

**futures_income_history**(*\*\*params*)

**futures_klines**(*\*\*params*)

**futures_leverage_bracket**(*\*\*params*)

**futures_liquidation_orders**(*\*\*params*)

**futures_mark_price**(*\*\*params*)

**futures_open_interest**(*\*\*params*)

**futures_open_interest_hist**(*\*\*params*)

**futures_order_book**(*\*\*params*)

**futures_orderbook_ticker**(*\*\*params*)

**futures_ping**()

**futures_place_batch_order**(*\*\*params*)

**futures_position_information**(*\*\*params*)

**futures_position_margin_history**(*\*\*params*)

**futures_recent_trades**(*\*\*params*)

**futures_stream_close**(*listenKey*)

**futures_stream_get_listen_key**()

**futures_stream_keepalive**(*listenKey*)

**futures_symbol_ticker**(*\*\*params*)

**futures_ticker**(*\*\*params*)

**futures_time**()

**futures_top_longshort_account_ratio**(*\*\*params*)

**futures_top_longshort_position_ratio**(*\*\*params*)

**get_account**(*\*\*params*)
  Get current account information.

  https://binance-docs.github.io/apidocs/spot/en/#account-information-user_data

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

  > **Returns** API response

```json
{
    "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** BinanceRequestException, BinanceAPIException

**get_account_api_permissions**(*\*\*params*)

> Fetch api key permissions.
>
> https://binance-docs.github.io/apidocs/spot/en/#get-api-key-permission-user_data
>
> > **Parameters** **recvWindow** (*int*) – the number of milliseconds the request is valid for
> >
> > **Returns** API response

```json
{
   "ipRestrict": false,
   "createTime": 1623840271000,
   "enableWithdrawals": false,   // This option allows you to withdraw via
→API. You must apply the IP Access Restriction filter in order to enable
→withdrawals
   "enableInternalTransfer": true,  // This option authorizes this key to
→transfer funds between your master account and your sub account instantly
   "permitsUniversalTransfer": true,  // Authorizes this key to be used for a
→dedicated universal transfer API to transfer multiple supported currencies.
→Each business's own transfer API rights are not affected by this
→authorization
   "enableVanillaOptions": false,  //  Authorizes this key to Vanilla options
→trading
   "enableReading": true,
   "enableFutures": false,  //  API Key created before your futures account
→opened does not support futures API service
   "enableMargin": false,   //  This option can be adjusted after the Cross
→Margin account transfer is completed
   "enableSpotAndMarginTrading": false, // Spot and margin trading
   "tradingAuthorityExpirationTime": 1628985600000  // Expiration time for
→spot and margin trading permission
}
```

**get_account_api_trading_status**(*\*\*params*)

---

Fetch account api trading status detail.

https://binance-docs.github.io/apidocs/spot/en/#account-api-trading-status-sapi-user_data

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

> **Returns** API response

```
{
    "data": {           // API trading status detail
        "isLocked": false,   // API trading function is locked or not
        "plannedRecoverTime": 0,  // If API trading function is locked, this
→is the planned recover time
        "triggerCondition": {
                "GCR": 150,  // Number of GTC orders
                "IFER": 150, // Number of FOK/IOC orders
                "UFR": 300   // Number of orders
        },
        "indicators": {  // The indicators updated every 30 seconds
            "BTCUSDT": [  // The symbol
                {
                    "i": "UFR",  // Unfilled Ratio (UFR)
                    "c": 20,     // Count of all orders
                    "v": 0.05,   // Current UFR value
                    "t": 0.995   // Trigger UFR value
                },
                {
                    "i": "IFER", // IOC/FOK Expiration Ratio (IFER)
                    "c": 20,     // Count of FOK/IOC orders
                    "v": 0.99,   // Current IFER value
                    "t": 0.99    // Trigger IFER value
                },
                {
                    "i": "GCR",  // GTC Cancellation Ratio (GCR)
                    "c": 20,     // Count of GTC orders
                    "v": 0.99,   // Current GCR value
                    "t": 0.99    // Trigger GCR value
                }
            ],
            "ETHUSDT": [
                {
                    "i": "UFR",
                    "c": 20,
                    "v": 0.05,
                    "t": 0.995
                },
                {
                    "i": "IFER",
                    "c": 20,
                    "v": 0.99,
                    "t": 0.99
                },
                {
                    "i": "GCR",
                    "c": 20,
                    "v": 0.99,
                    "t": 0.99
                }
            ]
```

```
        },
        "updateTime": 1547630471725
    }
}
```

**get_account_snapshot**(*\*\*params*)

**get_account_status**(*\*\*params*)
    Get account status detail.

    https://binance-docs.github.io/apidocs/spot/en/#account-status-sapi-user_data

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

        **Returns** API response

```
{
    "data": "Normal"
}
```

**get_aggregate_trades**(*\*\*params*) → Dict[KT, VT]
    Get compressed, aggregate trades. Trades that fill at the time, from the same order, with the same price
    will have the quantity aggregated.

    https://binance-docs.github.io/apidocs/spot/en/#compressed-aggregate-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 1000.

        **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** BinanceRequestException, BinanceAPIException

**get_all_coins_info**(*\*\*params*)

**get_all_isolated_margin_symbols**(*\*\*params*)

**get_all_margin_orders**(*\*\*params*)

**get_all_orders**(*\*\*params*)

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

https://binance-docs.github.io/apidocs/spot/en/#all-orders-user_data

>   Parameters
>
>   - **symbol** (*str*) – required
>
>   - **orderId** (*int*) – The unique order id
>
>   - **startTime** (*int*) – optional
>
>   - **endTime** (*int*) – optional
>
>   - **limit** (*int*) – Default 500; max 1000.
>
>   - **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   BinanceRequestException, BinanceAPIException

**get_all_tickers**(*symbol: Optional[str] = None*) → List[Dict[str, str]]

Latest price for all symbols.

https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker

>   Returns   List of market tickers

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

>   Raises   BinanceRequestException, BinanceAPIException

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

    Get current asset balance.

        **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** BinanceRequestException, BinanceAPIException

**get_asset_details**(*\*\*params*)

    Fetch details on assets.

    https://binance-docs.github.io/apidocs/spot/en/#asset-detail-sapi-user_data

        **Parameters**

- **asset** (*str*) – optional

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

        **Returns** API response

```
{
        "CTR": {
            "minWithdrawAmount": "70.00000000", //min withdraw amount
            "depositStatus": false,//deposit status (false if ALL of networks
↪' are false)
            "withdrawFee": 35, // withdraw fee
            "withdrawStatus": true, //withdraw status (false if ALL of␣
↪networks' are false)
            "depositTip": "Delisted, Deposit Suspended" //reason
        },
        "SKY": {
            "minWithdrawAmount": "0.02000000",
            "depositStatus": true,
            "withdrawFee": 0.01,
            "withdrawStatus": true
        }
}
```

**get_asset_dividend_history**(*\*\*params*)

    Query asset dividend record.

    https://binance-docs.github.io/apidocs/spot/en/#asset-dividend-record-user_data

        **Parameters**

- **asset** (*str*) – optional

- **startTime** (*long*) – optional

- **endTime** (*long*) – optional

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

```
result = client.get_asset_dividend_history()
```

**Returns** API response

```
{
    "rows":[
        {
            "amount":"10.00000000",
            "asset":"BHFT",
            "divTime":1563189166000,
            "enInfo":"BHFT distribution",
            "tranId":2968885920
        },
        {
            "amount":"10.00000000",
            "asset":"BHFT",
            "divTime":1563189165000,
            "enInfo":"BHFT distribution",
            "tranId":2968885920
        }
    ],
    "total":2
}
```

**Raises** BinanceRequestException, BinanceAPIException

**get_avg_price**(*\*\*params*)
Current average price for a symbol.

https://binance-docs.github.io/apidocs/spot/en/#current-average-price

**Parameters symbol** (*str*) –

**Returns** API response

```
{
    "mins": 5,
    "price": "9.35751834"
}
```

**get_bnb_burn_spot_margin**(*\*\*params*)

**get_c2c_trade_history**(*\*\*params*)

**get_deposit_address**(*coin: str*, *network: Optional[str] = None*, *\*\*params*)
Fetch a deposit address for a symbol

https://binance-docs.github.io/apidocs/spot/en/#deposit-address-supporting-network-user_data

**Parameters**

- **coin** (*str*) – required

- **network** (*str*) – optional

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

**Returns** API response

```json
{
    "address": "1HPn8Rx2y6nNSfagQBKy27GB99Vbzg89wv",
    "coin": "BTC",
    "tag": "",
    "url": "https://btc.com/1HPn8Rx2y6nNSfagQBKy27GB99Vbzg89wv"
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_deposit_history**(*\*\*params*)
> Fetch deposit history.
>
> https://binance-docs.github.io/apidocs/spot/en/#deposit-history-supporting-network-user_data
>
> **Parameters**
>
> - **coin** (`str`) – optional
>
> - **startTime** (`long`) – optional
>
> - **endTime** (`long`) – optional
>
> - **offset** (`long`) – optional - default:0
>
> - **limit** (`long`) – optional
>
> - **recvWindow** (`int`) – the number of milliseconds the request is valid for
>
> **Returns** API response

```json
[
    {
        "amount":"0.00999800",
        "coin":"PAXG",
        "network":"ETH",
        "status":1,
        "address":"0x788cabe9236ce061e5a892e1a59395a81fc8d62c",
        "addressTag":"",
        "txId":
→"0xaad4654a3234aa6118af9b4b335f5ae81c360b2394721c019b5d1e75328b09f3",
        "insertTime":1599621997000,
        "transferType":0,
        "confirmTimes":"12/12"
    },
    {
        "amount":"0.50000000",
        "coin":"IOTA",
        "network":"IOTA",
        "status":1,
        "address":
→"SIZ9VLMHWATXKV99LH99CIGFJFUMLEHGWVZVNNZXRJJVWBPHYWPPBOSDORZ9EQSHCZAMPVAPGFYQAUUV9DROOXJLN
→",
        "addressTag":"",
        "txId":
→"ESBFVQUTPIWQNJSPXFNHNYHSQNTGKRVKPRABQWTAXCDWOAKDKYWPTVG9BGXNVNKTLEJGESAVXIKIZ9999
→",
        "insertTime":1599620082000,
        "transferType":0,
        "confirmTimes":"1/1"
```

(continues on next page)

```
        }
    ]
```

        **Raises** BinanceRequestException, BinanceAPIException

**get_dust_log**(*\*\*params*)

    Get log of small amounts exchanged for BNB.

    https://binance-docs.github.io/apidocs/spot/en/#dustlog-sapi-user_data

        **Parameters**

- **startTime** (*int*) – optional

- **endTime** (*int*) – optional

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

        **Returns** API response

```
{
    "total": 8,   //Total counts of exchange
    "userAssetDribblets": [
        {
            "totalTransferedAmount": "0.00132256",   // Total transfered BNB␣
→amount for this exchange.
            "totalServiceChargeAmount": "0.00002699",    //Total service␣
→charge amount for this exchange.
            "transId": 45178372831,
            "userAssetDribbletDetails": [           //Details of  this␣
→exchange.
                {
                    "transId": 4359321,
                    "serviceChargeAmount": "0.000009",
                    "amount": "0.0009",
                    "operateTime": 1615985535000,
                    "transferedAmount": "0.000441",
                    "fromAsset": "USDT"
                },
                {
                    "transId": 4359321,
                    "serviceChargeAmount": "0.00001799",
                    "amount": "0.0009",
                    "operateTime": "2018-05-03 17:07:04",
                    "transferedAmount": "0.00088156",
                    "fromAsset": "ETH"
                }
            ]
        },
        {
            "operateTime":1616203180000,
            "totalTransferedAmount": "0.00058795",
            "totalServiceChargeAmount": "0.000012",
            "transId": 4357015,
            "userAssetDribbletDetails": [
                {
                    "transId": 4357015,
                    "serviceChargeAmount": "0.00001"
```

```
                    "amount": "0.001",
                    "operateTime": 1616203180000,
                    "transferedAmount": "0.00049",
                    "fromAsset": "USDT"
                },
                {
                    "transId": 4357015,
                    "serviceChargeAmount": "0.000002"
                    "amount": "0.0001",
                    "operateTime": 1616203180000,
                    "transferedAmount": "0.00009795",
                    "fromAsset": "ETH"
                }
            ]
        }
    ]
}
```

**get_exchange_info**() → Dict[KT, VT]
    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** BinanceRequestException, BinanceAPIException

**get_fiat_deposit_withdraw_history**(*\*\*params*)

**get_fiat_payments_history**(*\*\*params*)

**get_fixed_activity_project_list**(*\*\*params*)

**get_historical_klines**(*symbol*, *interval*, *start_str*, *end_str=None*, *limit=500*, *klines_type: binance.enums.HistoricalKlinesType = <HistoricalKlinesType.SPOT: 1>*)
> Get Historical Klines from Binance

> **Parameters**
> - **symbol** (*str*) – Name of symbol pair e.g BNBBTC
> - **interval** (*str*) – Binance Kline interval
> - **start_str** (*str|int*) – Start date string in UTC format or timestamp in milliseconds
> - **end_str** (*str|int*) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)
> - **limit** (*int*) – Default 500; max 1000.
> - **klines_type** (*HistoricalKlinesType*) – Historical klines type: SPOT or FUTURES

> **Returns** list of OHLCV values

**get_historical_klines_generator**(*symbol*, *interval*, *start_str*, *end_str=None*, *klines_type: binance.enums.HistoricalKlinesType = <HistoricalKlinesType.SPOT: 1>*)
> Get Historical Klines generator from Binance

> **Parameters**
> - **symbol** (*str*) – Name of symbol pair e.g BNBBTC
> - **interval** (*str*) – Binance Kline interval
> - **start_str** (*str|int*) – Start date string in UTC format or timestamp in milliseconds
> - **end_str** (*str|int*) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)

- **klines_type** (*HistoricalKlinesType*) – Historical klines type: SPOT or FU-
TURES

**Returns** generator of OHLCV values

**get_historical_trades**(*\*\*params*) → Dict[KT, VT]
Get older trades.

https://binance-docs.github.io/apidocs/spot/en/#old-trade-lookup

**Parameters**

- **symbol** (*str*) – required

- **limit** (*int*) – Default 500; max 1000.

- **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** BinanceRequestException, BinanceAPIException

**get_isolated_margin_account**(*\*\*params*)

**get_isolated_margin_symbol**(*\*\*params*)

**get_klines**(*\*\*params*) → Dict[KT, VT]
Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.

https://binance-docs.github.io/apidocs/spot/en/#kline-candlestick-data

**Parameters**

- **symbol** (*str*) – required

- **interval** (*str*) –

  –

- **limit** (*int*) –

  – Default 500; max 1000.

- **startTime** (*int*) –

- **endTime** (*int*) –

**Returns** API response

```
[
    [
        1499040000000,      # Open time
        "0.01634790",       # Open
```

(continues on next page)

```
        "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** BinanceRequestException, BinanceAPIException

**get_lending_account** (*\*\*params*)

**get_lending_daily_quota_left** (*\*\*params*)

**get_lending_daily_redemption_quota** (*\*\*params*)

**get_lending_interest_history** (*\*\*params*)

**get_lending_position** (*\*\*params*)

**get_lending_product_list** (*\*\*params*)

**get_lending_purchase_history** (*\*\*params*)

**get_lending_redemption_history** (*\*\*params*)

**get_margin_account** (*\*\*params*)
> Query cross-margin account details
>
> https://binance-docs.github.io/apidocs/spot/en/#query-cross-margin-account-details-user_data
>
> > **Returns** API response

```
{
    "borrowEnabled": true,
    "marginLevel": "11.64405625",
    "totalAssetOfBtc": "6.82728457",
    "totalLiabilityOfBtc": "0.58633215",
    "totalNetAssetOfBtc": "6.24095242",
    "tradeEnabled": true,
    "transferEnabled": true,
    "userAssets": [
        {
            "asset": "BTC",
            "borrowed": "0.00000000",
            "free": "0.00499500",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00499500"
        },
        {
            "asset": "BNB",
            "borrowed": "201.66666672",
            "free": "2346.50000000",
```

```
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "2144.83333328"
        },
        {
            "asset": "ETH",
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000"
        },
        {
            "asset": "USDT",
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000"
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_margin_all_assets**(*\*\*params*)

**get_margin_all_pairs**(*\*\*params*)

**get_margin_asset**(*\*\*params*)

**get_margin_force_liquidation_rec**(*\*\*params*)

**get_margin_interest_history**(*\*\*params*)

**get_margin_loan_details**(*\*\*params*)

**get_margin_oco_order**(*\*\*params*)

**get_margin_order**(*\*\*params*)

**get_margin_price_index**(*\*\*params*)

**get_margin_repay_details**(*\*\*params*)

**get_margin_symbol**(*\*\*params*)

**get_margin_trades**(*\*\*params*)

**get_max_margin_loan**(*\*\*params*)

**get_max_margin_transfer**(*\*\*params*)

**get_my_trades**(*\*\*params*)
   Get trades for a specific symbol.

   https://binance-docs.github.io/apidocs/spot/en/#account-trade-list-user_data

   **Parameters**

   - **symbol** (*str*) – required
   - **startTime** (*int*) – optional

- **endTime** (*int*) – optional

- **limit** (*int*) – Default 500; max 1000.

- **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 BinanceRequestException, BinanceAPIException

**get_open_margin_oco_orders**(*\*\*params*)

**get_open_margin_orders**(*\*\*params*)

**get_open_orders**(*\*\*params*)
Get all open orders on a symbol.

https://binance-docs.github.io/apidocs/spot/en/#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** BinanceRequestException, BinanceAPIException

**get_order**(*\*\*params*)

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

https://binance-docs.github.io/apidocs/spot/en/#query-order-user_data

**Parameters**

- **symbol** (*str*) – required
- **orderId** (*int*) – The unique order id
- **origClientOrderId** (*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** BinanceRequestException, BinanceAPIException

**get_order_book**(*\*\*params*) → Dict[KT, VT]

Get the Order Book for the market

https://binance-docs.github.io/apidocs/spot/en/#order-book

**Parameters**

- **symbol** (*str*) – required
- **limit** (*int*) – Default 100; max 1000

**Returns** API response

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

```
                    "4.00000200",
                    "12.00000000",
                    []
                ]
            ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_orderbook_ticker**(*\*\*params*)
>    Latest price for a symbol or symbols.

>    https://binance-docs.github.io/apidocs/spot/en/#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": "100000.00000000",
        "askQty": "1000.00000000"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_orderbook_tickers**() → Dict[KT, VT]
>    Best price/qty on the order book for all symbols.

>    https://binance-docs.github.io/apidocs/spot/en/#symbol-order-book-ticker

>    **Parameters symbol** (*str*) – optional

>    **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": "100000.00000000",
        "askQty": "1000.00000000"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_products**() → Dict[KT, VT]
 Return list of products currently listed on Binance

 Use get_exchange_info() call instead

> **Returns** list - List of product dictionaries

> **Raises** BinanceRequestException, BinanceAPIException

**get_recent_trades**(*\*\*params*) → Dict[KT, VT]
 Get recent trades (up to last 500).

 https://binance-docs.github.io/apidocs/spot/en/#recent-trades-list

> **Parameters**
>
>  • **symbol** (*str*) – required
>
>  • **limit** (*int*) – Default 500; max 1000.

> **Returns** API response

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

> **Raises** BinanceRequestException, BinanceAPIException

**get_server_time**() → Dict[KT, VT]
 Test connectivity to the Rest API and get the current server time.

 https://binance-docs.github.io/apidocs/spot/en/#check-server-time

> **Returns** Current server time

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

> > Raises BinanceRequestException, BinanceAPIException

**get_sub_account_assets**(*\*\*params*)

**get_sub_account_futures_transfer_history**(*\*\*params*)

**get_sub_account_list**(*\*\*params*)

**get_sub_account_transfer_history**(*\*\*params*)

**get_subaccount_deposit_address**(*\*\*params*)

**get_subaccount_deposit_history**(*\*\*params*)

**get_subaccount_futures_details**(*\*\*params*)

**get_subaccount_futures_margin_status**(*\*\*params*)

**get_subaccount_futures_positionrisk**(*\*\*params*)

**get_subaccount_futures_summary**(*\*\*params*)

**get_subaccount_margin_details**(*\*\*params*)

**get_subaccount_margin_summary**(*\*\*params*)

**get_subaccount_transfer_history**(*\*\*params*)

**get_symbol_info**(*symbol*) → Optional[Dict[KT, VT]]
> 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** BinanceRequestException, BinanceAPIException

**get_symbol_ticker**(*\*\*params*)
    Latest price for a symbol or symbols.

    https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker

    **Parameters symbol** (*str*) –

    **Returns** API response

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

OR

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

**Raises** BinanceRequestException, BinanceAPIException

**get_system_status**()
    Get system status detail.

    https://binance-docs.github.io/apidocs/spot/en/#system-status-sapi-system

    **Returns** API response

```
{
    "status": 0,        # 0: normal1system maintenance
    "msg": "normal"     # normal or System maintenance.
}
```

**Raises** BinanceAPIException

**get_ticker**(*\*\*params*)
    24 hour price change statistics.

    https://binance-docs.github.io/apidocs/spot/en/#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** BinanceRequestException, BinanceAPIException

**get_trade_fee**(*\*\*params*)
> Get trade fee.
>
> https://binance-docs.github.io/apidocs/spot/en/#trade-fee-sapi-user_data
>
> **Parameters**
> - **symbol** (*str*) – optional
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns** API response

```
[
    {
        "symbol": "ADABNB",
        "makerCommission": "0.001",
        "takerCommission": "0.001"
    },
    {
        "symbol": "BNBBTC",
        "makerCommission": "0.001",
        "takerCommission": "0.001"
    }
]
```

**get_universal_transfer_history**(*\*\*params*)

**get_withdraw_history**(*\*\*params*)
    Fetch withdraw history.

    https://binance-docs.github.io/apidocs/spot/en/#withdraw-history-supporting-network-user_data

   **Parameters**

   - **coin** (`str`) – optional

   - **offset** (`int`) – optional - default:0

   - **limit** (`int`) – optional

   - **startTime** (`int`) – optional - Default: 90 days from current timestamp

   - **endTime** (`int`) – optional - Default: present timestamp

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

   **Returns** API response

```
[
    {
        "address": "0x94df8b352de7f46f64b01d3666bf6e936e44ce60",
        "amount": "8.91000000",
        "applyTime": "2019-10-12 11:12:02",
        "coin": "USDT",
        "id": "b6ae22b3aa844210a7041aee7589627c",
        "withdrawOrderId": "WITHDRAWtest123", // will not be returned if there
↪'s no withdrawOrderId for this withdraw.
        "network": "ETH",
        "transferType": 0,   // 1 for internal transfer, 0 for external␣
↪transfer
        "status": 6,
        "txId":
↪"0xb5ef8c13b968a406cc62a93a8bd80f9e9a906ef1b3fcf20a2e48573c17659268"
    },
    {
        "address": "1FZdVHtiBqMrWdjPyRPULCUceZPJ2WLCsB",
        "amount": "0.00150000",
        "applyTime": "2019-09-24 12:43:45",
        "coin": "BTC",
        "id": "156ec387f49b41df8724fa744fa82719",
        "network": "BTC",
        "status": 6,
        "txId":
↪"60fd9007ebfddc753455f95fafa808c4302c836e4d1eebc5a132c36c1d8ac354"
```

```
        }
    ]
```

>    **Raises** BinanceRequestException, BinanceAPIException

**get_withdraw_history_id**(*withdraw_id*, *\*\*params*)
>    Fetch withdraw history.

>    https://binance-docs.github.io/apidocs/spot/en/#withdraw-history-supporting-network-user_data

>    **Parameters**

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

>    **Returns** API response

```
{
    "id":"7213fea8e94b4a5593d507237e5a555b",
    "withdrawOrderId": None,
    "amount": 0.99,
    "transactionFee": 0.01,
    "address": "0x6915f16f8791d0a1cc2bf47c13a6b2a92000504b",
    "asset": "ETH",
    "txId":
→"0xdf33b22bdb2b28b1f75ccd201a4a4m6e7g83jy5fc5d5a9d1340961598cfcb0a1",
    "applyTime": 1508198532000,
    "status": 4
}
```

>    **Raises** BinanceRequestException, BinanceAPIException

**isolated_margin_stream_close**(*symbol*, *listenKey*)

**isolated_margin_stream_get_listen_key**(*symbol*)

**isolated_margin_stream_keepalive**(*symbol*, *listenKey*)

**make_subaccount_futures_transfer**(*\*\*params*)

**make_subaccount_margin_transfer**(*\*\*params*)

**make_subaccount_to_master_transfer**(*\*\*params*)

**make_subaccount_to_subaccount_transfer**(*\*\*params*)

**make_subaccount_universal_transfer**(*\*\*params*)

**make_universal_transfer**(*\*\*params*)
>    User Universal Transfer

>    https://binance-docs.github.io/apidocs/spot/en/#user-universal-transfer

>    **Parameters**

>    - **type** (*str (ENUM)*) – required

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

```
transfer_status = client.make_universal_transfer(params)
```

> **Returns** API response

```
{
    "tranId":13526853623
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**margin_stream_close**(*listenKey*)

**margin_stream_get_listen_key**()

**margin_stream_keepalive**(*listenKey*)

**new_transfer_history**(*\*\*params*)

**options_account_info**(*\*\*params*)

**options_bill**(*\*\*params*)

**options_cancel_all_orders**(*\*\*params*)

**options_cancel_batch_order**(*\*\*params*)

**options_cancel_order**(*\*\*params*)

**options_exchange_info**()

**options_funds_transfer**(*\*\*params*)

**options_historical_trades**(*\*\*params*)

**options_index_price**(*\*\*params*)

**options_info**()

**options_klines**(*\*\*params*)

**options_mark_price**(*\*\*params*)

**options_order_book**(*\*\*params*)

**options_ping**()

**options_place_batch_order**(*\*\*params*)

**options_place_order**(*\*\*params*)

**options_positions**(*\*\*params*)

**options_price**(*\*\*params*)

**options_query_order**(*\*\*params*)

**options_query_order_history**(*\*\*params*)

**options_query_pending_orders**(*\*\*params*)

**options_recent_trades**(*\*\*params*)

**options_time**()

**options_user_trades**(*\*\*params*)

**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** (*str*) – required
> - **quantity** (*decimal*) – required
> - **price** (*str*) – required
> - **timeInForce** (*str*) – 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** (*str*) – 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** BinanceRequestException, 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** (*str*) – 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** (*str*) – 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** BinanceRequestException, 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** (*str*) – 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** (*str*) – 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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_market**(*\*\*params*)
Send in a new market order

> **Parameters**
>
> - **symbol** (*str*) – required
> - **side** (*str*) – required
> - **quantity** (*decimal*) – required
> - **quoteOrderQty** (*decimal*) – amount the user wants to spend (when buying) or receive (when selling) of the quote asset
> - **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
> - **newOrderRespType** (*str*) – 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** BinanceRequestException, 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
> > - **quoteOrderQty** (*decimal*) – the amount the user wants to spend of the quote asset
> > - **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
> > - **newOrderRespType** (*str*) – 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** BinanceRequestException, 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
> > - **quoteOrderQty** (*decimal*) – the amount the user wants to receive of the quote asset
> > - **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.
> > - **newOrderRespType** (*str*) – 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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_oco_buy**(*\*\*params*)
> Send in a new OCO buy order
>
> > **Parameters**
> >
> > - **symbol** (*str*) – required
> > - **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.

- **quantity** (*decimal*) – required

- **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.

- **price** (*str*) – required

- **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.

- **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated if not sent.

- **stopPrice** (*str*) – required

- **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.

- **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.

- **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.

- **newOrderRespType** (*str*) – 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 OCO order endpoint for full response options

> **Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_oco_sell**(*\*\*params*)
Send in a new OCO sell order

> **Parameters**

- **symbol** (*str*) – required

- **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.

- **quantity** (*decimal*) – required

- **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.

- **price** (*str*) – required

- **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.

- **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated if not sent.

- **stopPrice** (*str*) – required

- **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.

- **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.

- **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.

- **newOrderRespType** (*str*) – 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 OCO order endpoint for full response options

**Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**ping**() → Dict[KT, VT]
Test connectivity to the Rest API.

https://binance-docs.github.io/apidocs/spot/en/#test-connectivity

    **Returns** Empty array

```
{}
```

    **Raises** BinanceRequestException, BinanceAPIException

**purchase_lending_product**(*\*\*params*)

**query_subaccount_spot_summary**(*\*\*params*)

**query_universal_transfer_history**(*\*\*params*)
Query User Universal Transfer History

https://binance-docs.github.io/apidocs/spot/en/#query-user-universal-transfer-history

    **Parameters**

- **type** (*str (ENUM)*) – required

- **startTime** (*int*) – optional

- **endTime** (*int*) – optional

- **current** (*int*) – optional - Default 1

- **size** (*int*) – required - Default 10, Max 100

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

```
transfer_status = client.query_universal_transfer_history(params)
```

    **Returns** API response

```
{
    "total":2,
    "rows":[
        {
            "asset":"USDT",
            "amount":"1",
            "type":"MAIN_UMFUTURE"
            "status": "CONFIRMED",
            "tranId": 11415955596,
            "timestamp":1544433328000
        },
```

```
        {
            "asset":"USDT",
            "amount":"2",
            "type":"MAIN_UMFUTURE",
            "status": "CONFIRMED",
            "tranId": 11366865406,
            "timestamp":1544433328000
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**redeem_lending_product**(*\*\*params*)

**repay_margin_loan**(*\*\*params*)

**stream_close**(*listenKey*)
    Close out a user data stream.

    https://binance-docs.github.io/apidocs/spot/en/#listen-key-spot

> **Parameters** **listenKey** (*str*) – required

> **Returns** API response

```
{}
```

> **Raises** BinanceRequestException, 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://binance-docs.github.io/apidocs/spot/en/#listen-key-spot

> **Returns** API response

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

> **Raises** BinanceRequestException, BinanceAPIException

**stream_keepalive**(*listenKey*)
    PING a user data stream to prevent a time out.

    https://binance-docs.github.io/apidocs/spot/en/#listen-key-spot

> **Parameters** **listenKey** (*str*) – required

> **Returns** API response

```
{}
```

> **Raises** BinanceRequestException, BinanceAPIException

**toggle_bnb_burn_spot_margin**(*\*\*params*)

**transfer_dust**(*\*\*params*)
> Convert dust assets to BNB.
>
> https://binance-docs.github.io/apidocs/spot/en/#dust-transfer-user_data
>
> > **Parameters**
> >
> > - **asset** (*str*) – The asset being converted. e.g: 'ONE'
> >
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for

```
result = client.transfer_dust(asset='ONE')
```

> > **Returns** API response

```
{
    "totalServiceCharge":"0.02102542",
    "totalTransfered":"1.05127099",
    "transferResult":[
        {
            "amount":"0.03000000",
            "fromAsset":"ETH",
            "operateTime":1563368549307,
            "serviceChargeAmount":"0.00500000",
            "tranId":2970932918,
            "transferedAmount":"0.25000000"
        }
    ]
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**transfer_history**(*\*\*params*)

**transfer_isolated_margin_to_spot**(*\*\*params*)

**transfer_margin_to_spot**(*\*\*params*)

**transfer_spot_to_isolated_margin**(*\*\*params*)

**transfer_spot_to_margin**(*\*\*params*)

**universal_transfer**(*\*\*params*)

**withdraw**(*\*\*params*)
> Submit a withdraw request.
>
> https://binance-docs.github.io/apidocs/spot/en/#withdraw-sapi
>
> 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**

- **coin** (*str*) – required

- **withdrawOrderId** (*str*) – optional - client id for withdraw

- **network** (*str*) – optional

- **address** (*str*) – optional

- **amount** (*decimal*) – required

- **transactionFeeFlag** (*bool*) – required - When making internal transfer, true for
  returning the fee to the destination account; false for returning the fee back to the departure
  account. Default false.

- **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

```
{
    "id":"7213fea8e94b4a5593d507237e5a555b"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**class** binance.client.**BaseClient**(*api_key: Optional[str] = None, api_secret: Optional[str] = None, requests_params: Dict[str, str] = None, tld: str = 'com', testnet: bool = False*)

Bases: `object`

**AGG_BEST_MATCH = 'M'**

**AGG_BUYER_MAKES = 'm'**

**AGG_FIRST_TRADE_ID = 'f'**

**AGG_ID = 'a'**

**AGG_LAST_TRADE_ID = 'l'**

**AGG_PRICE = 'p'**

**AGG_QUANTITY = 'q'**

**AGG_TIME = 'T'**

**API_TESTNET_URL = 'https://testnet.binance.vision/api'**

**API_URL = 'https://api.binance.{}/api'**

**COIN_FUTURE_TO_SPOT = 'CMFUTURE_MAIN'**

**FIAT_TO_MINING = 'C2C_MINING'**

**FIAT_TO_SPOT = 'C2C_MAIN'**

**FIAT_TO_USDT_FUTURE = 'C2C_UMFUTURE'**

**FUTURES_API_VERSION = 'v1'**

**FUTURES_API_VERSION2 = 'v2'**

**FUTURES_COIN_DATA_TESTNET_URL = 'https://testnet.binancefuture.com/futures/data'**

**FUTURES_COIN_DATA_URL = 'https://dapi.binance.{}/futures/data'**

```
FUTURES_COIN_TESTNET_URL = 'https://testnet.binancefuture.com/dapi'

FUTURES_COIN_URL = 'https://dapi.binance.{}/dapi'

FUTURES_DATA_TESTNET_URL = 'https://testnet.binancefuture.com/futures/data'

FUTURES_DATA_URL = 'https://fapi.binance.{}/futures/data'

FUTURES_TESTNET_URL = 'https://testnet.binancefuture.com/fapi'

FUTURES_URL = 'https://fapi.binance.{}/fapi'

FUTURE_ORDER_TYPE_LIMIT = 'LIMIT'

FUTURE_ORDER_TYPE_LIMIT_MAKER = 'LIMIT_MAKER'

FUTURE_ORDER_TYPE_MARKET = 'MARKET'

FUTURE_ORDER_TYPE_STOP = 'STOP'

FUTURE_ORDER_TYPE_STOP_MARKET = 'STOP_MARKET'

FUTURE_ORDER_TYPE_TAKE_PROFIT = 'TAKE_PROFIT'

FUTURE_ORDER_TYPE_TAKE_PROFIT_MARKET = 'TAKE_PROFIT_MARKET'

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'

MARGIN_API_URL = 'https://api.binance.{}/sapi'

MARGIN_API_VERSION = 'v1'

MARGIN_CROSS_TO_SPOT = 'MARGIN_MAIN'

MARGIN_CROSS_TO_USDT_FUTURE = 'MARGIN_UMFUTURE'

MINING_TO_FIAT = 'MINING_C2C'

MINING_TO_SPOT = 'MINING_MAIN'

MINING_TO_USDT_FUTURE = 'MINING_UMFUTURE'

OPTIONS_API_VERSION = 'v1'
```

```
OPTIONS_TESTNET_URL = 'https://testnet.binanceops.{}/vapi'

OPTIONS_URL = 'https://vapi.binance.{}/vapi'

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'

REQUEST_TIMEOUT = 10

SIDE_BUY = 'BUY'

SIDE_SELL = 'SELL'

SPOT_TO_COIN_FUTURE = 'MAIN_CMFUTURE'

SPOT_TO_FIAT = 'MAIN_C2C'

SPOT_TO_MARGIN_CROSS = 'MAIN_MARGIN'

SPOT_TO_MINING = 'MAIN_MINING'

SPOT_TO_USDT_FUTURE = 'MAIN_UMFUTURE'

SYMBOL_TYPE_SPOT = 'SPOT'

TIME_IN_FORCE_FOK = 'FOK'

TIME_IN_FORCE_GTC = 'GTC'

TIME_IN_FORCE_IOC = 'IOC'

USDT_FUTURE_TO_FIAT = 'UMFUTURE_C2C'

USDT_FUTURE_TO_MARGIN_CROSS = 'UMFUTURE_MARGIN'

USDT_FUTURE_TO_SPOT = 'UMFUTURE_MAIN'
```

```
WEBSITE_URL = 'https://www.binance.{}'
```

**__init__**(*api_key: Optional[str] = None*, *api_secret: Optional[str] = None*, *requests_params: Dict[str, str] = None*, *tld: str = 'com'*, *testnet: bool = False*)
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
>
>    - **testnet** (*bool*) – Use testnet environment - only available for vanilla options at the moment

**class** binance.client.**Client**(*api_key: Optional[str] = None*, *api_secret: Optional[str] = None*, *requests_params: Dict[str, str] = None*, *tld: str = 'com'*, *testnet: bool = False*)
Bases: *binance.client.BaseClient*

**__init__**(*api_key: Optional[str] = None*, *api_secret: Optional[str] = None*, *requests_params: Dict[str, str] = None*, *tld: str = 'com'*, *testnet: bool = False*)
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
>
>    - **testnet** (*bool*) – Use testnet environment - only available for vanilla options at the moment

**aggregate_trade_iter**(*symbol: str*, *start_str=None*, *last_id=None*)
Iterate over aggregate trade data from (start_time or last_id) to the end of the history so far.

If start_time is specified, start with the first trade after start_time. Meant to initialise a local cache of trade data.

If last_id is specified, start with the trade after it. This is meant for updating a pre-existing local trade data cache.

Only allows start_str or last_id—not both. Not guaranteed to work right if you're running more than one of these simultaneously. You will probably hit your rate limit.

See dateparser 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*) – Symbol string e.g. ETHBTC
>
>    - **start_str** – Start date string in UTC format or timestamp in milliseconds. The iterator will

return the first trade occurring later than this time. :type start_str: str|int :param last_id: aggregate trade ID of the last known aggregate trade. Not a regular trade ID. See https://binance-docs.github.io/apidocs/spot/en/#compressed-aggregate-trades-list

>    **Returns** an iterator of JSON objects, one per trade. The format of

each object is identical to Client.aggregate_trades().

**cancel_margin_oco_order** (*\*\*params*)
  Cancel an entire Order List for a margin account.

  https://binance-docs.github.io/apidocs/spot/en/#margin-account-cancel-oco-trade

  **Parameters**

  - **symbol** (*str*) – required

  - **isIsolated** – for isolated margin or not, "TRUE", "FALSE"default "FALSE"

  - **orderListId** (*int*) – Either orderListId or listClientOrderId must be provided

  - **listClientOrderId** (*str*) – Either orderListId or listClientOrderId must be provided

  - **newClientOrderId** (*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

```
{
    "orderListId": 0,
    "contingencyType": "OCO",
    "listStatusType": "ALL_DONE",
    "listOrderStatus": "ALL_DONE",
    "listClientOrderId": "C3wyj4WVEktd7u9aVBRXcN",
    "transactionTime": 1574040868128,
    "symbol": "LTCBTC",
    "isIsolated": false,       // if isolated margin
    "orders": [
        {
            "symbol": "LTCBTC",
            "orderId": 2,
            "clientOrderId": "pO9ufTiFGg3nw2fOdgeOXa"
        },
        {
            "symbol": "LTCBTC",
            "orderId": 3,
            "clientOrderId": "TXOvglzXuaubXAaENpaRCB"
        }
    ],
    "orderReports": [
        {
            "symbol": "LTCBTC",
            "origClientOrderId": "pO9ufTiFGg3nw2fOdgeOXa",
            "orderId": 2,
            "orderListId": 0,
            "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
            "price": "1.00000000",
            "origQty": "10.00000000",
            "executedQty": "0.00000000",
            "cummulativeQuoteQty": "0.00000000",
            "status": "CANCELED",
            "timeInForce": "GTC",
            "type": "STOP_LOSS_LIMIT",
```

```
            "side": "SELL",
            "stopPrice": "1.00000000"
        },
        {
            "symbol": "LTCBTC",
            "origClientOrderId": "TXOvglzXuaubXAaENpaRCB",
            "orderId": 3,
            "orderListId": 0,
            "clientOrderId": "unfWT8ig8i0uj6lPuYLez6",
            "price": "3.00000000",
            "origQty": "10.00000000",
            "executedQty": "0.00000000",
            "cummulativeQuoteQty": "0.00000000",
            "status": "CANCELED",
            "timeInForce": "GTC",
            "type": "LIMIT_MAKER",
            "side": "SELL"
        }
    ]
}
```

**cancel_margin_order**(*\*\*params*)

Cancel an active order for margin account.

Either orderId or origClientOrderId must be sent.

https://binance-docs.github.io/apidocs/spot/en/#margin-account-cancel-order-trade

> **Parameters**
>
> - **symbol** (*str*) – required
>
> - **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')
>
> - **orderId** (*str*) –
>
> - **origClientOrderId** (*str*) –
>
> - **newClientOrderId** (*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", "orderId": 28, "origClientOrderId": "myOrder1", "clientOrderId": "cancelMyOrder1", "transactTime": 1507725176595, "price": "1.00000000", "origQty": "10.00000000", "executedQty": "8.00000000", "cummulativeQuoteQty": "8.00000000", "status": "CANCELED", "timeInForce": "GTC", "type": "LIMIT", "side": "SELL"
>
> **}**
>
> **Raises** BinanceRequestException, BinanceAPIException

**cancel_order**(*\*\*params*)

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

https://binance-docs.github.io/apidocs/spot/en/#cancel-order-trade

> **Parameters**

---

**6.1. Contents**

- **symbol** (*str*) – required

- **orderId** (*int*) – The unique order id

- **origClientOrderId** (*str*) – optional

- **newClientOrderId** (*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",
    "origClientOrderId": "myOrder1",
    "orderId": 1,
    "clientOrderId": "cancelMyOrder1"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**change_fixed_activity_to_daily_position** (*\*\*params*)
Change Fixed/Activity Position to Daily Position

https://binance-docs.github.io/apidocs/spot/en/#change-fixed-activity-position-to-daily-position-user_data

**close_connection** ()

**create_isolated_margin_account** (*\*\*params*)
Create isolated margin account for symbol

https://binance-docs.github.io/apidocs/spot/en/#create-isolated-margin-account-margin

**Parameters**

- **base** (*str*) – Base asset of symbol

- **quote** (*str*) – Quote asset of symbol

```
pair_details = client.create_isolated_margin_account(base='USDT', quote='BTC')
```

**Returns** API response

```
{
    "success": true,
    "symbol": "BTCUSDT"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**create_margin_loan** (*\*\*params*)
Apply for a loan in cross-margin or isolated-margin account.

https://binance-docs.github.io/apidocs/spot/en/#margin-account-borrow-margin

**Parameters**

- **asset** (*str*) – name of the asset

- **amount** (*str*) – amount to transfer

- **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')

- **symbol** (*str*) – Isolated margin symbol (default blank for cross-margin)

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

```
transaction = client.margin_create_loan(asset='BTC', amount='1.1')

transaction = client.margin_create_loan(asset='BTC', amount='1.1',
                                        isIsolated='TRUE', symbol='ETHBTC')
```

**Returns** API response

```
{
    "tranId": 100000001
}
```

**Raises** BinanceRequestException, BinanceAPIException

**create_margin_oco_order**(*\*\*params*)
    Post a new OCO trade for margin account.

    https://binance-docs.github.io/apidocs/spot/en/#margin-account-new-oco-trade

    **Parameters**

- **symbol** (*str*) – required

- **isIsolated** – for isolated margin or not, "TRUE", "FALSE"default "FALSE"

- **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.

- **side** (*str*) – required

- **quantity** (*decimal*) – required

- **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.

- **price** (*str*) – required

- **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.

- **stopClientOrderId** (*str*) – A unique Id for the stop loss/stop loss limit leg. Automatically generated if not sent.

- **stopPrice** (*str*) – required

- **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.

- **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.

- **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.

- **newOrderRespType** (*str*) – Set the response JSON. ACK, RESULT, or FULL; default: RESULT.

- **sideEffectType** (*str*) – NO_SIDE_EFFECT, MARGIN_BUY, AUTO_REPAY; default NO_SIDE_EFFECT.

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

Returns  API response

```
{
    "orderListId": 0,
    "contingencyType": "OCO",
    "listStatusType": "EXEC_STARTED",
    "listOrderStatus": "EXECUTING",
    "listClientOrderId": "JYVpp3F0f5CAG15DhtrqLp",
    "transactionTime": 1563417480525,
    "symbol": "LTCBTC",
    "marginBuyBorrowAmount": "5",       // will not return if no margin trade
→happens
    "marginBuyBorrowAsset": "BTC",     // will not return if no margin trade
→happens
    "isIsolated": false,        // if isolated margin
    "orders": [
        {
            "symbol": "LTCBTC",
            "orderId": 2,
            "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos"
        },
        {
            "symbol": "LTCBTC",
            "orderId": 3,
            "clientOrderId": "xTXKaGYd4bluPVp78IVRvl"
        }
    ],
    "orderReports": [
        {
            "symbol": "LTCBTC",
            "orderId": 2,
            "orderListId": 0,
            "clientOrderId": "Kk7sqHb9J6mJWTMDVW7Vos",
            "transactTime": 1563417480525,
            "price": "0.000000",
            "origQty": "0.624363",
            "executedQty": "0.000000",
            "cummulativeQuoteQty": "0.000000",
            "status": "NEW",
            "timeInForce": "GTC",
            "type": "STOP_LOSS",
            "side": "BUY",
            "stopPrice": "0.960664"
        },
        {
            "symbol": "LTCBTC",
            "orderId": 3,
            "orderListId": 0,
            "clientOrderId": "xTXKaGYd4bluPVp78IVRvl",
            "transactTime": 1563417480525,
            "price": "0.036435",
            "origQty": "0.624363",
            "executedQty": "0.000000",
            "cummulativeQuoteQty": "0.000000",
```

(continues on next page)

```
            "status": "NEW",
            "timeInForce": "GTC",
            "type": "LIMIT_MAKER",
            "side": "BUY"
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_margin_order** (*\*\*params*)

Post a new order for margin account.

https://binance-docs.github.io/apidocs/spot/en/#margin-account-new-order-trade

**Parameters**

- **symbol** (`str`) – required
- **isIsolated** (`str`) – set to 'TRUE' for isolated margin (default 'FALSE')
- **side** (`str`) – required
- **type** (`str`) – required
- **quantity** (`decimal`) – required
- **price** (`str`) – required
- **stopPrice** (`str`) – Used with STOP_LOSS, STOP_LOSS_LIMIT, TAKE_PROFIT, and TAKE_PROFIT_LIMIT orders.
- **timeInForce** (`str`) – required if limit order GTC,IOC,FOK
- **newClientOrderId** (`str`) – A unique id for the order. Automatically generated if not sent.
- **icebergQty** (`str`) – Used with LIMIT, STOP_LOSS_LIMIT, and TAKE_PROFIT_LIMIT to create an iceberg order.
- **newOrderRespType** (`str`) – Set the response JSON. ACK, RESULT, or FULL; MARKET and LIMIT order types default to FULL, all other orders default to ACK.
- **recvWindow** (`int`) – the number of milliseconds the request is valid for

**Returns** API response

Response ACK:

```
{
    "symbol": "BTCUSDT",
    "orderId": 28,
    "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
    "transactTime": 1507725176595
}
```

Response RESULT:

```
{
    "symbol": "BTCUSDT",
    "orderId": 28,
    "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
    "transactTime": 1507725176595,
    "price": "1.00000000",
    "origQty": "10.00000000",
    "executedQty": "10.00000000",
    "cummulativeQuoteQty": "10.00000000",
    "status": "FILLED",
    "timeInForce": "GTC",
    "type": "MARKET",
    "side": "SELL"
}
```

Response FULL:

```
{
    "symbol": "BTCUSDT",
    "orderId": 28,
    "clientOrderId": "6gCrw2kRUAF9CvJDGP16IP",
    "transactTime": 1507725176595,
    "price": "1.00000000",
    "origQty": "10.00000000",
    "executedQty": "10.00000000",
    "cummulativeQuoteQty": "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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOr-
> derMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalExcep-
> tion, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_oco_order**(*\*\*params*)

Send in a new OCO order

https://binance-docs.github.io/apidocs/spot/en/#new-oco-trade

> **Parameters**
>
> - **symbol** (*str*) – required
>
> - **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated
>   if not sent.
>
> - **side** (*str*) – required
>
> - **quantity** (*decimal*) – required
>
> - **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically gener-
>   ated if not sent.
>
> - **price** (*str*) – required
>
> - **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg
>   order.
>
> - **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated
>   if not sent.
>
> - **stopPrice** (*str*) – required
>
> - **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.
>
> - **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an ice-
>   berg order.
>
> - **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.
>
> - **newOrderRespType** (*str*) – Set the response JSON. ACK, RESULT, or FULL; de-
>   fault: RESULT.
>
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns** API response

Response ACK:

```
{
}
```

Response RESULT:

```
{
}
```

Response FULL:

```
{
}
```

> **Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_order**(*\*\*params*)

Send in a new order

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

https://binance-docs.github.io/apidocs/spot/en/#new-order-trade

> **Parameters**
>
> - **symbol** (*str*) – required
> - **side** (*str*) – required
> - **type** (*str*) – required
> - **timeInForce** (*str*) – required if limit order
> - **quantity** (*decimal*) – required
> - **quoteOrderQty** (*decimal*) – amount the user wants to spend (when buying) or receive (when selling) of the quote asset, applicable to MARKET orders
> - **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** (*str*) – 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",
    "cummulativeQuoteQty": "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",
    "cummulativeQuoteQty": "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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**create_sub_account_futures_transfer**(*\*\*params*)
    Execute sub-account Futures transfer

    https://github.com/binance-exchange/binance-official-api-docs/blob/9dbe0e961b80557bb19708a707c7fad08842b28e/wapi-api.md#sub-account-transferfor-master-account

    **Parameters**

    - **fromEmail** (*str*) – required - Sender email
    - **toEmail** (*str*) – required - Recipient email
    - **futuresType** (*int*) – required
    - **asset** (*str*) – required
    - **amount** (*decimal*) – required
    - **recvWindow** (*int*) – optional

    **Returns** API response

```
{
    "success":true,
    "txnId":"2934662589"
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**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://binance-docs.github.io/apidocs/spot/en/#test-new-order-trade

    **Parameters**

    - **symbol** (*str*) – required
    - **side** (*str*) – required
    - **type** (*str*) – required
    - **timeInForce** (*str*) – 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** (*str*) – 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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**disable_fast_withdraw_switch**(*\*\*params*)
  Disable Fast Withdraw Switch

  https://binance-docs.github.io/apidocs/spot/en/#disable-fast-withdraw-switch-user_data

  **Parameters recvWindow** (*int*) – optional

  **Returns** API response

  **Raises** BinanceRequestException, BinanceAPIException

**disable_isolated_margin_account**(*\*\*params*)
  Disable isolated margin account for a specific symbol. Each trading pair can only be deactivated once every 24 hours.

  https://binance-docs.github.io/apidocs/spot/en/#disable-isolated-margin-account-trade

  **Parameters symbol** –

  **Returns** API response

```
{
  "success": true,
  "symbol": "BTCUSDT"
}
```

**enable_fast_withdraw_switch**(*\*\*params*)
  Enable Fast Withdraw Switch

  https://binance-docs.github.io/apidocs/spot/en/#enable-fast-withdraw-switch-user_data

  **Parameters recvWindow** (*int*) – optional

  **Returns** API response

  **Raises** BinanceRequestException, BinanceAPIException

**enable_isolated_margin_account**(*\*\*params*)
  Enable isolated margin account for a specific symbol.

  https://binance-docs.github.io/apidocs/spot/en/#enable-isolated-margin-account-trade

  **Parameters symbol** –

  **Returns** API response

```
{
  "success": true,
  "symbol": "BTCUSDT"
}
```

**enable_subaccount_futures**(*\*\*params*)
　　Enable Futures for Sub-account (For Master Account)

　　https://binance-docs.github.io/apidocs/spot/en/#enable-futures-for-sub-account-for-master-account

　　　　**Parameters**

　　　　　　• **email** (*str*) – required - Sub account email

　　　　　　• **recvWindow** (*int*) – optional

　　　　**Returns** API response

```
{

    "email":"123@test.com",

    "isFuturesEnabled": true  // true or false

}
```

　　　　**Raises** BinanceRequestException, BinanceAPIException

**enable_subaccount_margin**(*\*\*params*)
　　Enable Margin for Sub-account (For Master Account)

　　https://binance-docs.github.io/apidocs/spot/en/#enable-margin-for-sub-account-for-master-account

　　　　**Parameters**

　　　　　　• **email** (*str*) – required - Sub account email

　　　　　　• **recvWindow** (*int*) – optional

　　　　**Returns** API response

```
{

    "email":"123@test.com",

    "isMarginEnabled": true

}
```

　　　　**Raises** BinanceRequestException, BinanceAPIException

**futures_account**(*\*\*params*)
　　Get current account information.

　　https://binance-docs.github.io/apidocs/futures/en/#account-information-user_data

**futures_account_balance**(*\*\*params*)
　　Get futures account balance

　　https://binance-docs.github.io/apidocs/futures/en/#future-account-balance-user_data

**futures_account_trades**(*\*\*params*)
Get trades for the authenticated account and symbol.

https://binance-docs.github.io/apidocs/futures/en/#account-trade-list-user_data

**futures_account_transfer**(*\*\*params*)
Execute transfer between spot account and futures account.

https://binance-docs.github.io/apidocs/futures/en/#new-future-account-transfer

**futures_adl_quantile_estimate**(*\*\*params*)
Get Position ADL Quantile Estimate

https://binance-docs.github.io/apidocs/futures/en/#position-adl-quantile-estimation-user_data

**futures_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://binance-docs.github.io/apidocs/futures/en/#compressed-aggregate-trades-list-market_data

**futures_cancel_all_open_orders**(*\*\*params*)
Cancel all open futures orders

https://binance-docs.github.io/apidocs/futures/en/#cancel-all-open-orders-trade

**futures_cancel_order**(*\*\*params*)
Cancel an active futures order.

https://binance-docs.github.io/apidocs/futures/en/#cancel-order-trade

**futures_cancel_orders**(*\*\*params*)
Cancel multiple futures orders

https://binance-docs.github.io/apidocs/futures/en/#cancel-multiple-orders-trade

**futures_change_leverage**(*\*\*params*)
Change user's initial leverage of specific symbol market

https://binance-docs.github.io/apidocs/futures/en/#change-initial-leverage-trade

**futures_change_margin_type**(*\*\*params*)
Change the margin type for a symbol

https://binance-docs.github.io/apidocs/futures/en/#change-margin-type-trade

**futures_change_multi_assets_mode**(*multiAssetsMargin: bool*)
Change user's Multi-Assets mode (Multi-Assets Mode or Single-Asset Mode) on Every symbol

https://binance-docs.github.io/apidocs/futures/en/#change-multi-assets-mode-trade

**futures_change_position_margin**(*\*\*params*)
Change the position margin for a symbol

https://binance-docs.github.io/apidocs/futures/en/#modify-isolated-position-margin-trade

**futures_change_position_mode**(*\*\*params*)
Change position mode for authenticated account

https://binance-docs.github.io/apidocs/futures/en/#change-position-mode-trade

**futures_coin_account**(*\*\*params*)
Get current account information.

https://binance-docs.github.io/apidocs/delivery/en/#account-information-user_data

**futures_coin_account_balance**(*\*\*params*)
　　Get futures account balance

　　https://binance-docs.github.io/apidocs/delivery/en/#futures-account-balance-user_data

**futures_coin_account_trades**(*\*\*params*)
　　Get trades for the authenticated account and symbol.

　　https://binance-docs.github.io/apidocs/delivery/en/#account-trade-list-user_data

**futures_coin_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://binance-docs.github.io/apidocs/delivery/en/#compressed-aggregate-trades-list

**futures_coin_cancel_all_open_orders**(*\*\*params*)
　　Cancel all open futures orders

　　https://binance-docs.github.io/apidocs/delivery/en/#cancel-all-open-orders-trade

**futures_coin_cancel_order**(*\*\*params*)
　　Cancel an active futures order.

　　https://binance-docs.github.io/apidocs/delivery/en/#cancel-order-trade

**futures_coin_cancel_orders**(*\*\*params*)
　　Cancel multiple futures orders

　　https://binance-docs.github.io/apidocs/delivery/en/#cancel-multiple-orders-trade

**futures_coin_change_leverage**(*\*\*params*)
　　Change user's initial leverage of specific symbol market

　　https://binance-docs.github.io/apidocs/delivery/en/#change-initial-leverage-trade

**futures_coin_change_margin_type**(*\*\*params*)
　　Change the margin type for a symbol

　　https://binance-docs.github.io/apidocs/delivery/en/#change-margin-type-trade

**futures_coin_change_position_margin**(*\*\*params*)
　　Change the position margin for a symbol

　　https://binance-docs.github.io/apidocs/delivery/en/#modify-isolated-position-margin-trade

**futures_coin_change_position_mode**(*\*\*params*)
　　Change user's position mode (Hedge Mode or One-way Mode ) on EVERY symbol

　　https://binance-docs.github.io/apidocs/delivery/en/#change-position-mode-trade

**futures_coin_continous_klines**(*\*\*params*)
　　Kline/candlestick bars for a specific contract type. Klines are uniquely identified by their open time.

　　https://binance-docs.github.io/apidocs/delivery/en/#continuous-contract-kline-candlestick-data

**futures_coin_create_order**(*\*\*params*)
　　Send in a new order.

　　https://binance-docs.github.io/apidocs/delivery/en/#new-order-trade

**futures_coin_exchange_info**()
　　Current exchange trading rules and symbol information

　　https://binance-docs.github.io/apidocs/delivery/en/#exchange-information

**futures_coin_funding_rate**(*\*\*params*)
> Get funding rate history
>
> https://binance-docs.github.io/apidocs/delivery/en/#get-funding-rate-history-of-perpetual-futures

**futures_coin_get_all_orders**(*\*\*params*)
> Get all futures account orders; active, canceled, or filled.
>
> https://binance-docs.github.io/apidocs/delivery/en/#all-orders-user_data

**futures_coin_get_open_orders**(*\*\*params*)
> Get all open orders on a symbol.
>
> https://binance-docs.github.io/apidocs/delivery/en/#current-all-open-orders-user_data

**futures_coin_get_order**(*\*\*params*)
> Check an order's status.
>
> https://binance-docs.github.io/apidocs/delivery/en/#query-order-user_data

**futures_coin_get_position_mode**(*\*\*params*)
> Get user's position mode (Hedge Mode or One-way Mode ) on EVERY symbol
>
> https://binance-docs.github.io/apidocs/delivery/en/#get-current-position-mode-user_data

**futures_coin_historical_trades**(*\*\*params*)
> Get older market historical trades.
>
> https://binance-docs.github.io/apidocs/delivery/en/#old-trades-lookup-market_data

**futures_coin_income_history**(*\*\*params*)
> Get income history for authenticated account
>
> https://binance-docs.github.io/apidocs/delivery/en/#get-income-history-user_data

**futures_coin_index_price_klines**(*\*\*params*)
> Kline/candlestick bars for the index price of a pair..
>
> https://binance-docs.github.io/apidocs/delivery/en/#index-price-kline-candlestick-data

**futures_coin_klines**(*\*\*params*)
> Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.
>
> https://binance-docs.github.io/apidocs/delivery/en/#kline-candlestick-data

**futures_coin_leverage_bracket**(*\*\*params*)
> Notional and Leverage Brackets
>
> https://binance-docs.github.io/apidocs/delivery/en/#notional-bracket-for-pair-user_data

**futures_coin_liquidation_orders**(*\*\*params*)
> Get all liquidation orders
>
> https://binance-docs.github.io/apidocs/delivery/en/#user-39-s-force-orders-user_data

**futures_coin_mark_price**(*\*\*params*)
> Get Mark Price and Funding Rate
>
> https://binance-docs.github.io/apidocs/delivery/en/#index-price-and-mark-price

**futures_coin_mark_price_klines**(*\*\*params*)
> Kline/candlestick bars for the index price of a pair..
>
> https://binance-docs.github.io/apidocs/delivery/en/#mark-price-kline-candlestick-data

**futures_coin_open_interest**(*\*\*params*)
Get present open interest of a specific symbol.

https://binance-docs.github.io/apidocs/delivery/en/#open-interest

**futures_coin_open_interest_hist**(*\*\*params*)
Get open interest statistics of a specific symbol.

https://binance-docs.github.io/apidocs/delivery/en/#open-interest-statistics-market-data

**futures_coin_order_book**(*\*\*params*)
Get the Order Book for the market

https://binance-docs.github.io/apidocs/delivery/en/#order-book

**futures_coin_orderbook_ticker**(*\*\*params*)
Best price/qty on the order book for a symbol or symbols.

https://binance-docs.github.io/apidocs/delivery/en/#symbol-order-book-ticker

**futures_coin_ping**()
Test connectivity to the Rest API

https://binance-docs.github.io/apidocs/delivery/en/#test-connectivity

**futures_coin_place_batch_order**(*\*\*params*)
Send in new orders.

https://binance-docs.github.io/apidocs/delivery/en/#place-multiple-orders-trade

To avoid modifying the existing signature generation and parameter order logic, the url encoding is done on the special query param, batchOrders, in the early stage.

**futures_coin_position_information**(*\*\*params*)
Get position information

https://binance-docs.github.io/apidocs/delivery/en/#position-information-user_data

**futures_coin_position_margin_history**(*\*\*params*)
Get position margin change history

https://binance-docs.github.io/apidocs/delivery/en/#get-position-margin-change-history-trade

**futures_coin_recent_trades**(*\*\*params*)
Get recent trades (up to last 500).

https://binance-docs.github.io/apidocs/delivery/en/#recent-trades-list

**futures_coin_stream_close**(*listenKey*)

**futures_coin_stream_get_listen_key**()

**futures_coin_stream_keepalive**(*listenKey*)

**futures_coin_symbol_ticker**(*\*\*params*)
Latest price for a symbol or symbols.

https://binance-docs.github.io/apidocs/delivery/en/#symbol-price-ticker

**futures_coin_ticker**(*\*\*params*)
24 hour rolling window price change statistics.

https://binance-docs.github.io/apidocs/delivery/en/#24hr-ticker-price-change-statistics

**futures_coin_time**()
Test connectivity to the Rest API and get the current server time.

https://binance-docs.github.io/apidocs/delivery/en/#check-server-time

**futures_continous_klines**(*\*\*params*)

Kline/candlestick bars for a specific contract type. Klines are uniquely identified by their open time.

https://binance-docs.github.io/apidocs/futures/en/#continuous-contract-kline-candlestick-data

**futures_create_order**(*\*\*params*)

Send in a new order.

https://binance-docs.github.io/apidocs/futures/en/#new-order-trade

**futures_exchange_info**()

Current exchange trading rules and symbol information

https://binance-docs.github.io/apidocs/futures/en/#exchange-information-market_data

**futures_funding_rate**(*\*\*params*)

Get funding rate history

https://binance-docs.github.io/apidocs/futures/en/#get-funding-rate-history-market_data

**futures_get_all_orders**(*\*\*params*)

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

https://binance-docs.github.io/apidocs/futures/en/#all-orders-user_data

**futures_get_multi_assets_mode**()

Get user's Multi-Assets mode (Multi-Assets Mode or Single-Asset Mode) on Every symbol

https://binance-docs.github.io/apidocs/futures/en/#get-current-multi-assets-mode-user_data

**futures_get_open_orders**(*\*\*params*)

Get all open orders on a symbol.

https://binance-docs.github.io/apidocs/futures/en/#current-open-orders-user_data

**futures_get_order**(*\*\*params*)

Check an order's status.

https://binance-docs.github.io/apidocs/futures/en/#query-order-user_data

**futures_get_position_mode**(*\*\*params*)

Get position mode for authenticated account

https://binance-docs.github.io/apidocs/futures/en/#get-current-position-mode-user_data

**futures_global_longshort_ratio**(*\*\*params*)

Get present global long to short ratio of a specific symbol.

https://binance-docs.github.io/apidocs/futures/en/#long-short-ratio

**futures_historical_klines**(*symbol*, *interval*, *start_str*, *end_str=None*, *limit=500*)

Get historical futures klines from Binance

> **Parameters**
>
> - **symbol** (*str*) – Name of symbol pair e.g BNBBTC
>
> - **interval** (*str*) – Binance Kline interval
>
> - **start_str** (*str/int*) – Start date string in UTC format or timestamp in milliseconds
>
> - **end_str** (*str/int*) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)
>
> - **limit** (*int*) – Default 500; max 1000.

> **Returns** list of OHLCV values

**futures_historical_klines_generator**(*symbol*, *interval*, *start_str*, *end_str=None*)
    Get historical futures klines generator from Binance

> **Parameters**
>
>   - **symbol** (`str`) – Name of symbol pair e.g BNBBTC
>
>   - **interval** (`str`) – Binance Kline interval
>
>   - **start_str** (`str`/`int`) – Start date string in UTC format or timestamp in milliseconds
>
>   - **end_str** (`str`/`int`) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)
>
> **Returns** generator of OHLCV values

**futures_historical_trades**(*\*\*params*)
    Get older market historical trades.

    https://binance-docs.github.io/apidocs/futures/en/#old-trades-lookup-market_data

**futures_income_history**(*\*\*params*)
    Get income history for authenticated account

    https://binance-docs.github.io/apidocs/futures/en/#get-income-history-user_data

**futures_klines**(*\*\*params*)
    Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.

    https://binance-docs.github.io/apidocs/futures/en/#kline-candlestick-data-market_data

**futures_leverage_bracket**(*\*\*params*)
    Notional and Leverage Brackets

    https://binance-docs.github.io/apidocs/futures/en/#notional-and-leverage-brackets-market_data

**futures_liquidation_orders**(*\*\*params*)
    Get all liquidation orders

    https://binance-docs.github.io/apidocs/futures/en/#get-all-liquidation-orders-market_data

**futures_mark_price**(*\*\*params*)
    Get Mark Price and Funding Rate

    https://binance-docs.github.io/apidocs/futures/en/#mark-price-market_data

**futures_open_interest**(*\*\*params*)
    Get present open interest of a specific symbol.

    https://binance-docs.github.io/apidocs/futures/en/#open-interest

**futures_open_interest_hist**(*\*\*params*)
    Get open interest statistics of a specific symbol.

    https://binance-docs.github.io/apidocs/futures/en/#open-interest-statistics

**futures_order_book**(*\*\*params*)
    Get the Order Book for the market

    https://binance-docs.github.io/apidocs/futures/en/#order-book-market_data

**futures_orderbook_ticker**(*\*\*params*)
    Best price/qty on the order book for a symbol or symbols.

    https://binance-docs.github.io/apidocs/futures/en/#symbol-order-book-ticker-market_data

**futures_ping**()
:   Test connectivity to the Rest API

    https://binance-docs.github.io/apidocs/futures/en/#test-connectivity

**futures_place_batch_order**(*\*\*params*)
:   Send in new orders.

    https://binance-docs.github.io/apidocs/futures/en/#place-multiple-orders-trade

    To avoid modifying the existing signature generation and parameter order logic, the url encoding is done on the special query param, batchOrders, in the early stage.

**futures_position_information**(*\*\*params*)
:   Get position information

    https://binance-docs.github.io/apidocs/futures/en/#position-information-user_data

**futures_position_margin_history**(*\*\*params*)
:   Get position margin change history

    https://binance-docs.github.io/apidocs/futures/en/#get-postion-margin-change-history-trade

**futures_recent_trades**(*\*\*params*)
:   Get recent trades (up to last 500).

    https://binance-docs.github.io/apidocs/futures/en/#recent-trades-list-market_data

**futures_stream_close**(*listenKey*)

**futures_stream_get_listen_key**()

**futures_stream_keepalive**(*listenKey*)

**futures_symbol_ticker**(*\*\*params*)
:   Latest price for a symbol or symbols.

    https://binance-docs.github.io/apidocs/futures/en/#symbol-price-ticker-market_data

**futures_ticker**(*\*\*params*)
:   24 hour rolling window price change statistics.

    https://binance-docs.github.io/apidocs/futures/en/#24hr-ticker-price-change-statistics-market_data

**futures_time**()
:   Test connectivity to the Rest API and get the current server time.

    https://binance-docs.github.io/apidocs/futures/en/#check-server-time

**futures_top_longshort_account_ratio**(*\*\*params*)
:   Get present long to short ratio for top accounts of a specific symbol.

    https://binance-docs.github.io/apidocs/futures/en/#top-trader-long-short-ratio-accounts-market_data

**futures_top_longshort_position_ratio**(*\*\*params*)
:   Get present long to short ratio for top positions of a specific symbol.

    https://binance-docs.github.io/apidocs/futures/en/#top-trader-long-short-ratio-positions

**get_account**(*\*\*params*)
:   Get current account information.

    https://binance-docs.github.io/apidocs/spot/en/#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** BinanceRequestException, BinanceAPIException

**get_account_api_permissions**(*\*\*params*)

> Fetch api key permissions.
>
> https://binance-docs.github.io/apidocs/spot/en/#get-api-key-permission-user_data
>
> > **Parameters** **recvWindow** (*int*) – the number of milliseconds the request is valid for
> >
> > **Returns** API response

```
{
   "ipRestrict": false,
   "createTime": 1623840271000,
   "enableWithdrawals": false,   // This option allows you to withdraw via␣
↪API. You must apply the IP Access Restriction filter in order to enable␣
↪withdrawals
   "enableInternalTransfer": true,  // This option authorizes this key to␣
↪transfer funds between your master account and your sub account instantly
   "permitsUniversalTransfer": true,  // Authorizes this key to be used for a␣
↪dedicated universal transfer API to transfer multiple supported currencies.␣
↪Each business's own transfer API rights are not affected by this␣
↪authorization
   "enableVanillaOptions": false,  //  Authorizes this key to Vanilla options␣
↪trading
   "enableReading": true,
   "enableFutures": false,  //  API Key created before your futures account␣
↪opened does not support futures API service
   "enableMargin": false,    //  This option can be adjusted after the Cross␣
↪Margin account transfer is completed
   "enableSpotAndMarginTrading": false, // Spot and margin trading
   "tradingAuthorityExpirationTime": 1628985600000  // Expiration time for␣
↪spot and margin trading permission
}
```

**get_account_api_trading_status**(*\*\*params*)

Fetch account api trading status detail.

https://binance-docs.github.io/apidocs/spot/en/#account-api-trading-status-sapi-user_data

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

**Returns** API response

```
{
    "data": {          // API trading status detail
        "isLocked": false,   // API trading function is locked or not
        "plannedRecoverTime": 0,  // If API trading function is locked, this
→is the planned recover time
        "triggerCondition": {
                "GCR": 150,  // Number of GTC orders
                "IFER": 150, // Number of FOK/IOC orders
                "UFR": 300   // Number of orders
        },
        "indicators": {  // The indicators updated every 30 seconds
            "BTCUSDT": [  // The symbol
                {
                    "i": "UFR",  // Unfilled Ratio (UFR)
                    "c": 20,     // Count of all orders
                    "v": 0.05,   // Current UFR value
                    "t": 0.995   // Trigger UFR value
                },
                {
                    "i": "IFER", // IOC/FOK Expiration Ratio (IFER)
                    "c": 20,     // Count of FOK/IOC orders
                    "v": 0.99,   // Current IFER value
                    "t": 0.99    // Trigger IFER value
                },
                {
                    "i": "GCR",  // GTC Cancellation Ratio (GCR)
                    "c": 20,     // Count of GTC orders
                    "v": 0.99,   // Current GCR value
                    "t": 0.99    // Trigger GCR value
                }
            ],
            "ETHUSDT": [
                {
                    "i": "UFR",
                    "c": 20,
                    "v": 0.05,
                    "t": 0.995
                },
                {
                    "i": "IFER",
                    "c": 20,
                    "v": 0.99,
                    "t": 0.99
                },
                {
                    "i": "GCR",
                    "c": 20,
                    "v": 0.99,
                    "t": 0.99
                }
            ]
```

```
        },
        "updateTime": 1547630471725
    }
}
```

**get_account_snapshot** (*\*\*params*)

Get daily account snapshot of specific type.

https://binance-docs.github.io/apidocs/spot/en/#daily-account-snapshot-user_data

> **Parameters**
>
> - **type** (*string*) – required. Valid values are SPOT/MARGIN/FUTURES.
> - **startTime** (*int*) – optional
> - **endTime** (*int*) – optional
> - **limit** (*int*) – optional
> - **recvWindow** (*int*) – optional
>
> **Returns** API response

```
{
    "code":200, // 200 for success; others are error codes
    "msg":"", // error message
    "snapshotVos":[
        {
            "data":{
                "balances":[
                    {
                        "asset":"BTC",
                        "free":"0.09905021",
                        "locked":"0.00000000"
                    },
                    {
                        "asset":"USDT",
                        "free":"1.89109409",
                        "locked":"0.00000000"
                    }
                ],
                "totalAssetOfBtc":"0.09942700"
            },
            "type":"spot",
            "updateTime":1576281599000
        }
    ]
}
```

OR

```
{
    "code":200, // 200 for success; others are error codes
    "msg":"", // error message
    "snapshotVos":[
        {
            "data":{
                "marginLevel":"2748.02909813",
```

---

```
            "totalAssetOfBtc":"0.00274803",
            "totalLiabilityOfBtc":"0.00000100",
            "totalNetAssetOfBtc":"0.00274750",
            "userAssets":[
                {
                    "asset":"XRP",
                    "borrowed":"0.00000000",
                    "free":"1.00000000",
                    "interest":"0.00000000",
                    "locked":"0.00000000",
                    "netAsset":"1.00000000"
                }
            ]
        },
        "type":"margin",
        "updateTime":1576281599000
    }
  ]
}
```

OR

```
{
    "code":200, // 200 for success; others are error codes
    "msg":"", // error message
    "snapshotVos":[
        {
            "data":{
                "assets":[
                    {
                        "asset":"USDT",
                        "marginBalance":"118.99782335",
                        "walletBalance":"120.23811389"
                    }
                ],
                "position":[
                    {
                        "entryPrice":"7130.41000000",
                        "markPrice":"7257.66239673",
                        "positionAmt":"0.01000000",
                        "symbol":"BTCUSDT",
                        "unRealizedProfit":"1.24029054"
                    }
                ]
            },
            "type":"futures",
            "updateTime":1576281599000
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_account_status** (*\*\*params*)
    Get account status detail.

    https://binance-docs.github.io/apidocs/spot/en/#account-status-sapi-user_data

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

> **Returns** API response

```
{
    "data": "Normal"
}
```

`get_aggregate_trades`(*\*\*params*) → Dict[KT, VT]
> Get compressed, aggregate trades. Trades that fill at the time, from the same order, with the same price will have the quantity aggregated.

> https://binance-docs.github.io/apidocs/spot/en/#compressed-aggregate-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 1000.

> **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** BinanceRequestException, BinanceAPIException

`get_all_coins_info`(*\*\*params*)
> Get information of coins (available for deposit and withdraw) for user.

> https://binance-docs.github.io/apidocs/spot/en/#all-coins-39-information-user_data

> **Parameters** `recvWindow` (`int`) – optional

> **Returns** API response

```
{
    "coin": "BTC",
    "depositAllEnable": true,
    "withdrawAllEnable": true,
    "name": "Bitcoin",
    "free": "0",
    "locked": "0",
    "freeze": "0",
    "withdrawing": "0",
```

```
        "ipoing": "0",
        "ipoable": "0",
        "storage": "0",
        "isLegalMoney": false,
        "trading": true,
        "networkList": [
            {
                "network": "BNB",
                "coin": "BTC",
                "withdrawIntegerMultiple": "0.00000001",
                "isDefault": false,
                "depositEnable": true,
                "withdrawEnable": true,
                "depositDesc": "",
                "withdrawDesc": "",
                "specialTips": "Both a MEMO and an Address are required to
→successfully deposit your BEP2-BTCB tokens to Binance.",
                "name": "BEP2",
                "resetAddressStatus": false,
                "addressRegex": "^(bnb1)[0-9a-z]{38}$",
                "memoRegex": "^[0-9A-Za-z-_]{1,120}$",
                "withdrawFee": "0.0000026",
                "withdrawMin": "0.0000052",
                "withdrawMax": "0",
                "minConfirm": 1,
                "unLockConfirm": 0
            },
            {
                "network": "BTC",
                "coin": "BTC",
                "withdrawIntegerMultiple": "0.00000001",
                "isDefault": true,
                "depositEnable": true,
                "withdrawEnable": true,
                "depositDesc": "",
                "withdrawDesc": "",
                "specialTips": "",
                "name": "BTC",
                "resetAddressStatus": false,
                "addressRegex": "^[13][a-km-zA-HJ-NP-Z1-9]{25,34}$|^(bc1)[0-9A-Za-
→z]{39,59}$",
                "memoRegex": "",
                "withdrawFee": "0.0005",
                "withdrawMin": "0.001",
                "withdrawMax": "0",
                "minConfirm": 1,
                "unLockConfirm": 2
            }
        ]
}
```

**Raises** BinanceRequestException, BinanceAPIException

**get_all_isolated_margin_symbols**(*\*\*params*)
    Query isolated margin symbol info for all pairs

    https://binance-docs.github.io/apidocs/spot/en/#get-all-isolated-margin-symbol-user_data

```
pair_details = client.get_all_isolated_margin_symbols()
```

> **Returns** API response

```
[
    {
        "base": "BNB",
        "isBuyAllowed": true,
        "isMarginTrade": true,
        "isSellAllowed": true,
        "quote": "BTC",
        "symbol": "BNBBTC"
    },
    {
        "base": "TRX",
        "isBuyAllowed": true,
        "isMarginTrade": true,
        "isSellAllowed": true,
        "quote": "BTC",
        "symbol": "TRXBTC"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_all_margin_orders**(*\*\*params*)
    Query all margin accounts orders

    If orderId is set, it will get orders >= that orderId. Otherwise most recent orders are returned.

    For some historical orders cummulativeQuoteQty will be < 0, meaning the data is not available at this time.

    https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-all-order-user_data

> **Parameters**
> - **symbol** (*str*) – required
> - **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')
> - **orderId** (*str*) – optional
> - **startTime** (*str*) – optional
> - **endTime** (*str*) – optional
> - **limit** (*int*) – Default 500; max 1000
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for

> **Returns**

> API response

> [

>> { "id": 43123876, "price": "0.00395740", "qty": "4.06000000", "quoteQty": "0.01606704", "symbol": "BNBBTC", "time": 1556089977693

>> }, {

> > "id": 43123877, "price": "0.00395740", "qty": "0.77000000", "quoteQty": "0.00304719", "symbol": "BNBBTC", "time": 1556089977693
>
> > }, {
>
> > "id": 43253549, "price": "0.00428930", "qty": "23.30000000", "quoteQty": "0.09994069", "symbol": "BNBBTC", "time": 1556163963504
>
> > }
>
> ]

> **Raises** BinanceRequestException, BinanceAPIException

**get_all_orders**(*\*\*params*)
> Get all account orders; active, canceled, or filled.

> https://binance-docs.github.io/apidocs/spot/en/#all-orders-user_data

> > **Parameters**
> >
> > - **symbol** (*str*) – required
> > - **orderId** (*int*) – The unique order id
> > - **startTime** (*int*) – optional
> > - **endTime** (*int*) – optional
> > - **limit** (*int*) – Default 500; max 1000.
> > - **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** BinanceRequestException, BinanceAPIException

**get_all_tickers**() → List[Dict[str, str]]
> Latest price for all symbols.

> https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker

> > **Returns** List of market tickers

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

> **Raises** BinanceRequestException, BinanceAPIException

**get_asset_balance**(*asset*, *\*\*params*)
> Get current asset balance.
>
> > **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** BinanceRequestException, BinanceAPIException

**get_asset_details**(*\*\*params*)
> Fetch details on assets.
>
> https://binance-docs.github.io/apidocs/spot/en/#asset-detail-sapi-user_data
>
> > **Parameters**
> >
> > - **asset** (*str*) – optional
> >
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for
> >
> > **Returns** API response

```
{
        "CTR": {
            "minWithdrawAmount": "70.00000000", //min withdraw amount
            "depositStatus": false,//deposit status (false if ALL of networks
↪' are false)
            "withdrawFee": 35, // withdraw fee
            "withdrawStatus": true, //withdraw status (false if ALL of
↪networks' are false)
            "depositTip": "Delisted, Deposit Suspended" //reason
        },
        "SKY": {
            "minWithdrawAmount": "0.02000000",
            "depositStatus": true,
            "withdrawFee": 0.01,
```

(continues on next page)

```
            "withdrawStatus": true
        }
}
```

**get_asset_dividend_history**(*\*\*params*)
    Query asset dividend record.

    https://binance-docs.github.io/apidocs/spot/en/#asset-dividend-record-user_data

    > **Parameters**

    >> - **asset** (*str*) – optional

    >> - **startTime** (*long*) – optional

    >> - **endTime** (*long*) – optional

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

```
result = client.get_asset_dividend_history()
```

    > **Returns** API response

```
{
    "rows":[
        {
            "amount":"10.00000000",
            "asset":"BHFT",
            "divTime":1563189166000,
            "enInfo":"BHFT distribution",
            "tranId":2968885920
        },
        {
            "amount":"10.00000000",
            "asset":"BHFT",
            "divTime":1563189165000,
            "enInfo":"BHFT distribution",
            "tranId":2968885920
        }
    ],
    "total":2
}
```

    > **Raises** BinanceRequestException, BinanceAPIException

**get_avg_price**(*\*\*params*)
    Current average price for a symbol.

    https://binance-docs.github.io/apidocs/spot/en/#current-average-price

    > **Parameters symbol** (*str*) –

    > **Returns** API response

```
{
    "mins": 5,
    "price": "9.35751834"
}
```

**get_bnb_burn_spot_margin**(*\*\*params*)
  Get BNB Burn Status

  https://binance-docs.github.io/apidocs/spot/en/#get-bnb-burn-status-user_data

```
status = client.get_bnb_burn_spot_margin()
```

  **Returns** API response

```
{
    "spotBNBBurn":true,
    "interestBNBBurn": false
}
```

  **Raises** BinanceRequestException, BinanceAPIException

**get_c2c_trade_history**(*\*\*params*)
  Get C2C Trade History

  https://binance-docs.github.io/apidocs/spot/en/#get-c2c-trade-history-user_data

  **Parameters**

  - **tradeType** (*str*) – required - BUY, SELL
  - **startTimestamp** – optional
  - **endTimestamp** (*int*) – optional
  - **page** (*int*) – optional - default 1
  - **rows** (*int*) – optional - default 100, max 100
  - **recvWindow** (*int*) – optional

  **Returns**

  API response

  **{** "code": "000000", "message": "success", "data": [

    **{** "orderNumber":"20219644646554779648", "advNo": "11218246497340923904", "tradeType": "SELL", "asset": "BUSD", "fiat": "CNY", "fiatSymbol": "", "amount": "5000.00000000", // Quantity (in Crypto) "totalPrice": "33400.00000000", "unitPrice": "6.68", // Unit Price (in Fiat) "orderStatus": "COMPLETED", // PENDING, TRADING, BUYER_PAYED, DISTRIBUTING, COMPLETED, IN_APPEAL, CANCELLED, CANCELLED_BY_SYSTEM "createTime": 1619361369000, "commission": "0", // Transaction Fee (in Crypto) "counterPartNickName": "ab***", "advertisementRole": "TAKER"

    **}**

  ], "total": 1, "success": true

  **}**

**get_deposit_address**(*coin: str, network: Optional[str] = None, \*\*params*)
  Fetch a deposit address for a symbol

  https://binance-docs.github.io/apidocs/spot/en/#deposit-address-supporting-network-user_data

  **Parameters**

- **coin** (`str`) – required

- **network** (`str`) – optional

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

**Returns** API response

```
{
    "address": "1HPn8Rx2y6nNSfagQBKy27GB99Vbzg89wv",
    "coin": "BTC",
    "tag": "",
    "url": "https://btc.com/1HPn8Rx2y6nNSfagQBKy27GB99Vbzg89wv"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**get_deposit_history**(*\*\*params*)
    Fetch deposit history.

https://binance-docs.github.io/apidocs/spot/en/#deposit-history-supporting-network-user_data

**Parameters**

- **coin** (`str`) – optional

- **startTime** (`long`) – optional

- **endTime** (`long`) – optional

- **offset** (`long`) – optional - default:0

- **limit** (`long`) – optional

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

**Returns** API response

```
[
    {
        "amount":"0.00999800",
        "coin":"PAXG",
        "network":"ETH",
        "status":1,
        "address":"0x788cabe9236ce061e5a892e1a59395a81fc8d62c",
        "addressTag":"",
        "txId":
↪"0xaad4654a3234aa6118af9b4b335f5ae81c360b2394721c019b5d1e75328b09f3",
        "insertTime":1599621997000,
        "transferType":0,
        "confirmTimes":"12/12"
    },
    {
        "amount":"0.50000000",
        "coin":"IOTA",
        "network":"IOTA",
        "status":1,
        "address":
↪"SIZ9VLMHWATXKV99LH99CIGFJFUMLEHGWVZVNNZXRJJVWBPHYWPPBOSDORZ9EQSHCZAMPVAPGFYQAUUV9DROOXJL
↪",
        "addressTag":"",
```

(continues on next page)

```
        "txId":
→"ESBFVQUTPIWQNJSPXFNHNYHSQNTGKRVKPRABQWTAXCDWOAKDKYWPTVG9BGXNVNKTLEJGESAVXIKIZ9999
→",
        "insertTime":1599620082000,
        "transferType":0,
        "confirmTimes":"1/1"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_dust_log**(*\*\*params*)
> Get log of small amounts exchanged for BNB.

> https://binance-docs.github.io/apidocs/spot/en/#dustlog-sapi-user_data

>> **Parameters**
>>
>> • **startTime** (*int*) – optional
>>
>> • **endTime** (*int*) – optional
>>
>> • **recvWindow** (*int*) – the number of milliseconds the request is valid for

>> **Returns** API response

```
{
    "total": 8,   //Total counts of exchange
    "userAssetDribblets": [
        {
            "totalTransferedAmount": "0.00132256",   // Total transfered BNB
→amount for this exchange.
            "totalServiceChargeAmount": "0.00002699",    //Total service
→charge amount for this exchange.
            "transId": 45178372831,
            "userAssetDribbletDetails": [          //Details of  this
→exchange.
                {
                    "transId": 4359321,
                    "serviceChargeAmount": "0.000009",
                    "amount": "0.0009",
                    "operateTime": 1615985535000,
                    "transferedAmount": "0.000441",
                    "fromAsset": "USDT"
                },
                {
                    "transId": 4359321,
                    "serviceChargeAmount": "0.00001799",
                    "amount": "0.0009",
                    "operateTime": "2018-05-03 17:07:04",
                    "transferedAmount": "0.00088156",
                    "fromAsset": "ETH"
                }
            ]
        },
        {
            "operateTime":1616203180000,
            "totalTransferedAmount": "0.00058795",
```

```
                "totalServiceChargeAmount": "0.000012",
                "transId": 4357015,
                "userAssetDribbletDetails": [
                    {
                        "transId": 4357015,
                        "serviceChargeAmount": "0.00001"
                        "amount": "0.001",
                        "operateTime": 1616203180000,
                        "transferedAmount": "0.00049",
                        "fromAsset": "USDT"
                    },
                    {
                        "transId": 4357015,
                        "serviceChargeAmount": "0.000002"
                        "amount": "0.0001",
                        "operateTime": 1616203180000,
                        "transferedAmount": "0.00009795",
                        "fromAsset": "ETH"
                    }
                ]
            }
        ]
}
```

**get_exchange_info**() → Dict[KT, VT]
    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** BinanceRequestException, BinanceAPIException

**get_fiat_deposit_withdraw_history**(*\*\*params*)
>   Get Fiat Deposit/Withdraw History

>   https://binance-docs.github.io/apidocs/spot/en/#get-fiat-deposit-withdraw-history-user_data

>   **Parameters**
>   >   - **transactionType** (*str*) – required - 0-deposit,1-withdraw
>   >   - **beginTime** (*int*) – optional
>   >   - **endTime** (*int*) – optional
>   >   - **page** (*int*) – optional - default 1
>   >   - **rows** (*int*) – optional - default 100, max 500
>   >   - **recvWindow** (*int*) – optional

**get_fiat_payments_history**(*\*\*params*)
>   Get Fiat Payments History

>   https://binance-docs.github.io/apidocs/spot/en/#get-fiat-payments-history-user_data

>   **Parameters**
>   >   - **transactionType** (*str*) – required - 0-buy,1-sell
>   >   - **beginTime** (*int*) – optional
>   >   - **endTime** (*int*) – optional
>   >   - **page** (*int*) – optional - default 1
>   >   - **rows** (*int*) – optional - default 100, max 500
>   >   - **recvWindow** (*int*) – optional

**get_fixed_activity_project_list**(*\*\*params*)
    Get Fixed and Activity Project List

    https://binance-docs.github.io/apidocs/spot/en/#get-fixed-and-activity-project-list-user_data

> **Parameters**
>
> - **asset** (*str*) – optional
>
> - **type** (*str*) – required - "ACTIVITY", "CUSTOMIZED_FIXED"
>
> - **status** (*str*) – optional - "ALL", "SUBSCRIBABLE", "UNSUBSCRIBABLE"; default "ALL"
>
> - **sortBy** (*str*) – optional - "START_TIME", "LOT_SIZE", "INTEREST_RATE", "DURATION"; default "START_TIME"
>
> - **current** (*int*) – optional - Currently querying page. Start from 1. Default:1
>
> - **size** (*int*) – optional - Default:10, Max:100
>
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns** API response

```
[
    {
        "asset": "USDT",
        "displayPriority": 1,
        "duration": 90,
        "interestPerLot": "1.35810000",
        "interestRate": "0.05510000",
        "lotSize": "100.00000000",
        "lotsLowLimit": 1,
        "lotsPurchased": 74155,
        "lotsUpLimit": 80000,
        "maxLotsPerUser": 2000,
        "needKyc": False,
        "projectId": "CUSDT90DAYSS001",
        "projectName": "USDT",
        "status": "PURCHASING",
        "type": "CUSTOMIZED_FIXED",
        "withAreaLimitation": False
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_historical_klines**(*symbol*, *interval*, *start_str*, *end_str=None*, *limit=500*, *klines_type: binance.enums.HistoricalKlinesType = <HistoricalKlinesType.SPOT: 1>*)
    Get Historical Klines from Binance

> **Parameters**
>
> - **symbol** (*str*) – Name of symbol pair e.g BNBBTC
>
> - **interval** (*str*) – Binance Kline interval
>
> - **start_str** (*str* | *int*) – Start date string in UTC format or timestamp in milliseconds

- **end_str** (`str`|`int`) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)

- **limit** (`int`) – Default 500; max 1000.

- **klines_type** (`HistoricalKlinesType`) – Historical klines type: SPOT or FUTURES

> **Returns** list of OHLCV values

**get_historical_klines_generator**(*symbol*, *interval*, *start_str*, *end_str=None*, *klines_type: binance.enums.HistoricalKlinesType = <HistoricalKlinesType.SPOT: 1>*)

Get Historical Klines generator from Binance

> **Parameters**
>
> - **symbol** (`str`) – Name of symbol pair e.g BNBBTC
>
> - **interval** (`str`) – Binance Kline interval
>
> - **start_str** (`str`|`int`) – Start date string in UTC format or timestamp in milliseconds
>
> - **end_str** (`str`|`int`) – optional - end date string in UTC format or timestamp in milliseconds (default will fetch everything up to now)
>
> - **klines_type** (`HistoricalKlinesType`) – Historical klines type: SPOT or FUTURES
>
> **Returns** generator of OHLCV values

**get_historical_trades**(*\*\*params*) → Dict[KT, VT]

Get older trades.

https://binance-docs.github.io/apidocs/spot/en/#old-trade-lookup

> **Parameters**
>
> - **symbol** (`str`) – required
>
> - **limit** (`int`) – Default 500; max 1000.
>
> - **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** BinanceRequestException, BinanceAPIException

**get_isolated_margin_account**(*\*\*params*)

Query isolated margin account details

https://binance-docs.github.io/apidocs/spot/en/#query-isolated-margin-account-info-user_data

> **Parameters symbols** – optional up to 5 margin pairs as a comma separated string

```
account_info = client.get_isolated_margin_account()
account_info = client.get_isolated_margin_account(symbols="BTCUSDT,ETHUSDT")
```

**Returns** API response

```
If "symbols" is not sent:

    {
    "assets":[
        {
            "baseAsset":
            {
            "asset": "BTC",
            "borrowEnabled": true,
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000",
            "netAssetOfBtc": "0.00000000",
            "repayEnabled": true,
            "totalAsset": "0.00000000"
            },
            "quoteAsset":
            {
            "asset": "USDT",
            "borrowEnabled": true,
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000",
            "netAssetOfBtc": "0.00000000",
            "repayEnabled": true,
            "totalAsset": "0.00000000"
            },
            "symbol": "BTCUSDT"
            "isolatedCreated": true,
            "marginLevel": "0.00000000",
            "marginLevelStatus": "EXCESSIVE", // "EXCESSIVE", "NORMAL",
→"MARGIN_CALL", "PRE_LIQUIDATION", "FORCE_LIQUIDATION"
            "marginRatio": "0.00000000",
            "indexPrice": "10000.00000000"
            "liquidatePrice": "1000.00000000",
            "liquidateRate": "1.00000000"
            "tradeEnabled": true
        }
        ],
        "totalAssetOfBtc": "0.00000000",
        "totalLiabilityOfBtc": "0.00000000",
        "totalNetAssetOfBtc": "0.00000000"
    }

If "symbols" is sent:
```

```
    {
    "assets":[
        {
            "baseAsset":
            {
            "asset": "BTC",
            "borrowEnabled": true,
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000",
            "netAssetOfBtc": "0.00000000",
            "repayEnabled": true,
            "totalAsset": "0.00000000"
            },
            "quoteAsset":
            {
            "asset": "USDT",
            "borrowEnabled": true,
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000",
            "netAssetOfBtc": "0.00000000",
            "repayEnabled": true,
            "totalAsset": "0.00000000"
            },
            "symbol": "BTCUSDT"
            "isolatedCreated": true,
            "marginLevel": "0.00000000",
            "marginLevelStatus": "EXCESSIVE", // "EXCESSIVE", "NORMAL",
→"MARGIN_CALL", "PRE_LIQUIDATION", "FORCE_LIQUIDATION"
            "marginRatio": "0.00000000",
            "indexPrice": "10000.00000000"
            "liquidatePrice": "1000.00000000",
            "liquidateRate": "1.00000000"
            "tradeEnabled": true
        }
        ]
    }
```

**get_isolated_margin_symbol**(*\*\*params*)

Query isolated margin symbol info

https://binance-docs.github.io/apidocs/spot/en/#query-isolated-margin-symbol-user_data

> **Parameters symbol** (*str*) – name of the symbol pair

```
pair_details = client.get_isolated_margin_symbol(symbol='BTCUSDT')
```

> **Returns** API response

```
{
"symbol":"BTCUSDT",
```

```
"base":"BTC",
"quote":"USDT",
"isMarginTrade":true,
"isBuyAllowed":true,
"isSellAllowed":true
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_klines**(*\*\*params*) → Dict[KT, VT]

> Kline/candlestick bars for a symbol. Klines are uniquely identified by their open time.
>
> https://binance-docs.github.io/apidocs/spot/en/#kline-candlestick-data
>
> **Parameters**
>
> - **symbol** (*str*) – required
> - **interval** (*str*) –
>   –
> - **limit** (*int*) –
>   – Default 500; max 1000.
> - **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** BinanceRequestException, BinanceAPIException

**get_lending_account**(*\*\*params*)

> Get Lending Account Details
>
> https://binance-docs.github.io/apidocs/spot/en/#lending-account-user_data

**get_lending_daily_quota_left**(*\*\*params*)

> Get Left Daily Purchase Quota of Flexible Product.

https://binance-docs.github.io/apidocs/spot/en/#get-left-daily-purchase-quota-of-flexible-product-user_data

**get_lending_daily_redemption_quota**(*\*\*params*)
 Get Left Daily Redemption Quota of Flexible Product

 https://binance-docs.github.io/apidocs/spot/en/#get-left-daily-redemption-quota-of-flexible-product-user_data

**get_lending_interest_history**(*\*\*params*)
 Get Lending Interest History

 https://binance-docs.github.io/apidocs/spot/en/#get-interest-history-user_data-2

**get_lending_position**(*\*\*params*)
 Get Flexible Product Position

 https://binance-docs.github.io/apidocs/spot/en/#get-flexible-product-position-user_data

**get_lending_product_list**(*\*\*params*)
 Get Lending Product List

 https://binance-docs.github.io/apidocs/spot/en/#get-flexible-product-list-user_data

**get_lending_purchase_history**(*\*\*params*)
 Get Lending Purchase History

 https://binance-docs.github.io/apidocs/spot/en/#get-purchase-record-user_data

**get_lending_redemption_history**(*\*\*params*)
 Get Lending Redemption History

 https://binance-docs.github.io/apidocs/spot/en/#get-redemption-record-user_data

**get_margin_account**(*\*\*params*)
 Query cross-margin account details

 https://binance-docs.github.io/apidocs/spot/en/#query-cross-margin-account-details-user_data

 **Returns** API response

```
{
    "borrowEnabled": true,
    "marginLevel": "11.64405625",
    "totalAssetOfBtc": "6.82728457",
    "totalLiabilityOfBtc": "0.58633215",
    "totalNetAssetOfBtc": "6.24095242",
    "tradeEnabled": true,
    "transferEnabled": true,
    "userAssets": [
        {
            "asset": "BTC",
            "borrowed": "0.00000000",
            "free": "0.00499500",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00499500"
        },
        {
            "asset": "BNB",
            "borrowed": "201.66666672",
            "free": "2346.50000000",
            "interest": "0.00000000",
```

(continues on next page)

```
                "locked": "0.00000000",
                "netAsset": "2144.83333328"
            },
            {
                "asset": "ETH",
                "borrowed": "0.00000000",
                "free": "0.00000000",
                "interest": "0.00000000",
                "locked": "0.00000000",
                "netAsset": "0.00000000"
            },
            {
                "asset": "USDT",
                "borrowed": "0.00000000",
                "free": "0.00000000",
                "interest": "0.00000000",
                "locked": "0.00000000",
                "netAsset": "0.00000000"
            }
        ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_margin_all_assets**(*\*\*params*)
> Get All Margin Assets (MARKET_DATA)
>
> https://binance-docs.github.io/apidocs/spot/en/#get-all-margin-assets-market_data

```
margin_assets = client.get_margin_all_assets()
```

> **Returns** API response

```
[
    {
        "assetFullName": "USD coin",
        "assetName": "USDC",
        "isBorrowable": true,
        "isMortgageable": true,
        "userMinBorrow": "0.00000000",
        "userMinRepay": "0.00000000"
    },
    {
        "assetFullName": "BNB-coin",
        "assetName": "BNB",
        "isBorrowable": true,
        "isMortgageable": true,
        "userMinBorrow": "1.00000000",
        "userMinRepay": "0.00000000"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_margin_all_pairs**(*\*\*params*)
    Get All Cross Margin Pairs (MARKET_DATA)

    https://binance-docs.github.io/apidocs/spot/en/#get-all-cross-margin-pairs-market_data

```
margin_pairs = client.get_margin_all_pairs()
```

**Returns** API response

```
[
    {
        "base": "BNB",
        "id": 351637150141315861,
        "isBuyAllowed": true,
        "isMarginTrade": true,
        "isSellAllowed": true,
        "quote": "BTC",
        "symbol": "BNBBTC"
    },
    {
        "base": "TRX",
        "id": 351637923235429141,
        "isBuyAllowed": true,
        "isMarginTrade": true,
        "isSellAllowed": true,
        "quote": "BTC",
        "symbol": "TRXBTC"
    }
]
```

**Raises** BinanceRequestException, BinanceAPIException

**get_margin_asset**(*\*\*params*)
    Query cross-margin asset

    https://binance-docs.github.io/apidocs/spot/en/#query-margin-asset-market_data

    **Parameters** **asset** (*str*) – name of the asset

```
asset_details = client.get_margin_asset(asset='BNB')
```

**Returns** API response

```
{
    "assetFullName": "Binance Coin",
    "assetName": "BNB",
    "isBorrowable": false,
    "isMortgageable": true,
    "userMinBorrow": "0.00000000",
    "userMinRepay": "0.00000000"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**get_margin_force_liquidation_rec**(*\*\*params*)
    Get Force Liquidation Record (USER_DATA)

https://binance-docs.github.io/apidocs/spot/en/#get-force-liquidation-record-user_data

> **Parameters**
>
> > - **startTime** (*str*) –
> > - **endTime** (*str*) –
> > - **isolatedSymbol** (*str*) – isolated symbol (if querying isolated margin)
> > - **current** (*str*) – Currently querying page. Start from 1. Default:1
> > - **size** (*int*) – Default:10 Max:100
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns**
>
> > API response
> >
> > {
> >
> > > "rows": [
> > >
> > > > { "avgPrice": "0.00388359", "executedQty": "31.39000000", "orderId": 180015097, "price": "0.00388110", "qty": "31.39000000", "side": "SELL", "symbol": "BNBBTC", "timeInForce": "GTC", "isIsolated": true, "updatedTime": 1558941374745
> > > >
> > > > }
> > >
> > > ], "total": 1
> >
> > }

**get_margin_interest_history**(*\*\*params*)

> Get Interest History (USER_DATA)

https://binance-docs.github.io/apidocs/spot/en/#get-interest-history-user_data

> **Parameters**
>
> > - **asset** (*str*) –
> > - **isolatedSymbol** (*str*) – isolated symbol (if querying isolated margin)
> > - **startTime** (*str*) –
> > - **endTime** (*str*) –
> > - **current** (*str*) – Currently querying page. Start from 1. Default:1
> > - **size** (*int*) – Default:10 Max:100
> > - **archived** (*bool*) – Default: false. Set to true for archived data from 6 months ago
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns**
>
> > API response
> >
> > {
> >
> > > "rows":[

{ "isolatedSymbol": "BNBUSDT", // isolated symbol, will not be returned for crossed margin "asset": "BNB", "interest": "0.02414667", "interestAccuredTime": 1566813600000, "interestRate": "0.01600000", "principal": "36.22000000", "type": "ON_BORROW"

}

], "total": 1

}

**get_margin_loan_details**(*\*\*params*)
> Query loan record
>
> txId or startTime must be sent. txId takes precedence.
>
> https://binance-docs.github.io/apidocs/spot/en/#query-loan-record-user_data
>
> **Parameters**
>> - **asset** (*str*) – required
>> - **isolatedSymbol** (*str*) – isolated symbol (if querying isolated margin)
>> - **txId** (*str*) – the tranId in of the created loan
>> - **startTime** (*str*) – earliest timestamp to filter transactions
>> - **endTime** (*str*) – Used to uniquely identify this cancel. Automatically generated by default.
>> - **current** (*str*) – Currently querying page. Start from 1. Default:1
>> - **size** (*int*) – Default:10 Max:100
>> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns**
>> API response
>>
>> {
>>> "rows": [
>>>> { "asset": "BNB", "principal": "0.84624403", "timestamp": 1555056425000, //one of PENDING (pending to execution), CONFIRMED (successfully loaned), FAILED (execution failed, nothing happened to your account); "status": "CONFIRMED"
>>>>
>>>> }
>>>
>>> ], "total": 1
>>
>> }
>
> **Raises** BinanceRequestException, BinanceAPIException

**get_margin_oco_order**(*\*\*params*)
> Retrieves a specific OCO based on provided optional parameters
>
> https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-oco-user_data
>
> **Parameters**
>> - **isIsolated** – for isolated margin or not, "TRUE", "FALSE"default "FALSE"
>> - **symbol** (*str*) – mandatory for isolated margin, not supported for cross margin

- **orderListId** (*int*) – Either orderListId or listClientOrderId must be provided
- **listClientOrderId** (*str*) – Either orderListId or listClientOrderId must be provided
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

**Returns**

API response

{ "orderListId": 27, "contingencyType": "OCO", "listStatusType": "EXEC_STARTED", "listOrderStatus": "EXECUTING", "listClientOrderId": "h2USkA5YQpaXHPIrkd96xE", "transactionTime": 1565245656253, "symbol": "LTCBTC", "isIsolated": false, // if isolated margin "orders": [

{ "symbol": "LTCBTC", "orderId": 4, "clientOrderId": "qD1gy3kc3Gx0rihm9Y3xwS"

}, {

"symbol": "LTCBTC", "orderId": 5, "clientOrderId": "ARzZ9I00CPM8i3NhmU9Ega"

}

]

}

**get_margin_order**(*\*\*params*)

Query margin accounts order

Either orderId or origClientOrderId must be sent.

For some historical orders cummulativeQuoteQty will be < 0, meaning the data is not available at this time.

https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-order-user_data

**Parameters**

- **symbol** (*str*) – required
- **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')
- **orderId** (*str*) –
- **origClientOrderId** (*str*) –
- **recvWindow** (*int*) – the number of milliseconds the request is valid for

**Returns**

API response

{ "clientOrderId": "ZwfQzuDIGpceVhKW5DvCmO", "cummulativeQuoteQty": "0.00000000", "executedQty": "0.00000000", "icebergQty": "0.00000000", "isWorking": true, "orderId": 213205622, "origQty": "0.30000000", "price": "0.00493630", "side": "SELL", "status": "NEW", "stopPrice": "0.00000000", "symbol": "BNBBTC", "time": 1562133008725, "timeInForce": "GTC", "type": "LIMIT", "updateTime": 1562133008725

}

**Raises** BinanceRequestException, BinanceAPIException

**get_margin_price_index**(*\*\*params*)
> Query margin priceIndex

> https://binance-docs.github.io/apidocs/spot/en/#query-margin-priceindex-market_data

>> **Parameters symbol** (*str*) – name of the symbol pair

```
price_index_details = client.get_margin_price_index(symbol='BTCUSDT')
```

>> **Returns** API response

```
{
    "calcTime": 1562046418000,
    "price": "0.00333930",
    "symbol": "BNBBTC"
}
```

>> **Raises** BinanceRequestException, BinanceAPIException

**get_margin_repay_details**(*\*\*params*)
> Query repay record

> txId or startTime must be sent. txId takes precedence.

> https://binance-docs.github.io/apidocs/spot/en/#query-repay-record-user_data

>> **Parameters**

>>> • **asset** (*str*) – required

>>> • **isolatedSymbol** (*str*) – isolated symbol (if querying isolated margin)

>>> • **txId** (*str*) – the tranId in of the created loan

>>> • **startTime** (*str*) –

>>> • **endTime** (*str*) – Used to uniquely identify this cancel. Automatically generated by default.

>>> • **current** (*str*) – Currently querying page. Start from 1. Default:1

>>> • **size** (*int*) – Default:10 Max:100

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

>> **Returns**

>> API response

>> {

>> "rows": [

>>> { //Total amount repaid "amount": "14.00000000", "asset": "BNB", //Interest repaid "interest": "0.01866667", //Principal repaid "principal": "13.98133333", //one of PENDING (pending to execution), CONFIRMED (successfully loaned), FAILED (execution failed, nothing happened to your account); "status": "CONFIRMED", "timestamp": 1563438204000, "txId": 2970933056

>>> }

>> ], "total": 1

>> }

> > > **Raises** BinanceRequestException, BinanceAPIException

**get_margin_symbol**(*\*\*params*)

> Query cross-margin symbol info
>
> https://binance-docs.github.io/apidocs/spot/en/#query-cross-margin-pair-market_data
>
> > **Parameters symbol** (*str*) – name of the symbol pair

```
pair_details = client.get_margin_symbol(symbol='BTCUSDT')
```

> > **Returns** API response

```
{
    "id":323355778339572400,
    "symbol":"BTCUSDT",
    "base":"BTC",
    "quote":"USDT",
    "isMarginTrade":true,
    "isBuyAllowed":true,
    "isSellAllowed":true
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**get_margin_trades**(*\*\*params*)

> Query margin accounts trades
>
> If fromId is set, it will get orders >= that fromId. Otherwise most recent orders are returned.
>
> https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-trade-list-user_data
>
> > **Parameters**
> >
> > - **symbol** (*str*) – required
> > - **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')
> > - **fromId** (*str*) – optional
> > - **startTime** (*str*) – optional
> > - **endTime** (*str*) – optional
> > - **limit** (*int*) – Default 500; max 1000
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for
> >
> > **Returns**
> >
> > API response
> >
> > [
> >
> > > { "commission": "0.00006000", "commissionAsset": "BTC", "id": 34, "isBestMatch": true, "isBuyer": false, "isMaker": false, "orderId": 39324, "price": "0.02000000", "qty": "3.00000000", "symbol": "BNBBTC", "time": 1561973357171
> > >
> > > }, { "commission": "0.00002950", "commissionAsset": "BTC", "id": 32, "isBestMatch": true, "isBuyer": false, "isMaker": true, "orderId": 39319, "price": "0.00590000", "qty": "5.00000000", "symbol": "BNBBTC", "time": 1561964645345

> }
>
> ]

> **Raises** BinanceRequestException, BinanceAPIException

**get_max_margin_loan**(*\*\*params*)

Query max borrow amount for an asset

https://binance-docs.github.io/apidocs/spot/en/#query-max-borrow-user_data

> **Parameters**
>
> - **asset** (`str`) – required
>
> - **isolatedSymbol** (`str`) – isolated symbol (if querying isolated margin)
>
> - **recvWindow** (`int`) – the number of milliseconds the request is valid for

> **Returns**
>
> API response
>
> **{** "amount": "1.69248805"
>
> **}**

> **Raises** BinanceRequestException, BinanceAPIException

**get_max_margin_transfer**(*\*\*params*)

Query max transfer-out amount

https://binance-docs.github.io/apidocs/spot/en/#query-max-transfer-out-amount-user_data

> **Parameters**
>
> - **asset** (`str`) – required
>
> - **isolatedSymbol** (`str`) – isolated symbol (if querying isolated margin)
>
> - **recvWindow** (`int`) – the number of milliseconds the request is valid for

> **Returns**
>
> API response
>
> **{** "amount": "3.59498107"
>
> **}**

> **Raises** BinanceRequestException, BinanceAPIException

**get_my_trades**(*\*\*params*)

Get trades for a specific symbol.

https://binance-docs.github.io/apidocs/spot/en/#account-trade-list-user_data

> **Parameters**
>
> - **symbol** (`str`) – required
>
> - **startTime** (`int`) – optional
>
> - **endTime** (`int`) – optional
>
> - **limit** (`int`) – Default 500; max 1000.
>
> - **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** BinanceRequestException, BinanceAPIException

**get_open_margin_oco_orders**(*\*\*params*)
    Retrieves open OCO trades

    https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-open-oco-user_data

    **Parameters**

    - **isIsolated** – for isolated margin or not, "TRUE", "FALSE"default "FALSE"

    - **symbol** (*str*) – mandatory for isolated margin, not supported for cross margin

    - **fromId** (*int*) – If supplied, neither startTime or endTime can be provided

    - **startTime** (*int*) – optional

    - **endTime** (*int*) – optional

    - **limit** (*int*) – optional Default Value: 500; Max Value: 1000

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

    **Returns**

    API response

    [

        { "orderListId": 29, "contingencyType": "OCO", "listStatusType": "EXEC_STARTED", "listOrderStatus": "EXECUTING", "listClientOrderId": "amEEAXryFzFwYF1FeRpUoZ", "transactionTime": 1565245913483, "symbol": "LTCBTC", "isIsolated": true, // if isolated margin "orders": [

            { "symbol": "LTCBTC", "orderId": 4, "clientOrderId": "oD7aesZqjEGlZrbtRpy5zB"

            }, {

                "symbol": "LTCBTC", "orderId": 5, "clientOrderId": "Jr1h6xirOxgeJOUuYQS7V3"

            }

        ]

    }, {

"orderListId": 28, "contingencyType": "OCO", "listStatusType":
"EXEC_STARTED", "listOrderStatus": "EXECUTING", "list-
ClientOrderId": "hG7hFNxJV6cZy3Ze4AUT4d", "transactionTime":
1565245913407, "symbol": "LTCBTC", "orders": [

{ "symbol": "LTCBTC", "orderId": 2, "clientOrderId":
"j6lFOfbmFMRjTYA7rRJ0LP"

}, {

"symbol": "LTCBTC", "orderId": 3, "clientOrderId":
"z0KCjOdditiLS5ekAFtK81"

}

]

}

]

**get_open_margin_orders**(*\*\*params*)
Query margin accounts open orders

If the symbol is not sent, orders for all symbols will be returned in an array (cross-margin only).

If querying isolated margin orders, both the isIsolated='TRUE' and symbol=symbol_name must be set.

When all symbols are returned, the number of requests counted against the rate limiter is equal to the
number of symbols currently trading on the exchange.

https://binance-docs.github.io/apidocs/spot/en/#query-margin-account-39-s-open-order-user_data

> **Parameters**
>
> - **symbol** (*str*) – optional
>
> - **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')
>
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns**
>
> API response
>
> [
>
> { "clientOrderId": "qhcZw71gAkCCTv0t0k8LUK", "cummulativeQuoteQty":
> "0.00000000", "executedQty": "0.00000000", "icebergQty": "0.00000000",
> "isWorking": true, "orderId": 211842552, "origQty": "0.30000000",
> "price": "0.00475010", "side": "SELL", "status": "NEW", "stopPrice":
> "0.00000000", "symbol": "BNBBTC", "time": 1562040170089, "timeIn-
> Force": "GTC", "type": "LIMIT", "updateTime": 1562040170089
>
> }
>
> ]
>
> **Raises** BinanceRequestException, BinanceAPIException

**get_open_orders**(*\*\*params*)
Get all open orders on a symbol.

https://binance-docs.github.io/apidocs/spot/en/#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** BinanceRequestException, BinanceAPIException

**get_order**(*\*\*params*)

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

https://binance-docs.github.io/apidocs/spot/en/#query-order-user_data

**Parameters**

- **symbol** (*str*) – required

- **orderId** (*int*) – The unique order id

- **origClientOrderId** (*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** BinanceRequestException, BinanceAPIException

**get_order_book**(*\*\*params*) → Dict[KT, VT]
Get the Order Book for the market

https://binance-docs.github.io/apidocs/spot/en/#order-book

> **Parameters**
>
> > • **symbol** (*str*) – required
> >
> > • **limit** (*int*) – Default 100; max 1000
>
> **Returns** API response

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

> **Raises** BinanceRequestException, BinanceAPIException

**get_orderbook_ticker**(*\*\*params*)
Latest price for a symbol or symbols.

https://binance-docs.github.io/apidocs/spot/en/#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": "100000.00000000",
            "askQty": "1000.00000000"
        }
    ]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_orderbook_tickers**() → Dict[KT, VT]
> Best price/qty on the order book for all symbols.
>
> https://binance-docs.github.io/apidocs/spot/en/#symbol-order-book-ticker
>
> > **Parameters** **symbol** (*str*) – optional
> >
> > **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": "100000.00000000",
            "askQty": "1000.00000000"
        }
    ]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_products**() → Dict[KT, VT]
> Return list of products currently listed on Binance
>
> Use get_exchange_info() call instead
>
> > **Returns** list - List of product dictionaries
> >
> > **Raises** BinanceRequestException, BinanceAPIException

**get_recent_trades**(*\*\*params*) → Dict[KT, VT]
> Get recent trades (up to last 500).
>
> https://binance-docs.github.io/apidocs/spot/en/#recent-trades-list
>
> > **Parameters**
> >
> > > - **symbol** (*str*) – required
> > >
> > > - **limit** (*int*) – Default 500; max 1000.
> >
> > **Returns** API response

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

> Raises BinanceRequestException, BinanceAPIException

**get_server_time**() → Dict[KT, VT]
> Test connectivity to the Rest API and get the current server time.

> https://binance-docs.github.io/apidocs/spot/en/#check-server-time

> > **Returns** Current server time

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

> Raises BinanceRequestException, BinanceAPIException

**get_sub_account_assets**(*\*\*params*)
> Fetch sub-account assets

> https://binance-docs.github.io/apidocs/spot/en/#query-sub-account-assets-sapi-for-master-account

> > **Parameters**
> > - **email** (*str*) – required
> > - **recvWindow** (*int*) – optional

> > **Returns** API response

```
{
    "balances":[
        {
            "asset":"ADA",
            "free":10000,
            "locked":0
        },
        {
            "asset":"BNB",
            "free":10003,
            "locked":0
        },
        {
            "asset":"BTC",
            "free":11467.6399,
            "locked":0
        },
        {
            "asset":"ETH",
```

(continues on next page)

```
            "free":10004.995,
            "locked":0
        },
        {

            "asset":"USDT",
            "free":11652.14213,
            "locked":0
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_sub_account_futures_transfer_history**(*\*\*params*)
  Query Sub-account Futures Transfer History.

  https://binance-docs.github.io/apidocs/spot/en/#query-sub-account-futures-asset-transfer-history-for-master-account

  > **Parameters**
  >
  > - **email** (*str*) – required
  > - **futuresType** (*int*) – required
  > - **startTime** (*int*) – optional
  > - **endTime** (*int*) – optional
  > - **page** (*int*) – optional
  > - **limit** (*int*) – optional
  > - **recvWindow** (*int*) – optional
  >
  > **Returns** API response

```
{
    "success":true,
    "futuresType": 2,
    "transfers":[
        {
            "from":"aaa@test.com",
            "to":"bbb@test.com",
            "asset":"BTC",
            "qty":"1",
            "time":1544433328000
        },
        {

            "from":"bbb@test.com",
            "to":"ccc@test.com",
            "asset":"ETH",
            "qty":"2",
            "time":1544433328000
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**`get_sub_account_list`**(*\*\*params*)
    Query Sub-account List.

    https://binance-docs.github.io/apidocs/spot/en/#query-sub-account-list-sapi-for-master-account

    > **Parameters**
    >
    > - **`email`** (*str*) – optional - Sub-account email
    >
    > - **`isFreeze`** (*str*) – optional
    >
    > - **`page`** (*int*) – optional - Default value: 1
    >
    > - **`limit`** (*int*) – optional - Default value: 1, Max value: 200
    >
    > - **`recvWindow`** (*int*) – optional
    >
    > **Returns** API response

```
{
    "subAccounts":[
        {
            "email":"testsub@gmail.com",
            "isFreeze":false,
            "createTime":1544433328000
        },
        {
            "email":"virtual@oxebmvfonoemail.com",
            "isFreeze":false,
            "createTime":1544433328000
        }
    ]
}
```

    > **Raises** BinanceRequestException, BinanceAPIException

**`get_sub_account_transfer_history`**(*\*\*params*)
    Query Sub-account Transfer History.

    https://binance-docs.github.io/apidocs/spot/en/#query-sub-account-spot-asset-transfer-history-sapi-for-master-account

    > **Parameters**
    >
    > - **`fromEmail`** (*str*) – optional
    >
    > - **`toEmail`** (*str*) – optional
    >
    > - **`startTime`** (*int*) – optional
    >
    > - **`endTime`** (*int*) – optional
    >
    > - **`page`** (*int*) – optional - Default value: 1
    >
    > - **`limit`** (*int*) – optional - Default value: 500
    >
    > - **`recvWindow`** (*int*) – optional
    >
    > **Returns** API response

```
[
    {
        "from":"aaa@test.com",
        "to":"bbb@test.com",
        "asset":"BTC",
```

(continues on next page)

```
        "qty":"10",
        "status": "SUCCESS",
        "tranId": 6489943656,
        "time":1544433328000
    },
    {
        "from":"bbb@test.com",
        "to":"ccc@test.com",
        "asset":"ETH",
        "qty":"2",
        "status": "SUCCESS",
        "tranId": 6489938713,
        "time":1544433328000
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_deposit_address**(*\*\*params*)
> Get Sub-account Deposit Address (For Master Account)
>
> https://binance-docs.github.io/apidocs/spot/en/#get-sub-account-deposit-address-for-master-account
>
> > **Parameters**
> >
> > - **email** (*str*) – required - Sub account email
> >
> > - **coin** (*str*) – required
> >
> > - **network** (*str*) – optional
> >
> > - **recvWindow** (*int*) – optional
> >
> > **Returns** API response

```
{
    "address":"TDunhSa7jkTNuKrusUTU1MUHtqXoBPKETV",
    "coin":"USDT",
    "tag":"",
    "url":"https://tronscan.org/#/address/TDunhSa7jkTNuKrusUTU1MUHtqXoBPKETV
↪"
 }
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_deposit_history**(*\*\*params*)
> Get Sub-account Deposit History (For Master Account)
>
> https://binance-docs.github.io/apidocs/spot/en/#get-sub-account-deposit-address-for-master-account
>
> > **Parameters**
> >
> > - **email** (*str*) – required - Sub account email
> >
> > - **coin** (*str*) – optional
> >
> > - **status** (*int*) – optional - (0:pending,6: credited but cannot withdraw, 1:success)
> >
> > - **startTime** (*int*) – optional
> >
> > - **endTime** (*int*) – optional

- **limit** (*int*) – optional

- **offset** (*int*) – optional - default:0

- **recvWindow** (*int*) – optional

**Returns** API response

```
[
    {
        "amount":"0.00999800",
        "coin":"PAXG",
        "network":"ETH",
        "status":1,
        "address":"0x788cabe9236ce061e5a892e1a59395a81fc8d62c",
        "addressTag":"",
        "txId":
↪"0xaad4654a3234aa6118af9b4b335f5ae81c360b2394721c019b5d1e75328b09f3",
        "insertTime":1599621997000,
        "transferType":0,
        "confirmTimes":"12/12"
    },
    {
        "amount":"0.50000000",
        "coin":"IOTA",
        "network":"IOTA",
        "status":1,
        "address":
↪"SIZ9VLMHWATXKV99LH99CIGFJFUMLEHGWVZVNNZXRJJVWBPHYWPPBOSDORZ9EQSHCZAMPVAPGFYQAUUV9DROOXJL
↪",
        "addressTag":"",
        "txId":
↪"ESBFVQUTPIWQNJSPXFNHNYHSQNTGKRVKPRABQWTAXCDWOAKDKYWPTVG9BGXNVNKTLEJGESAVXIKIZ9999
↪",
        "insertTime":1599620082000,
        "transferType":0,
        "confirmTimes":"1/1"
    }
]
```

**Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_futures_details**(*\*\*params*)

Get Detail on Sub-account's Futures Account (For Master Account)

https://binance-docs.github.io/apidocs/spot/en/#get-detail-on-sub-account-39-s-futures-account-for-master-account

**Parameters**

- **email** (*str*) – required - Sub account email

- **recvWindow** (*int*) – optional

**Returns** API response

```
{
    "email": "abc@test.com",
    "asset": "USDT",
    "assets":[
        {
```

```
            "asset": "USDT",
            "initialMargin": "0.00000000",
            "maintenanceMargin": "0.00000000",
            "marginBalance": "0.88308000",
            "maxWithdrawAmount": "0.88308000",
            "openOrderInitialMargin": "0.00000000",
            "positionInitialMargin": "0.00000000",
            "unrealizedProfit": "0.00000000",
            "walletBalance": "0.88308000"
        }
    ],
    "canDeposit": true,
    "canTrade": true,
    "canWithdraw": true,
    "feeTier": 2,
    "maxWithdrawAmount": "0.88308000",
    "totalInitialMargin": "0.00000000",
    "totalMaintenanceMargin": "0.00000000",
    "totalMarginBalance": "0.88308000",
    "totalOpenOrderInitialMargin": "0.00000000",
    "totalPositionInitialMargin": "0.00000000",
    "totalUnrealizedProfit": "0.00000000",
    "totalWalletBalance": "0.88308000",
    "updateTime": 1576756674610
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_futures_margin_status**(*\*\*params*)
> Get Sub-account's Status on Margin/Futures (For Master Account)

> https://binance-docs.github.io/apidocs/spot/en/#get-sub-account-39-s-status-on-margin-futures-for-master-account

> **Parameters**
> - **email** (*str*) – optional - Sub account email
> - **recvWindow** (*int*) – optional

> **Returns** API response

```
[
    {
        "email":"123@test.com",        // user email
        "isSubUserEnabled": true,      // true or false
        "isUserActive": true,          // true or false
        "insertTime": 1570791523523    // sub account create time
        "isMarginEnabled": true,       // true or false for margin
        "isFutureEnabled": true        // true or false for futures.
        "mobile": 1570791523523        // user mobile number
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_futures_positionrisk**(*\*\*params*)
> Get Futures Position-Risk of Sub-account (For Master Account)

> **Parameters**
>
> > - **email** (`str`) – required - Sub account email
> >
> > - **recvWindow** (`int`) – optional
>
> **Returns** API response

```
[
    {
        "entryPrice": "9975.12000",
        "leverage": "50",               // current initial leverage
        "maxNotional": "1000000",       // notional value limit of current
→initial leverage
        "liquidationPrice": "7963.54",
        "markPrice": "9973.50770517",
        "positionAmount": "0.010",
        "symbol": "BTCUSDT",
        "unrealizedProfit": "-0.01612295"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_futures_summary**(*\*\*params*)

> Get Summary of Sub-account's Futures Account (For Master Account)

> **Parameters** **recvWindow** (`int`) – optional
>
> **Returns** API response

```
{
    "totalInitialMargin": "9.83137400",
    "totalMaintenanceMargin": "0.41568700",
    "totalMarginBalance": "23.03235621",
    "totalOpenOrderInitialMargin": "9.00000000",
    "totalPositionInitialMargin": "0.83137400",
    "totalUnrealizedProfit": "0.03219710",
    "totalWalletBalance": "22.15879444",
    "asset": "USDT",
    "subAccountList":[
        {
            "email": "123@test.com",
            "totalInitialMargin": "9.00000000",
            "totalMaintenanceMargin": "0.00000000",
            "totalMarginBalance": "22.12659734",
            "totalOpenOrderInitialMargin": "9.00000000",
            "totalPositionInitialMargin": "0.00000000",
            "totalUnrealizedProfit": "0.00000000",
            "totalWalletBalance": "22.12659734",
            "asset": "USDT"
        },
        {
            "email": "345@test.com",
            "totalInitialMargin": "0.83137400",
            "totalMaintenanceMargin": "0.41568700",
```

```
            "totalMarginBalance": "0.90575887",
            "totalOpenOrderInitialMargin": "0.00000000",
            "totalPositionInitialMargin": "0.83137400",
            "totalUnrealizedProfit": "0.03219710",
            "totalWalletBalance": "0.87356177",
            "asset": "USDT"
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_subaccount_margin_details**(*\*\*params*)
> Get Detail on Sub-account's Margin Account (For Master Account)
>
> https://binance-docs.github.io/apidocs/spot/en/#get-detail-on-sub-account-39-s-margin-account-for-master-account
>
> > **Parameters**
> >
> > - **email** (*str*) – required - Sub account email
> >
> > - **recvWindow** (*int*) – optional
> >
> > **Returns** API response

```
{
    "email":"123@test.com",
    "marginLevel": "11.64405625",
    "totalAssetOfBtc": "6.82728457",
    "totalLiabilityOfBtc": "0.58633215",
    "totalNetAssetOfBtc": "6.24095242",
    "marginTradeCoeffVo":
        {
            "forceLiquidationBar": "1.10000000",   // Liquidation margin
→ratio
            "marginCallBar": "1.50000000",         // Margin call margin
→ratio
            "normalBar": "2.00000000"              // Initial margin ratio
        },
    "marginUserAssetVoList": [
        {
            "asset": "BTC",
            "borrowed": "0.00000000",
            "free": "0.00499500",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00499500"
        },
        {
            "asset": "BNB",
            "borrowed": "201.66666672",
            "free": "2346.50000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "2144.83333328"
        },
        {
            "asset": "ETH",
```

```
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000"
        },
        {
            "asset": "USDT",
            "borrowed": "0.00000000",
            "free": "0.00000000",
            "interest": "0.00000000",
            "locked": "0.00000000",
            "netAsset": "0.00000000"
        }
    ]
}
```

Raises BinanceRequestException, BinanceAPIException

**get_subaccount_margin_summary**(*\*\*params*)
Get Summary of Sub-account's Margin Account (For Master Account)

https://binance-docs.github.io/apidocs/spot/en/#get-summary-of-sub-account-39-s-margin-account-for-master-account

Parameters **recvWindow** (*int*) – optional

Returns API response

```
{
    "totalAssetOfBtc": "4.33333333",
    "totalLiabilityOfBtc": "2.11111112",
    "totalNetAssetOfBtc": "2.22222221",
    "subAccountList":[
        {
            "email":"123@test.com",
            "totalAssetOfBtc": "2.11111111",
            "totalLiabilityOfBtc": "1.11111111",
            "totalNetAssetOfBtc": "1.00000000"
        },
        {
            "email":"345@test.com",
            "totalAssetOfBtc": "2.22222222",
            "totalLiabilityOfBtc": "1.00000001",
            "totalNetAssetOfBtc": "1.22222221"
        }
    ]
}
```

Raises BinanceRequestException, BinanceAPIException

**get_subaccount_transfer_history**(*\*\*params*)
Sub-account Transfer History (For Sub-account)

https://binance-docs.github.io/apidocs/spot/en/#transfer-to-master-for-sub-account

Parameters

- **asset** (*str*) – required - The asset being transferred, e.g., USDT

- **type** (*int*) – optional - 1: transfer in, 2: transfer out

- **startTime** (*int*) – optional

- **endTime** (*int*) – optional

- **limit** (*int*) – optional - Default 500

- **recvWindow** (*int*) – optional

**Returns** API response

```
[
  {
    "counterParty":"master",
    "email":"master@test.com",
    "type":1,  // 1 for transfer in, 2 for transfer out
    "asset":"BTC",
    "qty":"1",
    "status":"SUCCESS",
    "tranId":11798835829,
    "time":1544433325000
  },
  {
    "counterParty":"subAccount",
    "email":"sub2@test.com",
    "type":2,
    "asset":"ETH",
    "qty":"2",
    "status":"SUCCESS",
    "tranId":11798829519,
    "time":1544433326000
  }
]
```

**Raises** BinanceRequestException, BinanceAPIException

**get_symbol_info**(*symbol*) → Optional[Dict[KT, VT]]
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** BinanceRequestException, BinanceAPIException

**get_symbol_ticker**(*\*\*params*)

> Latest price for a symbol or symbols.
>
> https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker
>
> > **Parameters symbol** (*str*) –
> >
> > **Returns** API response

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

> OR

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

> **Raises** BinanceRequestException, BinanceAPIException

**get_system_status**()

> Get system status detail.
>
> https://binance-docs.github.io/apidocs/spot/en/#system-status-sapi-system
>
> > **Returns** API response

```
{
    "status": 0,          # 0: normal1system maintenance
    "msg": "normal"       # normal or System maintenance.
}
```

> **Raises** BinanceAPIException

**get_ticker**(*\*\*params*)

24 hour price change statistics.

https://binance-docs.github.io/apidocs/spot/en/#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** BinanceRequestException, BinanceAPIException

**get_trade_fee**(*\*\*params*)

Get trade fee.

https://binance-docs.github.io/apidocs/spot/en/#trade-fee-sapi-user_data

> **Parameters**

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

> **Returns** API response

```
[
    {
        "symbol": "ADABNB",
        "makerCommission": "0.001",
        "takerCommission": "0.001"
    },
    {
        "symbol": "BNBBTC",
        "makerCommission": "0.001",
        "takerCommission": "0.001"
    }
]
```

**get_universal_transfer_history**(*\*\*params*)
    Universal Transfer (For Master Account)

    https://binance-docs.github.io/apidocs/spot/en/#query-universal-transfer-history

> **Parameters**
>
> - **fromEmail** (*str*) – optional
>
> - **toEmail** (*str*) – optional
>
> - **startTime** (*int*) – optional
>
> - **endTime** (*int*) – optional
>
> - **page** (*int*) – optional
>
> - **limit** (*int*) – optional
>
> - **recvWindow** (*int*) – optional

> **Returns** API response

```
[
  {
    "tranId":11945860693,
    "fromEmail":"master@test.com",
    "toEmail":"subaccount1@test.com",
    "asset":"BTC",
    "amount":"0.1",
    "fromAccountType":"SPOT",
    "toAccountType":"COIN_FUTURE",
    "status":"SUCCESS",
    "createTimeStamp":1544433325000
  },
  {
    "tranId":11945857955,
    "fromEmail":"master@test.com",
    "toEmail":"subaccount2@test.com",
    "asset":"ETH",
    "amount":"0.2",
    "fromAccountType":"SPOT",
    "toAccountType":"USDT_FUTURE",
```

```
        "status":"SUCCESS",
        "createTimeStamp":1544433326000
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_withdraw_history**(*\*\*params*)

> Fetch withdraw history.
>
> https://binance-docs.github.io/apidocs/spot/en/#withdraw-history-supporting-network-user_data
>
> **Parameters**
>
> - **coin** (*str*) – optional
> - **offset** (*int*) – optional - default:0
> - **limit** (*int*) – optional
> - **startTime** (*int*) – optional - Default: 90 days from current timestamp
> - **endTime** (*int*) – optional - Default: present timestamp
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for
>
> **Returns** API response

```
[
    {
        "address": "0x94df8b352de7f46f64b01d3666bf6e936e44ce60",
        "amount": "8.91000000",
        "applyTime": "2019-10-12 11:12:02",
        "coin": "USDT",
        "id": "b6ae22b3aa844210a7041aee7589627c",
        "withdrawOrderId": "WITHDRAWtest123", // will not be returned if␣
→there's no withdrawOrderId for this withdraw.
        "network": "ETH",
        "transferType": 0,   // 1 for internal transfer, 0 for external␣
→transfer
        "status": 6,
        "txId":
→"0xb5ef8c13b968a406cc62a93a8bd80f9e9a906ef1b3fcf20a2e48573c17659268"
    },
    {
        "address": "1FZdVHtiBqMrWdjPyRPULCUceZPJ2WLCsB",
        "amount": "0.00150000",
        "applyTime": "2019-09-24 12:43:45",
        "coin": "BTC",
        "id": "156ec387f49b41df8724fa744fa82719",
        "network": "BTC",
        "status": 6,
        "txId":
→"60fd9007ebfddc753455f95fafa808c4302c836e4d1eebc5a132c36c1d8ac354"
    }
]
```

> **Raises** BinanceRequestException, BinanceAPIException

**get_withdraw_history_id**(*withdraw_id*, *\*\*params*)
> Fetch withdraw history.

> https://binance-docs.github.io/apidocs/spot/en/#withdraw-history-supporting-network-user_data

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

> > **Returns** API response

```
{
    "id":"7213fea8e94b4a5593d507237e5a555b",
    "withdrawOrderId": None,
    "amount": 0.99,
    "transactionFee": 0.01,
    "address": "0x6915f16f8791d0a1cc2bf47c13a6b2a92000504b",
    "asset": "ETH",
    "txId":
↪"0xdf33b22bdb2b28b1f75ccd201a4a4m6e7g83jy5fc5d5a9d1340961598cfcb0a1",
    "applyTime": 1508198532000,
    "status": 4
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**isolated_margin_stream_close**(*symbol*, *listenKey*)
> Close out an isolated margin data stream.

> https://binance-docs.github.io/apidocs/spot/en/#listen-key-isolated-margin

> > **Parameters**
> > > - **symbol** (*str*) – required - symbol for the isolated margin account
> > > - **listenKey** (*str*) – required

> > **Returns** API response

```
{}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**isolated_margin_stream_get_listen_key**(*symbol*)
> Start a new isolated margin 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 stream alive.

> https://binance-docs.github.io/apidocs/spot/en/#listen-key-isolated-margin

> > **Parameters** **symbol** (*str*) – required - symbol for the isolated margin account

> > **Returns** API response

```
{
    "listenKey":
→"T3ee22BIYuWqmvne0HNq2A2WsFlEtLhvWCtItw6ffhhdmjifQ2tRbuKkTHhr"
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**isolated_margin_stream_keepalive**(*symbol*, *listenKey*)
> PING an isolated margin data stream to prevent a time out.
>
> https://binance-docs.github.io/apidocs/spot/en/#listen-key-isolated-margin
>
> > **Parameters**
> >
> > - **symbol** (*str*) – required - symbol for the isolated margin account
> >
> > - **listenKey** (*str*) – required
> >
> > **Returns** API response

```
{}
```

> **Raises** BinanceRequestException, BinanceAPIException

**make_subaccount_futures_transfer**(*\*\*params*)
> Futures Transfer for Sub-account (For Master Account)
>
> https://binance-docs.github.io/apidocs/spot/en/#futures-transfer-for-sub-account-for-master-account
>
> > **Parameters**
> >
> > - **email** (*str*) – required - Sub account email
> >
> > - **asset** (*str*) – required - The asset being transferred, e.g., USDT
> >
> > - **amount** (*float*) – required - The amount to be transferred
> >
> > - **type** (*int*) – required - 1: transfer from subaccount's spot account to its USDT-margined futures account 2: transfer from subaccount's USDT-margined futures account to its spot account 3: transfer from subaccount's spot account to its COIN-margined futures account 4: transfer from subaccount's COIN-margined futures account to its spot account
> >
> > **Returns** API response

```
{
    "txnId":"2966662589"
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**make_subaccount_margin_transfer**(*\*\*params*)
> Margin Transfer for Sub-account (For Master Account)
>
> https://binance-docs.github.io/apidocs/spot/en/#margin-transfer-for-sub-account-for-master-account
>
> > **Parameters**
> >
> > - **email** (*str*) – required - Sub account email
> >
> > - **asset** (*str*) – required - The asset being transferred, e.g., USDT

- **amount** (*float*) – required - The amount to be transferred

- **type** (*int*) – required - 1: transfer from subaccount's spot account to margin account 2: transfer from subaccount's margin account to its spot account

**Returns** API response

```
{
    "txnId":"2966662589"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**make_subaccount_to_master_transfer**(*\*\*params*)
  Transfer to Master (For Sub-account)

  https://binance-docs.github.io/apidocs/spot/en/#transfer-to-master-for-sub-account

  **Parameters**

- **asset** (*str*) – required - The asset being transferred, e.g., USDT

- **amount** (*float*) – required - The amount to be transferred

- **recvWindow** (*int*) – optional

  **Returns** API response

```
{
    "txnId":"2966662589"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**make_subaccount_to_subaccount_transfer**(*\*\*params*)
  Transfer to Sub-account of Same Master (For Sub-account)

  https://binance-docs.github.io/apidocs/spot/en/#transfer-to-sub-account-of-same-master-for-sub-account

  **Parameters**

- **toEmail** (*str*) – required - Sub account email

- **asset** (*str*) – required - The asset being transferred, e.g., USDT

- **amount** (*float*) – required - The amount to be transferred

- **recvWindow** (*int*) – optional

  **Returns** API response

```
{
    "txnId":"2966662589"
}
```

**Raises** BinanceRequestException, BinanceAPIException

**make_subaccount_universal_transfer**(*\*\*params*)
  Universal Transfer (For Master Account)

  https://binance-docs.github.io/apidocs/spot/en/#universal-transfer-for-master-account

> Parameters
>
> - **fromEmail** (*str*) – optional
>
> - **toEmail** (*str*) – optional
>
> - **fromAccountType** (*str*) – required - "SPOT","USDT_FUTURE","COIN_FUTURE"
>
> - **toAccountType** (*str*) – required - "SPOT","USDT_FUTURE","COIN_FUTURE"
>
> - **asset** (*str*) – required - The asset being transferred, e.g., USDT
>
> - **amount** (*float*) – required
>
> - **recvWindow** (*int*) – optional
>
> Returns  API response

```
{
    "tranId":11945860693
}
```

> Raises  BinanceRequestException, BinanceAPIException

**make_universal_transfer**(*\*\*params*)
> User Universal Transfer

> https://binance-docs.github.io/apidocs/spot/en/#user-universal-transfer

> Parameters
>
> - **type** (*str (ENUM)*) – required
>
> - **asset** (*str*) – required
>
> - **amount** (*str*) – required
>
> - **recvWindow** (*int*) – the number of milliseconds the request is valid for

```
transfer_status = client.make_universal_transfer(params)
```

> Returns  API response

```
{
    "tranId":13526853623
}
```

> Raises  BinanceRequestException, BinanceAPIException

**margin_stream_close**(*listenKey*)
> Close out a cross-margin data stream.

> https://binance-docs.github.io/apidocs/spot/en/#listen-key-margin

> Parameters  **listenKey** (*str*) – required

> Returns  API response

```
{}
```

> Raises  BinanceRequestException, BinanceAPIException

---

**margin_stream_get_listen_key**()
> Start a new cross-margin 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 stream alive.
>
> https://binance-docs.github.io/apidocs/spot/en/#listen-key-margin
>
> > **Returns** API response

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

> > **Raises** BinanceRequestException, BinanceAPIException

**margin_stream_keepalive**(*listenKey*)
> PING a cross-margin data stream to prevent a time out.
>
> https://binance-docs.github.io/apidocs/spot/en/#listen-key-margin
>
> > **Parameters** **listenKey** (*str*) – required
>
> > **Returns** API response

```
{}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**new_transfer_history**(*\*\*params*)
> Get future account transaction history list
>
> https://binance-docs.github.io/apidocs/delivery/en/#new-future-account-transfer

**options_account_info**(*\*\*params*)
> Account asset info (USER_DATA)
>
> https://binance-docs.github.io/apidocs/voptions/en/#account-asset-info-user_data
>
> > **Parameters** **recvWindow** (*int*) – optional

**options_bill**(*\*\*params*)
> Account funding flow (USER_DATA)
>
> https://binance-docs.github.io/apidocs/voptions/en/#account-funding-flow-user_data
>
> > **Parameters**
> >
> > - **currency** (*str*) – required - Asset type - USDT
> > - **recordId** (*int*) – optional - Return the recordId and subsequent data, the latest data is returned by default - 100000
> > - **startTime** (*int*) – optional - Start Time - 1593511200000
> > - **endTime** (*int*) – optional - End Time - 1593511200000
> > - **limit** (*int*) – optional - Number of result sets returned Default:100 Max:1000 - 100
> > - **recvWindow** (*int*) – optional

**options_cancel_all_orders**(*\*\*params*)
Cancel all Option orders (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#cancel-all-option-orders-trade

> Parameters

> > • **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

> > • **recvWindow** (*int*) – optional

**options_cancel_batch_order**(*\*\*params*)
Cancel Multiple Option orders (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#cancel-multiple-option-orders-trade

> Parameters

> > • **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

> > • **orderIds** – optional - Order ID - [4611875134427365377,4611875134427365378]

> > • **clientOrderIds** (*list*) – optional - User-defined order ID - ["my_id_1","my_id_2"]

> > • **recvWindow** (*int*) – optional

**options_cancel_order**(*\*\*params*)
Cancel Option order (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#cancel-option-order-trade

> Parameters

> > • **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

> > • **orderId** (*str*) – optional - Order ID - 4611875134427365377

> > • **clientOrderId** (*str*) – optional - User-defined order ID - 10000

> > • **recvWindow** (*int*) – optional

**options_exchange_info**()
Get current limit info and trading pair info

https://binance-docs.github.io/apidocs/voptions/en/#get-current-limit-info-and-trading-pair-info

**options_funds_transfer**(*\*\*params*)
Funds transfer (USER_DATA)

https://binance-docs.github.io/apidocs/voptions/en/#funds-transfer-user_data

> Parameters

> > • **currency** (*str*) – required - Asset type - USDT

> > • **type** (*str (ENUM)*) – required - IN: Transfer from spot account to option account OUT: Transfer from option account to spot account - IN

> > • **amount** (*float*) – required - Amount - 10000

> > • **recvWindow** (*int*) – optional

**options_historical_trades**(*\*\*params*)
Query trade history

https://binance-docs.github.io/apidocs/voptions/en/#query-trade-history

> Parameters

- **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

- **fromId** (*int*) – optional - The deal ID from which to return. The latest deal record is returned by default - 1592317127349

- **limit** (*int*) – optional - Number of records Default:100 Max:500 - 100

**options_index_price** (*\*\*params*)
    Get the spot index price

https://binance-docs.github.io/apidocs/voptions/en/#get-the-spot-index-price

**Parameters underlying** (*str*) – required - Spot pairOption contract underlying asset-BTCUSDT

**options_info** ()
    Get current trading pair info

https://binance-docs.github.io/apidocs/voptions/en/#get-current-trading-pair-info

**options_klines** (*\*\*params*)
    Candle data

https://binance-docs.github.io/apidocs/voptions/en/#candle-data

**Parameters**

- **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

- **interval** (*str*) – required - Time interval - 5m

- **startTime** (*int*) – optional - Start Time - 1592317127349

- **endTime** (*int*) – optional - End Time - 1592317127349

- **limit** (*int*) – optional - Number of records Default:500 Max:1500 - 500

**options_mark_price** (*\*\*params*)
    Get the latest mark price

https://binance-docs.github.io/apidocs/voptions/en/#get-the-latest-mark-price

**Parameters symbol** (*str*) – optional - Option trading pair - BTC-200730-9000-C

**options_order_book** (*\*\*params*)
    Depth information

https://binance-docs.github.io/apidocs/voptions/en/#depth-information

**Parameters**

- **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

- **limit** (*int*) – optional - Default:100 Max:1000.Optional value:[10, 20, 50, 100, 500, 1000] - 100

**options_ping** ()
    Test connectivity

https://binance-docs.github.io/apidocs/voptions/en/#test-connectivity

**options_place_batch_order** (*\*\*params*)
    Place Multiple Option orders (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#place-multiple-option-orders-trade

**Parameters**

- **orders** (*list*) – required - order list. Max 5 orders - [{"symbol":"BTC-210115-35000-C","price":"100","quantity":"0.0001","side":"BUY","type":"LIMIT"}]

- **recvWindow** (*int*) – optional

**options_place_order**(*\*\*params*)
> Option order (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#option-order-trade

> **Parameters**
> - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C
>
> - **side** (*str (ENUM)*) – required - Buy/sell direction: SELL, BUY - BUY
>
> - **type** (*str (ENUM)*) – required - Order Type: LIMIT, MARKET - LIMIT
>
> - **quantity** (*float*) – required - Order Quantity - 3
>
> - **price** (*float*) – optional - Order Price - 1000
>
> - **timeInForce** (*str (ENUM)*) – optional - Time in force methodDefault GTC) - GTC
>
> - **reduceOnly** (*bool*) – optional - Reduce Only (Default false) - false
>
> - **postOnly** (*bool*) – optional - Post Only (Default false) - false
>
> - **newOrderRespType** (*str (ENUM)*) – optional - "ACK", "RESULT", Default "ACK" - ACK
>
> - **clientOrderId** (*str*) – optional - User-defined order ID cannot be repeated in pending orders - 10000
>
> - **recvWindow** (*int*) – optional

**options_positions**(*\*\*params*)
> Option holdings info (USER_DATA)

https://binance-docs.github.io/apidocs/voptions/en/#option-holdings-info-user_data

> **Parameters**
> - **symbol** (*str*) – optional - Option trading pair - BTC-200730-9000-C
>
> - **recvWindow** (*int*) – optional

**options_price**(*\*\*params*)
> Get the latest price

https://binance-docs.github.io/apidocs/voptions/en/#get-the-latest-price

> **Parameters symbol** (*str*) – optional - Option trading pair - BTC-200730-9000-C

**options_query_order**(*\*\*params*)
> Query Option order (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#query-option-order-trade

> **Parameters**
> - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C
>
> - **orderId** (*str*) – optional - Order ID - 4611875134427365377
>
> - **clientOrderId** (*str*) – optional - User-defined order ID - 10000
>
> - **recvWindow** (*int*) – optional

**options_query_order_history**(*\*\*params*)
Query Option order history (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#query-option-order-history-trade

> **Parameters**
>
> > - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C
> >
> > - **orderId** (*str*) – optional - Returns the orderId and subsequent orders, the most recent order is returned by default - 100000
> >
> > - **startTime** (*int*) – optional - Start Time - 1593511200000
> >
> > - **endTime** (*int*) – optional - End Time - 1593511200000
> >
> > - **limit** (*int*) – optional - Number of result sets returned Default:100 Max:1000 - 100
> >
> > - **recvWindow** (*int*) – optional

**options_query_pending_orders**(*\*\*params*)
Query current pending Option orders (TRADE)

https://binance-docs.github.io/apidocs/voptions/en/#query-current-pending-option-orders-trade

> **Parameters**
>
> > - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C
> >
> > - **orderId** (*str*) – optional - Returns the orderId and subsequent orders, the most recent order is returned by default - 100000
> >
> > - **startTime** (*int*) – optional - Start Time - 1593511200000
> >
> > - **endTime** (*int*) – optional - End Time - 1593511200000
> >
> > - **limit** (*int*) – optional - Number of result sets returned Default:100 Max:1000 - 100
> >
> > - **recvWindow** (*int*) – optional

**options_recent_trades**(*\*\*params*)
Recently completed Option trades

https://binance-docs.github.io/apidocs/voptions/en/#recently-completed-option-trades

> **Parameters**
>
> > - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C
> >
> > - **limit** (*int*) – optional - Number of records Default:100 Max:500 - 100

**options_time**()
Get server time

https://binance-docs.github.io/apidocs/voptions/en/#get-server-time

**options_user_trades**(*\*\*params*)
Option Trade List (USER_DATA)

https://binance-docs.github.io/apidocs/voptions/en/#option-trade-list-user_data

> **Parameters**
>
> > - **symbol** (*str*) – required - Option trading pair - BTC-200730-9000-C

- **fromId** (*int*) – optional - Trade id to fetch from. Default gets most recent trades.
  - 4611875134427365376

- **startTime** (*int*) – optional - Start Time - 1593511200000

- **endTime** (*int*) – optional - End Time - 1593511200000

- **limit** (*int*) – optional - Number of result sets returned Default:100 Max:1000 -
  100

- **recvWindow** (*int*) – optional

**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** (*str*) – required
>
> - **quantity** (*decimal*) – required
>
> - **price** (*str*) – required
>
> - **timeInForce** (*str*) – 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** (*str*) – 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** BinanceRequestException, 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** (*str*) – 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** (*str*) – 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** BinanceRequestException, 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** (*str*) – 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** (*str*) – 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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_market**(*\*\*params*)
　　Send in a new market order

> **Parameters**

- **symbol** (*str*) – required

- **side** (*str*) – required

- **quantity** (*decimal*) – required

- **quoteOrderQty** (*decimal*) – amount the user wants to spend (when buying) or receive (when selling) of the quote asset

- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.

- **newOrderRespType** (*str*) – 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** BinanceRequestException, 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

- **quoteOrderQty** (*decimal*) – the amount the user wants to spend of the quote asset

- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.

- **newOrderRespType** (*str*) – 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** BinanceRequestException, 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

- **quoteOrderQty** (*decimal*) – the amount the user wants to receive of the quote asset

- **newClientOrderId** (*str*) – A unique id for the order. Automatically generated if not sent.

- **newOrderRespType** (*str*) – 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** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_oco_buy**(*\*\*params*)

> Send in a new OCO buy order

> **Parameters**
>
> - **symbol** (*str*) – required
> - **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.
> - **quantity** (*decimal*) – required
> - **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.
> - **price** (*str*) – required
> - **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.
> - **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated if not sent.
> - **stopPrice** (*str*) – required
> - **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.
> - **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.
> - **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.
> - **newOrderRespType** (*str*) – 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 OCO order endpoint for full response options

> **Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**order_oco_sell**(*\*\*params*)

> Send in a new OCO sell order

> **Parameters**
>
> - **symbol** (*str*) – required
> - **listClientOrderId** (*str*) – A unique id for the list order. Automatically generated if not sent.
> - **quantity** (*decimal*) – required
> - **limitClientOrderId** (*str*) – A unique id for the limit order. Automatically generated if not sent.

- **price** (*str*) – required

- **limitIcebergQty** (*decimal*) – Used to make the LIMIT_MAKER leg an iceberg order.

- **stopClientOrderId** (*str*) – A unique id for the stop order. Automatically generated if not sent.

- **stopPrice** (*str*) – required

- **stopLimitPrice** (*str*) – If provided, stopLimitTimeInForce is required.

- **stopIcebergQty** (*decimal*) – Used with STOP_LOSS_LIMIT leg to make an iceberg order.

- **stopLimitTimeInForce** (*str*) – Valid values are GTC/FOK/IOC.

- **newOrderRespType** (*str*) – 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 OCO order endpoint for full response options

**Raises** BinanceRequestException, BinanceAPIException, BinanceOrderException, BinanceOrderMinAmountException, BinanceOrderMinPriceException, BinanceOrderMinTotalException, BinanceOrderUnknownSymbolException, BinanceOrderInactiveSymbolException

**ping**() → Dict[KT, VT]
Test connectivity to the Rest API.

https://binance-docs.github.io/apidocs/spot/en/#test-connectivity

**Returns** Empty array

```
{}
```

**Raises** BinanceRequestException, BinanceAPIException

**purchase_lending_product**(*\*\*params*)
Purchase Flexible Product

https://binance-docs.github.io/apidocs/spot/en/#purchase-flexible-product-user_data

**query_subaccount_spot_summary**(*\*\*params*)
Query Sub-account Spot Assets Summary (For Master Account)

https://binance-docs.github.io/apidocs/spot/en/#query-sub-account-spot-assets-summary-for-master-account

**Parameters**

- **email** (*str*) – optional - Sub account email

- **page** (*int*) – optional - default 1

- **size** (*int*) – optional - default 10, max 20

- **recvWindow** (*int*) – optional

**Returns** API response

```
{
    "totalCount":2,
    "masterAccountTotalAsset": "0.23231201",
    "spotSubUserAssetBtcVoList":[
        {
            "email":"sub123@test.com",
            "totalAsset":"9999.00000000"
        },
        {
            "email":"test456@test.com",
            "totalAsset":"0.00000000"
        }
    ]
}
```

> **Raises** BinanceRequestException, BinanceAPIException

**query_universal_transfer_history**(*\*\*params*)
> Query User Universal Transfer History

> https://binance-docs.github.io/apidocs/spot/en/#query-user-universal-transfer-history

> **Parameters**

>> • **type** (*str (ENUM)*) – required

>> • **startTime** (*int*) – optional

>> • **endTime** (*int*) – optional

>> • **current** (*int*) – optional - Default 1

>> • **size** (*int*) – required - Default 10, Max 100

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

```
transfer_status = client.query_universal_transfer_history(params)
```

> **Returns** API response

```
{
    "total":2,
    "rows":[
        {
            "asset":"USDT",
            "amount":"1",
            "type":"MAIN_UMFUTURE"
            "status": "CONFIRMED",
            "tranId": 11415955596,
            "timestamp":1544433328000
        },
        {
            "asset":"USDT",
            "amount":"2",
            "type":"MAIN_UMFUTURE",
            "status": "CONFIRMED",
            "tranId": 11366865406,
            "timestamp":1544433328000
```

(continues on next page)

```
            }
        ]
}
```

>    **Raises** BinanceRequestException, BinanceAPIException

**redeem_lending_product**(*\*\*params*)
>    Redeem Flexible Product

>    https://binance-docs.github.io/apidocs/spot/en/#redeem-flexible-product-user_data

**repay_margin_loan**(*\*\*params*)
>    Repay loan in cross-margin or isolated-margin account.

>    If amount is more than the amount borrowed, the full loan will be repaid.

>    https://binance-docs.github.io/apidocs/spot/en/#margin-account-repay-margin

>    **Parameters**

>    - **asset** (*str*) – name of the asset

>    - **amount** (*str*) – amount to transfer

>    - **isIsolated** (*str*) – set to 'TRUE' for isolated margin (default 'FALSE')

>    - **symbol** (*str*) – Isolated margin symbol (default blank for cross-margin)

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

```
transaction = client.margin_repay_loan(asset='BTC', amount='1.1')

transaction = client.margin_repay_loan(asset='BTC', amount='1.1',
                                       isIsolated='TRUE', symbol='ETHBTC')
```

>    **Returns** API response

```
{
    "tranId": 100000001
}
```

>    **Raises** BinanceRequestException, BinanceAPIException

**stream_close**(*listenKey*)
>    Close out a user data stream.

>    https://binance-docs.github.io/apidocs/spot/en/#listen-key-spot

>    **Parameters** **listenKey** (*str*) – required

>    **Returns** API response

```
{}
```

>    **Raises** BinanceRequestException, 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://binance-docs.github.io/apidocs/spot/en/#listen-key-spot
>
> > **Returns** API response

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

> > **Raises** BinanceRequestException, BinanceAPIException

**stream_keepalive**(*listenKey*)
> PING a user data stream to prevent a time out.
>
> https://binance-docs.github.io/apidocs/spot/en/#listen-key-spot
>
> > **Parameters** **listenKey** (`str`) – required
> >
> > **Returns** API response

```
{}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**toggle_bnb_burn_spot_margin**(*\*\*params*)
> Toggle BNB Burn On Spot Trade And Margin Interest
>
> https://binance-docs.github.io/apidocs/spot/en/#toggle-bnb-burn-on-spot-trade-and-margin-interest-user_data
>
> > **Parameters**
> >
> > - **spotBNBBurn** (`bool`) – Determines whether to use BNB to pay for trading fees on SPOT
> >
> > - **interestBNBBurn** (`bool`) – Determines whether to use BNB to pay for margin loan's interest

```
response = client.toggle_bnb_burn_spot_margin()
```

> > **Returns** API response

```
{
   "spotBNBBurn":true,
   "interestBNBBurn": false
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**transfer_dust**(*\*\*params*)
> Convert dust assets to BNB.
>
> https://binance-docs.github.io/apidocs/spot/en/#dust-transfer-user_data

---

Parameters

- **asset** (*str*) – The asset being converted. e.g: 'ONE'

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

```
result = client.transfer_dust(asset='ONE')
```

**Returns** API response

```
{
    "totalServiceCharge":"0.02102542",
    "totalTransfered":"1.05127099",
    "transferResult":[
        {
            "amount":"0.03000000",
            "fromAsset":"ETH",
            "operateTime":1563368549307,
            "serviceChargeAmount":"0.00500000",
            "tranId":2970932918,
            "transferedAmount":"0.25000000"
        }
    ]
}
```

**Raises** BinanceRequestException, BinanceAPIException

**transfer_history**(*\*\*params*)

Get future account transaction history list

https://binance-docs.github.io/apidocs/futures/en/#get-future-account-transaction-history-list-user_data

**transfer_isolated_margin_to_spot**(*\*\*params*)

Execute transfer between isolated margin account and spot account.

https://binance-docs.github.io/apidocs/spot/en/#isolated-margin-account-transfer-margin

Parameters

- **asset** (*str*) – name of the asset

- **symbol** (*str*) – pair symbol

- **amount** (*str*) – amount to transfer

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

```
transfer = client.transfer_isolated_margin_to_spot(asset='BTC',
                                                    symbol='ETHBTC', amount=
→'1.1')
```

**Returns** API response

```
{
    "tranId": 100000001
}
```

**Raises** BinanceRequestException, BinanceAPIException

**transfer_margin_to_spot**(*\*\*params*)

> Execute transfer between cross-margin account and spot account.
>
> https://binance-docs.github.io/apidocs/spot/en/#cross-margin-account-transfer-margin
>
> > **Parameters**
> >
> > - **asset** (*str*) – name of the asset
> > - **amount** (*str*) – amount to transfer
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for

```
transfer = client.transfer_margin_to_spot(asset='BTC', amount='1.1')
```

> > **Returns** API response

```
{
    "tranId": 100000001
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**transfer_spot_to_isolated_margin**(*\*\*params*)

> Execute transfer between spot account and isolated margin account.
>
> https://binance-docs.github.io/apidocs/spot/en/#isolated-margin-account-transfer-margin
>
> > **Parameters**
> >
> > - **asset** (*str*) – name of the asset
> > - **symbol** (*str*) – pair symbol
> > - **amount** (*str*) – amount to transfer
> > - **recvWindow** (*int*) – the number of milliseconds the request is valid for

```
transfer = client.transfer_spot_to_isolated_margin(asset='BTC',
                                                    symbol='ETHBTC', amount=
→'1.1')
```

> > **Returns** API response

```
{
    "tranId": 100000001
}
```

> > **Raises** BinanceRequestException, BinanceAPIException

**transfer_spot_to_margin**(*\*\*params*)

> Execute transfer between spot account and cross-margin account.
>
> https://binance-docs.github.io/apidocs/spot/en/#cross-margin-account-transfer-margin
>
> > **Parameters**
> >
> > - **asset** (*str*) – name of the asset
> > - **amount** (*str*) – amount to transfer

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

```
transfer = client.transfer_spot_to_margin(asset='BTC', amount='1.1')
```

**Returns** API response

```
{
    "tranId": 100000001
}
```

**Raises** BinanceRequestException, BinanceAPIException

**universal_transfer**(*\*\*params*)

Unviversal transfer api accross different binance account types

https://binance-docs.github.io/apidocs/spot/en/#user-universal-transfer

**withdraw**(*\*\*params*)

Submit a withdraw request.

https://binance-docs.github.io/apidocs/spot/en/#withdraw-sapi

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**

- **coin** (*str*) – required
- **withdrawOrderId** (*str*) – optional - client id for withdraw
- **network** (*str*) – optional
- **address** (*str*) – optional
- **amount** (*decimal*) – required
- **transactionFeeFlag** (*bool*) – required - When making internal transfer, true for returning the fee to the destination account; false for returning the fee back to the departure account. Default false.
- **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

```
{
    "id":"7213fea8e94b4a5593d507237e5a555b"
}
```

**Raises** BinanceRequestException, BinanceAPIException

### depthcache module

**class** binance.depthcache.**BaseDepthCacheManager**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*)

> Bases: object
>
> **DEFAULT_REFRESH = 1800**
>
> **TIMEOUT = 60**
>
> **__init__**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*)
>
> > Create a DepthCacheManager instance
> >
> > > **Parameters**
> > >
> > > - **client** (*binance.Client*) – Binance API client
> > >
> > > - **loop** –
> > >
> > > - **symbol** (*string*) – Symbol to create depth cache for
> > >
> > > - **refresh_interval** (*int*) – Optional number of seconds between cache refresh, use 0 or None to disable
> > >
> > > - **bm** (*BinanceSocketManager*) – Optional BinanceSocketManager
> > >
> > > - **limit** (*int*) – Optional number of orders to get from orderbook
> > >
> > > - **conv_type** (*function.*) – Optional type to represent price, and amount, default is float.
>
> **close**()
>
> > Close the open socket for this manager
> >
> > > **Returns**
>
> **get_depth_cache**()
>
> > Get the current depth cache
> >
> > > **Returns** DepthCache object
>
> **get_symbol**()
>
> > Get the symbol
> >
> > > **Returns** symbol
>
> **recv**()

**class** binance.depthcache.**DepthCache**(*symbol*, *conv_type=<class 'float'>*)

> Bases: object
>
> **__init__**(*symbol*, *conv_type=<class 'float'>*)
>
> > Initialise the DepthCache
> >
> > > **Parameters**
> > >
> > > - **symbol** (*string*) – Symbol to create depth cache for
> > >
> > > - **conv_type** (*function.*) – Optional type to represent price, and amount, default is float.
>
> **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 conv_type.

```
[
    [
        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 conv_type

```
[
    [
        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*, *conv_type=<class 'float'>*)
    Sort bids or asks by price

**class** binance.depthcache.**DepthCacheManager**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=500*, *conv_type=<class 'float'>*, *ws_interval=None*)
    Bases: *binance.depthcache.BaseDepthCacheManager*

**__init__**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=500*, *conv_type=<class 'float'>*, *ws_interval=None*)
    Initialise the DepthCacheManager

    **Parameters**

    - **client** (*binance.Client*) – Binance API client

    - **loop** – asyncio loop

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

    - **refresh_interval** (*int*) – Optional number of seconds between cache refresh, use 0 or None to disable

    - **limit** (*int*) – Optional number of orders to get from orderbook

    - **conv_type** (*function.*) – Optional type to represent price, and amount, default is float.

    - **ws_interval** (*int*) – Optional interval for updates on websocket, default None. If not set, updates happen every second. Must be 0, None (1s) or 100 (100ms).

**class** binance.depthcache.**FuturesDepthCacheManager**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*)
    Bases: *binance.depthcache.BaseDepthCacheManager*

**class** binance.depthcache.**OptionsDepthCacheManager**(*client*, *symbol*, *loop=None*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*)
    Bases: *binance.depthcache.BaseDepthCacheManager*

**class** binance.depthcache.**ThreadedDepthCacheManager**(*api_key: Optional[str] = None*, *api_secret: Optional[str] = None*, *requests_params: Dict[str, str] = None*, *tld: str = 'com'*, *testnet: bool = False*)
    Bases: binance.threaded_stream.ThreadedApiManager

**__init__**(*api_key: Optional[str] = None*, *api_secret: Optional[str] = None*, *requests_params: Dict[str, str] = None*, *tld: str = 'com'*, *testnet: bool = False*)
    Initialise the BinanceSocketManager

**start_depth_cache**(*callback: Callable*, *symbol: str*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*, *ws_interval=0*) → str

**start_futures_depth_socket**(*callback: Callable*, *symbol: str*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*) → str

**start_options_depth_socket**(*callback: Callable*, *symbol: str*, *refresh_interval=None*, *bm=None*, *limit=10*, *conv_type=<class 'float'>*) → str

## exceptions module

**exception** binance.exceptions.**BinanceAPIException**(*response*, *status_code*, *text*)
  Bases: Exception

  **__init__**(*response*, *status_code*, *text*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderException**(*code*, *message*)
  Bases: Exception

  **__init__**(*code*, *message*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderInactiveSymbolException**(*value*)
  Bases: *binance.exceptions.BinanceOrderException*

  **__init__**(*value*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderMinAmountException**(*value*)
  Bases: *binance.exceptions.BinanceOrderException*

  **__init__**(*value*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderMinPriceException**(*value*)
  Bases: *binance.exceptions.BinanceOrderException*

  **__init__**(*value*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderMinTotalException**(*value*)
  Bases: *binance.exceptions.BinanceOrderException*

  **__init__**(*value*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceOrderUnknownSymbolException**(*value*)
  Bases: *binance.exceptions.BinanceOrderException*

  **__init__**(*value*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceRequestException**(*message*)
  Bases: Exception

  **__init__**(*message*)
    Initialize self. See help(type(self)) for accurate signature.

**exception** binance.exceptions.**BinanceWebsocketUnableToConnect**
  Bases: Exception

**exception** binance.exceptions.**NotImplementedException**(*value*)
  Bases: Exception

> **__init__** (*value*)
>> Initialize self. See help(type(self)) for accurate signature.

## helpers module

binance.helpers.**convert_ts_str** (*ts_str*)

binance.helpers.**date_to_milliseconds** (*date_str: str*) → int
> 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** – date in readable format, i.e. "January 01, 2018", "11 hours ago UTC", "now UTC"

binance.helpers.**interval_to_milliseconds** (*interval: str*) → Optional[int]
> Convert a Binance interval string to milliseconds
>> **Parameters interval** – Binance interval string, e.g.: 1m, 3m, 5m, 15m, 30m, 1h, 2h, 4h, 6h, 8h, 12h, 1d, 3d, 1w

>> **Returns** int value of interval in milliseconds None if interval prefix is not a decimal integer None if interval suffix is not one of m, h, d, w

binance.helpers.**round_step_size** (*quantity: Union[float, decimal.Decimal], step_size: Union[float, decimal.Decimal]*) → float
> Rounds a given quantity to a specific step size
>> **Parameters**

>>> • **quantity** – required

>>> • **step_size** – required

>> **Returns** decimal

## websockets module

**class** binance.streams.**BinanceSocketManager** (*client: binance.client.AsyncClient, loop=None, user_timeout=300*)
> Bases: object

> **DSTREAM_TESTNET_URL = 'wss://dstream.binancefuture.com/'**

> **DSTREAM_URL = 'wss://dstream.binance.{}/'**

> **FSTREAM_TESTNET_URL = 'wss://stream.binancefuture.com/'**

> **FSTREAM_URL = 'wss://fstream.binance.{}/'**

> **STREAM_TESTNET_URL = 'wss://testnet.binance.vision/'**

> **STREAM_URL = 'wss://stream.binance.{}:9443/'**

> **VSTREAM_TESTNET_URL = 'wss://testnetws.binanceops.{}/'**

> **VSTREAM_URL = 'wss://vstream.binance.{}/'**

> **WEBSOCKET_DEPTH_10 = '10'**

> **WEBSOCKET_DEPTH_20 = '20'**

> **WEBSOCKET_DEPTH_5 = '5'**

**__init__**(*client: binance.client.AsyncClient*, *loop=None*, *user_timeout=300*)
> Initialise the BinanceSocketManager

>> Parameters **client** (*binance.AsyncClient*) – Binance API client

**aggtrade_futures_socket**(*symbol: str*, *futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)
> Start a websocket for aggregate symbol trade data for the futures stream

>> **Parameters**
>>
>>> • **symbol** – required
>>>
>>> • **futures_type** – use USD-M or COIN-M futures default USD-M

>> **Returns** connection key string if successful, False otherwise

> Message Format

```
{
    "e": "aggTrade",  // Event type
    "E": 123456789,   // Event time
    "s": "BTCUSDT",    // Symbol
    "a": 5933014,     // Aggregate trade ID
    "p": "0.001",     // Price
    "q": "100",       // Quantity
    "f": 100,         // First trade ID
    "l": 105,         // Last trade ID
    "T": 123456785,   // Trade time
    "m": true,        // Is the buyer the market maker?
}
```

**aggtrade_socket**(*symbol: str*)
> 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

>> **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
}
```

**all_mark_price_socket**(*fast: bool = True*, *futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)
> Start a websocket for all futures mark price data By default all symbols are included in an array. https://binance-docs.github.io/apidocs/futures/en/#mark-price-stream-for-all-market :param fast: use faster or

1s default :param futures_type: use USD-M or COIN-M futures default USD-M :returns: connection key string if successful, False otherwise Message Format .. code-block:: python

> [
>> **{** "e": "markPriceUpdate", // Event type "E": 1562305380000, // Event time "s": "BT-CUSDT", // Symbol "p": "11185.87786614", // Mark price "r": "0.00030000", // Funding rate "T": 1562306400000 // Next funding time
>>
>> **}**
>
> ]

**all_ticker_futures_socket**(*futures_type:*     *binance.enums.FuturesType*    =    *<FuturesType.USD_M: 1>*)

Start a websocket for all ticker data By default all markets are included in an array. https://binance-docs.github.io/apidocs/futures/en/#all-book-tickers-stream :param futures_type: use USD-M or COIN-M futures default USD-M :returns: connection key string if successful, False otherwise Message Format .. code-block:: python

> [
>> **{** "u":400900217, // order book updateId "s":"BNBUSDT", // symbol "b":"25.35190000", // best bid price "B":"31.21000000", // best bid qty "a":"25.36520000", // best ask price "A":"40.66000000" // best ask qty
>>
>> **}**
>
> ]

**book_ticker_socket**()

Start a websocket for the best bid or ask's price or quantity for all symbols.

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

> **Returns** connection key string if successful, False otherwise

Message Format

```
{
    // Same as <symbol>@bookTicker payload
}
```

**coin_futures_socket**()

Start a websocket for coin futures data

> https://binance-docs.github.io/apidocs/delivery/en/#websocket-market-streams

> **Returns** connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

**depth_socket**(*symbol: str*, *depth: Optional[str] = None*, *interval: Optional[int] = 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

- **depth** (*str*) – optional Number of depth entries to return, default None. If passed returns a partial book instead of a diff

- **interval** (*int*) – optional interval for updates, default None. If not set, updates happen every second. Must be 0, None (1s) or 100 (100ms)

**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
        ]
    ]
}
```

**futures_depth_socket** (*symbol: str*, *depth: str = '10'*, *futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)
Subscribe to a depth data stream

https://binance-docs.github.io/apidocs/futures/en/#partial-book-depth-streams

**Parameters**

- **symbol** (*str*) – required

---

> • **depth** (*str*) – optional Number of depth entries to return, default 10.

**futures_multiplex_socket**(*streams: List[str], futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)

　　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 of stream names in lower case
> >
> > • **futures_type** – use USD-M or COIN-M futures default USD-M
>
> **Returns** connection key string if successful, False otherwise

　　Message Format - see Binance API docs for all types

**futures_socket**()

　　Start a websocket for futures data

> > https://binance-docs.github.io/apidocs/futures/en/#websocket-market-streams

> **Returns** connection key string if successful, False otherwise

　　Message Format - see Binance API docs for all types

**futures_user_socket**()

　　Start a websocket for coin futures user data

　　https://binance-docs.github.io/apidocs/futures/en/#user-data-streams

> **Returns** connection key string if successful, False otherwise

　　Message Format - see Binanace API docs for all types

**index_price_socket**(*symbol: str, fast: bool = True*)

　　Start a websocket for a symbol's futures mark price https://binance-docs.github.io/apidocs/delivery/en/#index-price-stream :param symbol: required :param fast: use faster or 1s default :returns: connection key string if successful, False otherwise

　　Message Format .. code-block:: python

> {
>
> > > "e": "indexPriceUpdate", // Event type "E": 1591261236000, // Event time
> > > "i": "BTCUSD", // Pair "p": "9636.57860000", // Index Price
> >
> > }

**individual_symbol_ticker_futures_socket**(*symbol: str, futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)

　　Start a futures websocket for a single symbol's ticker data https://binance-docs.github.io/apidocs/futures/en/#individual-symbol-ticker-streams :param symbol: required :type symbol: str :param futures_type: use USD-M or COIN-M futures default USD-M :returns: connection key string if successful, False otherwise .. code-block:: python

> { "e": "24hrTicker", // Event type "E": 123456789, // Event time "s": "BTCUSDT", // Symbol "p": "0.0015", // Price change
>
> }

---

**isolated_margin_socket**(*symbol: str*)
    Start a websocket for isolated margin data

    https://binance-docs.github.io/apidocs/spot/en/#listen-key-isolated-margin

        **Parameters symbol** (`str`) – required - symbol for the isolated margin account

        **Returns** connection key string if successful, False otherwise

    Message Format - see Binance API docs for all types

**kline_futures_socket**(*symbol: str, interval='1m', futures_type: binance.enums.FuturesType =
                <FuturesType.USD_M: 1>, contract_type: binance.enums.ContractType
                = <ContractType.PERPETUAL: 'perpetual'>*)
    Start a websocket for symbol kline data for the perpeual futures stream

    https://binance-docs.github.io/apidocs/futures/en/#continuous-contract-kline-candlestick-streams

        **Parameters**

            • **symbol** (`str`) – required

            • **interval** (`str`) – Kline interval, default KLINE_INTERVAL_1MINUTE

            • **futures_type** – use USD-M or COIN-M futures default USD-M

            • **contract_type** – use PERPETUAL or CURRENT_QUARTER or
              NEXT_QUARTER default PERPETUAL

        **Returns** connection key string if successful, False otherwise

    Message Format

```
    {
    "e":"continuous_kline",    // Event type
    "E":1607443058651,         // Event time
    "ps":"BTCUSDT",            // Pair
    "ct":"PERPETUAL"           // Contract type
    "k":{
        "t":1607443020000,     // Kline start time
        "T":1607443079999,     // Kline close time
        "i":"1m",              // Interval
        "f":116467658886,      // First trade ID
        "L":116468012423,      // Last trade ID
        "o":"18787.00",        // Open price
        "c":"18804.04",        // Close price
        "h":"18804.04",        // High price
        "l":"18786.54",        // Low price
        "v":"197.664",         // volume
        "n": 543,              // Number of trades
        "x":false,             // Is this kline closed?
        "q":"3715253.19494",   // Quote asset volume
        "V":"184.769",         // Taker buy volume
        "Q":"3472925.84746",   //Taker buy quote asset volume
        "B":"0"                // Ignore
    }
    }
    <pair>_<contractType>@continuousKline_<interval>
```

**kline_socket**(*symbol: str, 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
>
> - **interval** (*str*) – 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
        }
}
```

**margin_socket** ()
>    Start a websocket for cross-margin data

>    https://binance-docs.github.io/apidocs/spot/en/#listen-key-margin

>    **Returns** connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

**miniticker_socket** (*update_time: int = 1000*)
>    Start a miniticker websocket for all trades

>    This is not in the official Binance api docs, but this is what feeds the right column on a ticker page on Binance.

>    **Parameters** **update_time** (*int*) – time between callbacks in milliseconds, must be 1000 or greater

>    **Returns** connection key string if successful, False otherwise

Message Format

```
[
    {
        'e': '24hrMiniTicker',  # Event type
        'E': 1515906156273,     # Event time
        's': 'QTUMETH',         # Symbol
        'c': '0.03836900',      # close
        'o': '0.03953500',      # open
        'h': '0.04400000',      # high
        'l': '0.03756000',      # low
        'v': '147435.80000000', # volume
        'q': '5903.84338533'    # quote volume
    }
]
```

**multiplex_socket** (*streams: List[str]*)

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

> **Returns** connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

**options_depth_socket** (*symbol: str*, *depth: str = '10'*)

Subscribe to a depth data stream

https://binance-docs.github.io/apidocs/voptions/en/#market-streams-payload-depth

> **Parameters**
>
> - **symbol** (`str`) – required
>
> - **depth** (`str`) – optional Number of depth entries to return, default 10.

**options_kline_socket** (*symbol: str*, *interval='1m'*)

Subscribe to a candlestick data stream

https://binance-docs.github.io/apidocs/voptions/en/#market-streams-payload-candle

> **Parameters**
>
> - **symbol** (`str`) – required
>
> - **interval** (`str`) – Kline interval, default KLINE_INTERVAL_1MINUTE

**options_multiplex_socket** (*streams: List[str]*)

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://binance-docs.github.io/apidocs/voptions/en/#account-and-trading-interface

> **Parameters streams** (`list`) – list of stream names in lower case

> **Returns** connection key string if successful, False otherwise

Message Format - see Binance API docs for all types

**options_recent_trades_socket**(*symbol: str*)

    Subscribe to a latest completed trades stream

    https://binance-docs.github.io/apidocs/voptions/en/#market-streams-payload-latest-completed-trades

        **Parameters symbol** (`str`) – required

**options_ticker_socket**(*symbol: str*)

    Subscribe to a 24 hour ticker info stream

    https://binance-docs.github.io/apidocs/voptions/en/#market-streams-payload-24-hour-ticker

        **Parameters symbol** (`str`) – required

**symbol_book_ticker_socket**(*symbol: str*)

    Start a websocket for the best bid or ask's price or quantity for a specified symbol.

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

        **Parameters symbol** (`str`) – required

        **Returns** connection key string if successful, False otherwise

    Message Format

```
{
    "u":400900217,     // order book updateId
    "s":"BNBUSDT",     // symbol
    "b":"25.35190000", // best bid price
    "B":"31.21000000", // best bid qty
    "a":"25.36520000", // best ask price
    "A":"40.66000000"  // best ask qty
}
```

**symbol_mark_price_socket**(*symbol: str, fast: bool = True, futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*)

    Start a websocket for a symbol's futures mark price https://binance-docs.github.io/apidocs/futures/en/#mark-price-stream :param symbol: required :param fast: use faster or 1s default :param futures_type: use USD-M or COIN-M futures default USD-M :returns: connection key string if successful, False otherwise Message Format .. code-block:: python

        **{** "e": "markPriceUpdate", // Event type "E": 1562305380000, // Event time "s": "BT-CUSDT", // Symbol "p": "11185.87786614", // Mark price "r": "0.00030000", // Funding rate "T": 1562306400000 // Next funding time

        **}**

**symbol_miniticker_socket**(*symbol: str*)

    Start a websocket for a symbol's miniTicker data

    https://binance-docs.github.io/apidocs/spot/en/#individual-symbol-mini-ticker-stream

        **Parameters symbol** (`str`) – required

        **Returns** connection key string if successful, False otherwise

    Message Format

```
{
    "e": "24hrMiniTicker",  // Event type
    "E": 123456789,         // Event time
    "s": "BNBBTC",          // Symbol
```

```
    "c": "0.0025",          // Close price
    "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
}
```

**symbol_ticker_futures_socket**(*symbol: str*, *futures_type: binance.enums.FuturesType =
<FuturesType.USD_M: 1>*)

Start a websocket for a symbol's ticker data By default all markets are included in an array. https:
//binance-docs.github.io/apidocs/futures/en/#individual-symbol-book-ticker-streams :param symbol: re-
quired :param futures_type: use USD-M or COIN-M futures default USD-M :returns: connection key
string if successful, False otherwise .. code-block:: python

> [
>
>> { "u":400900217, // order book updateId "s":"BNBUSDT", // symbol
>>> "b":"25.35190000", // best bid price "B":"31.21000000", // best bid qty
>>> "a":"25.36520000", // best ask price "A":"40.66000000" // best ask qty
>>
>> }
>
> ]

**symbol_ticker_socket**(*symbol: str*)

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
>
> **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
}
```

**ticker_socket**()

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 coro** (*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'
    }
]
```

**trade_socket**(*symbol: str*)

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
>
> **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.
}
```

> **user_socket**()
> > Start a websocket for user data
> >
> > https://github.com/binance-exchange/binance-official-api-docs/blob/master/user-data-stream.md https://binance-docs.github.io/apidocs/spot/en/#listen-key-spot
> >
> > > **Returns** connection key string if successful, False otherwise
> >
> > Message Format - see Binance API docs for all types

**class** binance.streams.**BinanceSocketType**

> Bases: str, enum.Enum
>
> An enumeration.
>
> **ACCOUNT = 'Account'**
>
> **COIN_M_FUTURES = 'Coin_M_Futures'**
>
> **OPTIONS = 'Vanilla_Options'**
>
> **SPOT = 'Spot'**
>
> **USD_M_FUTURES = 'USD_M_Futures'**

**class** binance.streams.**KeepAliveWebsocket**(*client:   binance.client.AsyncClient,  loop,  url,
                                               keepalive_type,  prefix='ws/',  is_binary=False,
                                               exit_coro=None, user_timeout=None*)

> Bases: *binance.streams.ReconnectingWebsocket*
>
> **__init__**(*client: binance.client.AsyncClient, loop, url, keepalive_type, prefix='ws/', is_binary=False,
>            exit_coro=None, user_timeout=None*)
> > Initialize self. See help(type(self)) for accurate signature.

**class** binance.streams.**ReconnectingWebsocket**(*loop, url: str, path: Optional[str] = None,
                                                  prefix: str = 'ws/', is_binary: bool = False,
                                                  exit_coro=None*)

> Bases: object
>
> **MAX_QUEUE_SIZE = 100**
>
> **MAX_RECONNECTS = 5**
>
> **MAX_RECONNECT_SECONDS = 60**
>
> **MIN_RECONNECT_WAIT = 0.1**
>
> **NO_MESSAGE_RECONNECT_TIMEOUT = 60**
>
> **TIMEOUT = 10**

> **__init__** (*loop, url: str, path: Optional[str] = None, prefix: str = 'ws/', is_binary: bool = False,*
> *exit_coro=None*)
>> Initialize self. See help(type(self)) for accurate signature.

> **before_reconnect**()

> **connect**()

> **recv**()

**class** binance.streams.**ThreadedWebsocketManager**(*api_key:     Optional[str]   =   None,*
*api_secret:   Optional[str]   =   None,*
*requests_params: Dict[str, str] = None,*
*tld: str = 'com', testnet: bool = False*)
> Bases: binance.threaded_stream.ThreadedApiManager

> **__init__** (*api_key: Optional[str] = None, api_secret: Optional[str] = None, requests_params:*
> *Dict[str, str] = None, tld: str = 'com', testnet: bool = False*)
>> Initialise the BinanceSocketManager

> **start_aggtrade_futures_socket**(*callback:   Callable,  symbol:   str,  futures_type:   bi-*
> *nance.enums.FuturesType = <FuturesType.USD_M: 1>*)
> → str

> **start_aggtrade_socket**(*callback: Callable, symbol: str*) → str

> **start_all_mark_price_socket**(*callback:  Callable, fast:  bool = True, futures_type:  bi-*
> *nance.enums.FuturesType = <FuturesType.USD_M: 1>*) →
> str

> **start_all_ticker_futures_socket**(*callback:         Callable,       futures_type:       bi-*
> *nance.enums.FuturesType   =   <FuturesType.USD_M:*
> *1>*) → str

> **start_book_ticker_socket**(*callback: Callable*) → str

> **start_coin_futures_socket**(*callback: Callable*) → str

> **start_depth_socket**(*callback: Callable, symbol: str, depth: Optional[str] = None, interval: Op-*
> *tional[int] = None*) → str

> **start_futures_multiplex_socket**(*callback: Callable, streams: List[str], futures_type: bi-*
> *nance.enums.FuturesType = <FuturesType.USD_M: 1>*)
> → str

> **start_futures_socket**(*callback: Callable*) → str

> **start_index_price_socket**(*callback: Callable, symbol: str, fast: bool = True*) → str

> **start_individual_symbol_ticker_futures_socket**(*callback:        Callable,      sym-*
> *bol:   str,  futures_type:   bi-*
> *nance.enums.FuturesType      =*
> *<FuturesType.USD_M: 1>*) → str

> **start_isolated_margin_socket**(*callback: Callable, symbol: str*) → str

> **start_kline_futures_socket**(*callback: Callable, symbol: str, interval='1m', futures_type:*
> *binance.enums.FuturesType  =  <FuturesType.USD_M: 1>,*
> *contract_type:   binance.enums.ContractType = <Contract-*
> *Type.PERPETUAL: 'perpetual'>*) → str

> **start_kline_socket**(*callback: Callable, symbol: str, interval='1m'*) → str

> **start_margin_socket**(*callback: Callable*) → str

> **start_miniticker_socket**(*callback: Callable, update_time: int = 1000*) → str

**start_multiplex_socket**(*callback: Callable, streams: List[str]*) → str

**start_options_depth_socket**(*callback: Callable, symbol: str, depth: str = '10'*) → str

**start_options_kline_socket**(*callback: Callable, symbol: str, interval='1m'*) → str

**start_options_multiplex_socket**(*callback: Callable, streams: List[str]*) → str

**start_options_recent_trades_socket**(*callback: Callable, symbol: str*) → str

**start_options_ticker_socket**(*callback: Callable, symbol: str*) → str

**start_symbol_book_ticker_socket**(*callback: Callable, symbol: str*) → str

**start_symbol_mark_price_socket**(*callback: Callable, symbol: str, fast: bool = True, futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*) → str

**start_symbol_miniticker_socket**(*callback: Callable, symbol: str*) → str

**start_symbol_ticker_futures_socket**(*callback: Callable, symbol: str, futures_type: binance.enums.FuturesType = <FuturesType.USD_M: 1>*) → str

**start_symbol_ticker_socket**(*callback: Callable, symbol: str*) → str

**start_ticker_socket**(*callback: Callable*) → str

**start_trade_socket**(*callback: Callable, symbol: str*) → str

**start_user_socket**(*callback: Callable*) → str

**class** binance.streams.**WSListenerState**
Bases: enum.Enum

An enumeration.

**EXITING = 'Exiting'**

**INITIALISING = 'Initialising'**

**RECONNECTING = 'Reconnecting'**

**STREAMING = 'Streaming'**

binance.streams.**random**() → x in the interval [0, 1).

## 6.2 Index

- genindex

# Python Module Index

## b

## Symbols

## A

## V

## W