Namebase Exchange API Python Client

Release latest

Aug 14, 2020

Contents:

Python 3.6+ client for interacting with Namebase Exchange API. The goal of this project is to be the de facto Python client for working with the excellent Namebase Exchange API over its REST endpoints.

CHAPTER 1

Overview

1.1 Introduction

This library is intended to empower developers when interacting with the Namebase Exchange API by providing simple Python interfaces.

Namebase is the first Exchange for Handshake (HNS) and launched in Feb 2020 with the HNSBTC Trading pair.

The official Namebase Exchange API documentation can be found at https://github.com/namebasehq/exchange-api-documentation

1.2 API Requests

The Namebase Exchange API uses REST architecture to server data through its endpoints. The requests and responses of the endpoint use JSON format.

While the endpoint returns JSON this package turns the request into a Python dictionary for easier interoperability and function lookup.

All endpoints require authenticationt through the API Token.

1.3 Rate Limits

All endpoints are rate-limited by the Namebase team. Hard and fast rules are not given.

CHAPTER 2

Exchange Client - `exchange`

Table of Contents

Description: Implements Python client library for Namebase Exchange API. All calls require Authentication through a Bearer Token.

Usage: from namebase_exchange.exchange import *

class namebase_exchange.exchange.**Exchange**(*access_key: str, secret_key: str, api_root='https://www.namebase.io/api', api_version='/v0'*)

delete_order(*symbol: namebase_exchange.enums.Symbol, order_id: int, receive_window: Optional[int] = None*)

Function to cancel an active order. Execution of this function is as follows:

```
delete_order(symbol=Symbol, order_id=1)
```

The expected return result:

```
{
    "orderId": 28,
    "price": "1.00000000",
    "originalQuantity": "910.00000000",
    "executedQuantity": "19.00000000",
    "status": "CANCELED",
    "type": "LMT",
    "side": "SELL",
    "createdAt": 1555556529865,
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **order_id** (*int*) – The Order ID.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of Order Status.

generate_deposit_address(*asset: namebase_exchange.enums.Asset*, *receive_window: Optional[int] = None*)

Function to generate a deposit address. Execution of this function is as follows:

```
generate_deposit_address(asset=Symbol.HNS)
```

The expected return result:

```
{
    "address": "ts1qjg8chhk2t4zff4ltdaug3g9f7sxgne98jyv6ar",
    "success": true,
    "asset": "HNS"
}
```

Parameters

- **asset** (Asset) – The Trading Asset.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary

get_account_information(*receive_window: Optional[int] = None*)

Function to get basic account information. Execution of this function is as follows:

```
get_account_information()
```

The expected return result:

```
{
    "makerFee": 15, // in basis points, 0.15%
    "takerFee": 15, // in basis points, 0.15%
    "canTrade": true,
    "balances": [
        {
            "asset": "HNS",
            "unlocked": "779.900092",
            "lockedInOrders": "100.000000",
            "canDeposit": true,
            "canWithdraw": true
        },
        {
            "asset": "BTC",
            "unlocked": "5.10000012",
            "lockedInOrders": "1.000000",
            "canDeposit": true,
            "canWithdraw": true
        }
    ]
}
```

Parameters **receive_window** (*int*) – Optional Receive Window

Returns JSON Dictionary

get_account_limits(*receive_window: Optional[int] = None*)

Function to get your account's withdrawal limits for all assets. Withdrawal limits are applied on a 24-hour rolling basis. Start and End time is provided in the response startTime and endTime.

totalWithdrawn: how much asset has been withdrawn in past 24 hours. withdrawalLimit: how much can be withdrawn in the specified period.

Execution of this function is as follows:

```
get_account_limits()
```

The expected return result:

```
{
    "startTime": 1555467560001,
    "endTime": 1555553960000,
    "withdrawalLimits": [
        {
            "asset": "HNS",
            "totalWithdrawn": "500.000000",
            "withdrawalLimit": "10000.000000",
        },
        {
            "asset": "BTC",
            "totalWithdrawn": "0.50000000",
            "withdrawalLimit": "5.00000000",
        }
    ]
}
```

Parameters **receive_window** (*int*) – Optional Receive Window

Returns JSON Dictionary

get_account_trades(*symbol: namebase_exchange.enums.Symbol, trade_id: Optional[int], limit: int = 100, receive_window: Optional[int] = None*)

Function to get trades a specific account and symbol.

If trade_id is set, it will get trades >= trade_id. Otherwise you will get your most recent trades.

Execution of this function is as follows:

```
get_account_trades(symbol=Symbol.HNSBTC)
```

The expected return result:

```
[
    {
        "tradeId": 10921,
        "orderId": 61313,
        "price": "8.00000000",
        "quantity": "200.000000",
        "quoteQuantity": "1600.00000000",
        "commission": "4.500000",
        "commissionAsset": "HNS",
```

(continues on next page)

(continued from previous page)

```
        "createdAt": 1555556529865,
        "isBuyer": true,
        "isMaker": false,
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **trade_id** (*int*) – The Trade ID.
- **limit** (*int*) – Limit number of rows. Default 100.
- **receive_window** (*int*) – Optional Receive Window.

Returns List of Account Trades

get_all_orders(*symbol: namebase_exchange.enums.Symbol*, *order_id: Optional[int] = None*, *limit: int = 100*, *receive_window: Optional[int] = None*)

Function to get all account orders; active, cancelled, or filled. If order_id is provided, it will get orders >= order_id. Otherwise you will receive the most recent orders. Execution of this function is as follows:

```
get_all_orders(symbol=Symbol, limit=100)
```

The expected return result:

```
[
    {
        "orderId": 1,
        "price": "0.1",
        "originalQuantity": "1.0",
        "executedQuantity": "0.0",
        "status": "NEW",
        "type": "LMT",
        "side": "BUY",
        "createdAt": 1555556529865,
        "updatedAt": 1555556529865
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **order_id** (*int*) – The Order ID.
- **limit** (*int*) – Limit number of rows. Default 100.
- **receive_window** (*int*) – Optional Receive Window.

Returns List of All Orders.

get_deposit_history(*asset: namebase_exchange.enums.Asset*, *start_time: Optional[int] = None*, *end_time: Optional[int] = None*, *receive_window: Optional[int] = None*)

Function to get deposit history for an asset. Execution of this function is as follows:

```
get_deposit_history(asset=Symbol.HNS)
```

The expected return result:

```
[
    {
        "asset": "HNS",
        "amount": "31.853300",
        "address": "ts1qtq6ymgcep8mz2ag32ftrktwws0hr4uygprjurf",
        "txHash":
↪"e7714680a4d93e3b29348eab38c22bb99949ed4d8aea7006091ff5f9712d1ec6",
        "createdAt": 1555556529865,
    },
    {
        "asset": "HNS",
        "amount": "210.000333",
        "address": "n1M5Rw3r7WkujB2dG1L84M3a4pzr2NKvfp",
        "txHash":
↪"1d0827c642bd67781f80fe15c0fbb349aa4e35117adba06a52add4b207d334dd",
        "createdAt": 1555556529865,
    }
]
```

Parameters

- **asset** (Asset) – The Trading Asset.
- **start_time** (*int*) – The Start Time.
- **end_time** (*int*) – The End Time.
- **receive_window** (*int*) – Optional Receive Window

Returns List of Asset Deposits

get_depth (*symbol: namebase_exchange.enums.Symbol*, *limit: int = 100*) → dict

Function to get the Order Book Depth for a given Symbol. Execution of this function is as follows:

```
get_exchange_depth(symbol=Symbol.HNSBTC, limit=100)
```

The expected return result:

```
{
    "lastEventId": 6828,         // The last event id this includes
    "bids": [
        ["0.00003000",  "200.000000"] // [Price level, Quantity]
    ],
    "asks": [
        ["0.00003100", "100.000000"]
    ]
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **limit** (*int*) – The max number of rows to return, default 100.

Returns JSON Dictionary

get_dns_settings (*domain: str*)

Function to return the Handshake DNS settings for a domain. Execution of this function is as follows::

# https://github.com/namebasehq/api-documentation/blob/master/dns-settings-api.md
get_dns_settings(domain = 'test.testdomain')

The expected return result:

```
[
    {
        "success": boolean,
        "currentHeight": integer,
        "upToDate": boolean,
        "canUseSimpleUi": boolean, // false if the records were set using the␣
↪advanced settings
        "rawNameState": string, // hex of synced blockchain records
        "fee": string,
        "records": Record[],
    }
]
```

Parameters **domain** –

Returns List of records

get_exchange_info()

Function to fetch the Current Exchange trading rules and symbol information. Use to test connectivity to the Rest API. Execution of this function is as follows:

```
get_exchange_info()
```

The expected return result:

```
{
    "timezone": "UTC",
    "serverTime": 1555556529865,
    "symbols": [{
        "symbol": "HNSBTC",
        "status": "TRADING",
        "baseAsset": "HNS",
        "basePrecision": 6,
        "quoteAsset": "BTC",
        "quotePrecision": 8,
        "orderTypes": ["LMT", "MKT"]
    }]
}
```

get_kline(*symbol: namebase_exchange.enums.Symbol*, *interval: namebase_exchange.enums.Interval*, *start_time: Optional[int] = None*, *end_time: Optional[int] = None*, *limit: int = 100*)

Function to get Kline (candlestick) bars for a given symbol. Execution of this function is as follows:

```
get_kline(symbol=Symbol.HNSBTC, interval=Interval.ONE_HOUR, limit=100)
```

The expected return result:

```
[
    {
        "openTime": 1557190800000,
```

(continues on next page)

(continued from previous page)

```
        "closeTime": 1557190859999,
        "openPrice": "0.00002247",
        "highPrice": "0.00002256",
        "lowPrice": "0.00002243",
        "closePrice": "0.00002253",
        "volume": "10.001301",
        "quoteVolume": "0.000224824",
        "numberOfTrades": 42
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **interval** (Interval) – The Kline Interval.
- **start_time** (*int*) – Optional Start Time.
- **end_time** (*int*) – Optional End Time.
- **limit** (*int*) – Limit number of rows to be returned. Default 100.

Returns List of Candlestick Bars.

get_open_orders(*symbol: namebase_exchange.enums.Symbol*, *receive_window: Optional[int] = None*)

Function to get the most recent open orders on a symbol (limited to 500). Execution of this function is as follows:

```
get_open_orders(symbol=Symbol)
```

The expected return result:

```
[
    {
        "orderId": 1,
        "price": "0.1",
        "originalQuantity": "1.0",
        "executedQuantity": "0.0",
        "status": "NEW",
        "type": "LMT",
        "side": "BUY",
        "createdAt": 1555556529865,
        "updatedAt": 1555556529865
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary

get_order(*symbol: namebase_exchange.enums.Symbol*, *order_id: int*, *receive_window: Optional[int] = None*)

Function to get an order's status. Execution of this function is as follows:

```
get_order(symbol=Symbol, order_id=1)
```

The expected return result:

```
{
    "orderId": 1,
    "price": "0.1",
    "originalQuantity": "1.0",
    "executedQuantity": "0.0",
    "status": "NEW",
    "type": "LMT",
    "side": "BUY",
    "createdAt": 1555556529865,
    "updatedAt": 1555556529865
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **order_id** (*int*) – The Order ID.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of Order Information

get_order_trades(*symbol: namebase_exchange.enums.Symbol*, *order_id: int*, *receive_window: Optional[int] = None*)

Function to get trades a specific order and symbol. Execution of this function is as follows:

```
get_order_trades(symbol=Symbol.HNSBTC, order_id=61313)
```

The expected return result:

```
[
    {
        "tradeId": 10921,
        "orderId": 61313,
        "price": "8.00000000",
        "quantity": "200.000000",
        "quoteQuantity": "1600.00000000",
        "commission": "4.500000",
        "commissionAsset": "HNS",
        "createdAt": 1555556529865,
        "isBuyer": true,
        "isMaker": false,
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **order_id** (*int*) – The Order ID.
- **receive_window** (*int*) – Optional Receive Window.

Returns List of Trades

get_ticker_book(*symbol: namebase_exchange.enums.Symbol*)
Function to get best price/quantity on the order book for a symbol or symbols. Execution of this function is as follows:

```
get_ticker_book(symbol=Symbol.HNSBTC)
```

The expected return result:

```
{
    "bidPrice": "0.00002000",
    "bidQuantity": "100.000000",
    "askPrice": "0.00002300",
    "askQuantity": "9000.100000"
}
```

Parameters **symbol** (Symbol) – The Trading Symbol.

Returns JSON Dictionary

get_ticker_day(*symbol: namebase_exchange.enums.Symbol*)
Function to get 24 hour rolling window price change statistics. Execution of this function is as follows:

```
get_ticker_day(symbol=Symbol.HNSBTC)
```

The expected return result:

```
{
    "volumeWeightedAveragePrice": "0.00001959",
    "priceChange": "0.00000019",
    "priceChangePercent": "0.8528",
    "openPrice": "0.00002228",
    "highPrice": "0.00002247",
    "lowPrice": "0.00001414",
    "closePrice": "0.00002247",
    "volume": "11413.935399",
    "quoteVolume": "0.22363732",
    "openTime": 1555467560001,
    "closeTime": 1555553960000,
    "firstTradeId": 19761,
    "lastTradeId": 20926,
    "numberOfTrades": 1166
}
```

Parameters **symbol** (Symbol) – The Trading Symbol.

Returns JSON Dictionary

get_ticker_price(*symbol: namebase_exchange.enums.Symbol*)
Function to get latest price for a symbol or symbols. Execution of this function is as follows:

```
get_ticker_price(symbol=Symbol.HNSBTC)
```

The expected return result:

```
{
    "price": "0.00002300"
}
```

Parameters **symbol** (Symbol) – The Trading Symbol.

Returns JSON Dictionary

get_ticker_supply(*asset: namebase_exchange.enums.Asset*)

Function to get the circulating supply for the provided asset. Execution of this function is as follows:

```
get_ticker_supply(asset=Asset.HNS)
```

The expected return result:

```
{
    "height": 22012,
    "circulatingSupply": "116082412.354562",
}
```

Parameters **asset** (Asset) – Trading Asset.

Returns JSON Dictionary

get_trade(*symbol: namebase_exchange.enums.Symbol*, *trade_id: Optional[int] = None*, *limit: int = 100*, *receive_window: Optional[int] = None*) → dict

Function to get older trades. Execution of this function is as follows:

```
get_trade(symbol=Symbol.HNSBTC, trade_id=28457, limit=100)
```

The expected return result:

```
[
    {
        "tradeId": 28457,
        "price": "0.00003000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
        "createdAt": 1555556529865,
        "isBuyerMaker": true
    }
]
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **trade_id** (*int*) – The Trade ID (int) - not mandatory.
- **limit** (*int*) – Limit on number of rows to return - default 100.
- **receive_window** (*int*) – Receive Window - not mandatory.

Returns List of trades

get_withdraw_history(*asset: namebase_exchange.enums.Asset*, *start_time: Optional[int] = None*, *end_time: Optional[int] = None*, *receive_window: Optional[int] = None*)

Function to get withdraw history for an asset. Execution of this function is as follows:

```
get_withdraw_history(asset=Symbol.HNS)
```

The expected return result:

```
[
    {
        "id": "3333edc6-e5c6-4d23-bf84-7b1072a90e37",
        "asset": "HNS",
        "amount": "1.000000",
        "minerFee": "0.100000",
        "address": "ts1qtq6ymgcep8mz2ag32ftrktwws0hr4uygprjurf",
        "txHash":
↪"e7714680a4d93e3b29348eab38c22bb99949ed4d8aea7006091ff5f9712d1ec6",
        "createdAt": 1555556529865,
    },
    {
        "id": "180ceb4d-d303-4fed-9af6-213b5137255a",
        "asset": "HNS",
        "amount": "1200.000000",
        "minerFee": "0.200000",
        "address": "ts1qygv5nh38e9sl8npm4pcx8mqqqfp9sjaq4jrsn5",
        "txHash":
↪"c5c398802554b861bef2ec7c4805846ff400a90f71059619974685848bbc4fd3",
        "createdAt": 1555556529865,
    }
]
```

Parameters

- **asset** (Asset) – The Trading Asset.
- **start_time** (*int*) – The Start Time.
- **end_time** (*int*) – The End Time.
- **receive_window** (*int*) – Optional Receive Window.

Returns List of Withdraws

limit_buy (*symbol: namebase_exchange.enums.Symbol, price: str, quantity: str, receive_window: Optional[int]*)

Function to execute a Limit Buy. Execution of this function is as follows:

```
limit_buy(symbol=Symbol.HNSBTC, price='0.6',
quantity='1000.0')
```

The expected return result:

```
{
    "orderId": 174,
    "createdAt": 1555556529865,
    "price": "0.6",
    "originalQuantity": "1000.00000000",
    "executedQuantity": "1000.00000000",
    "status": "FILLED",
    "type": "LMT",
    "side": "BUY",
    "fills": [
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
```

(continues on next page)

(continued from previous page)


```
        "commission": "0.00000750",
        "commissionAsset": "BTC"
        },
        {
        "price": "0.6",
        "quantity": "500.000000",
        "quoteQuantity": "0.01000000",
        "commission": "0.00000500",
        "commissionAsset": "BTC"
        }
    ]
}
```


Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **quantity** (*str representation of decimal*) – The quantity of the base asset.
- **price** (*str representation of decimal*) – The price of the quote asset per 1 unit of base asset.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of immediate Order Status

limit_sell(*symbol: namebase_exchange.enums.Symbol, price: str, quantity: str, receive_window: Optional[int]*)

Function to execute a Limit Sell. Execution of this function is as follows:

```
limit_sell(symbol=Symbol.HNSBTC, price='0.6',
quantity='1000.0')
```

The expected return result:


```
{
    "orderId": 174,
    "createdAt": 1555556529865,
    "price": "0.6",
    "originalQuantity": "1000.00000000",
    "executedQuantity": "1000.00000000",
    "status": "FILLED",
    "type": "LMT",
    "side": "SELL",
    "fills": [
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
        "commission": "0.00000750",
        "commissionAsset": "BTC"
        },
        {
        "price": "0.6",
        "quantity": "500.000000",
        "quoteQuantity": "0.01000000",
        "commission": "0.00000500",
```


(continues on next page)

(continued from previous page)

```
        "commissionAsset": "BTC"
        }
    ]
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **quantity** (*str representation of decimal*) – The quantity of the base asset.
- **price** (*str representation of decimal*) – The price of the quote asset per 1 unit of base asset.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of immediate Order Status

market_buy(*symbol: namebase_exchange.enums.Symbol, quantity: str, receive_window: Optional[int]*)

Function to execute a Market Buy. Execution of this function is as follows:

```
market_buy(symbol=Symbol.HNSBTC, quantity='1000.0')
```

The expected return result:

```
{
    "orderId": 174,
    "createdAt": 1555556529865,
    "price": "0.0",
    "originalQuantity": "1000.00000000",
    "executedQuantity": "1000.00000000",
    "status": "FILLED",
    "type": "MKT",
    "side": "BUY",
    "fills": [
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
        "commission": "0.00000750",
        "commissionAsset": "BTC"
        },
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01000000",
        "commission": "0.00000500",
        "commissionAsset": "BTC"
        }
    ]
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **quantity** (*str representation of decimal*) – The quantity of the base asset.

- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of immediate Order Status

market_sell(*symbol: namebase_exchange.enums.Symbol, quantity: str, receive_window: Optional[int]*)

Function to execute a Market Sell. Execution of this function is as follows:

```
market_sell(symbol=Symbol.HNSBTC, quantity='1000.0')
```

The expected return result:

```
{
    "orderId": 174,
    "createdAt": 1555556529865,
    "price": "0.0",
    "originalQuantity": "1000.00000000",
    "executedQuantity": "1000.00000000",
    "status": "FILLED",
    "type": "MKT",
    "side": "SELL",
    "fills": [
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
        "commission": "0.00000750",
        "commissionAsset": "BTC"
        },
        {
        "price": "0.6000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01000000",
        "commission": "0.00000500",
        "commissionAsset": "BTC"
        }
    ]
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **quantity** (*str representation of decimal*) – The quantity of the base asset.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of immediate Order Status

new_order(*symbol: namebase_exchange.enums.Symbol, side: namebase_exchange.enums.OrderSide, order_type: namebase_exchange.enums.OrderType, quantity: str, price: Optional[str] = None, receive_window: Optional[int] = None*)

Function to send in a new order. Price is only required for Limit orders. Execution of this function is as follows:

```
new_order(symbol=Symbol.HNSBTC, side=OrderSide.SELL, type=OrderType.MARKET,
quantity='1000.0')
```

The expected return result:

```
{
    "orderId": 174,
    "createdAt": 1555556529865,
    "price": "0.00000000",
    "originalQuantity": "1000.00000000",
    "executedQuantity": "1000.00000000",
    "status": "FILLED",
    "type": "MKT",
    "side": "SELL",
    "fills": [
        {
        "price": "0.00003000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01500000",
        "commission": "0.00000750",
        "commissionAsset": "BTC"
        },
        {
        "price": "0.00002000",
        "quantity": "500.000000",
        "quoteQuantity": "0.01000000",
        "commission": "0.00000500",
        "commissionAsset": "BTC"
        }
    ]
}
```

Parameters

- **symbol** (Symbol) – The Trading Symbol.
- **side** (OrderSide) – The desired Order Side.
- **order_type** (OrderType) – The desired type of Order.
- **quantity** (*string representation of decimal*) – The quantity of the base asset.
- **price** (*string representation of decimal*) – The price of the quote asset per 1 unit of base asset. Only mandatory for LIMIT orders.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary of immediate Order Status

update_dns_settings(*domain: str*, *record_type: str = 'TXT'*, *value: str = ''*, *host: Optional[str] = None*, *ttl: Optional[int] = 0*) → dict

Function to updates the Handshake DNS settings for a domain. Execution of this function is as follows::

https://blog.sia.tech/skynet-handshake-d5d16e6b632f update_dns_settings(domain = 'test.testdomain', record_type = 'TXT',

value = 'AAApJJPnci_CzFnddB076HGu1_C64T6bfoiQqvsiVB5XeQ', host: '')

The expected return result:

```
[
    {
        "success": boolean,
        "txHash": string,
```

(continues on next page)

(continued from previous page)

```
        "rawNameState": string, // hex of synced blockchain records
        "records": Record[],
    }
]
```

Parameters

- **value** –
- **record_type** –
- **domain** –
- **host** –

Returns List of records

withdraw(*asset: namebase_exchange.enums.Asset*, *address: str*, *amount: float*, *receive_window: Optional[int] = None*)

Function to withdraw asset. Execution of this function is as follows:

```
withdraw(asset=Symbol.HNS, address=YOUR_ADDRESS, amount=1932.1)
```

The expected return result:

```
{
    "message": "success",
    "success": true,
    "id": "df7282ad-df8c-44f7-b747-5b09079ee852"
}
```

Parameters

- **asset** (Asset) – The Trading Asset.
- **address** (*str*) – The Address where the Asset is to be withdrawn.
- **amount** (*float*) – The amount of asset to be withdrawn.
- **receive_window** (*int*) – Optional Receive Window.

Returns JSON Dictionary

class namebase_exchange.exchange.**Symbol**

An enumeration.

HNSBTC = 'HNSBTC'

class namebase_exchange.exchange.**Asset**

An enumeration.

BTC = 'BTC'

HNS = 'HNS'

class namebase_exchange.exchange.**OrderType**

An enumeration.

LIMIT = 'LMT'

MARKET = 'MKT'

class namebase_exchange.exchange.**OrderSide**
An enumeration.

BUY = 'BUY'

SELL = 'SELL'

class namebase_exchange.exchange.**Interval**
An enumeration.

FIFTEEN_MINUTES = '15m'

FIVE_MINUTES = '5m'

FOUR_HOURS = '4h'

ONE_DAY = '1d'

ONE_HOUR = '1h'

ONE_MINUTE = '1m'

ONE_WEEK = '1w'

TWELVE_HOURS = '12h'

Python Module Index

n

Index

H

I

L

M

N

O

S

T

U

W