python-idex Documentation

Release 0.2.0

Sam McHardy

Aug 14, 2019

Contents

This is an unofficial Python wrapper for the IDEX exchanges REST API v1. I am in no way affiliated with IDEX, use at your own risk.

PyPi https://pypi.python.org/pypi/python-idex

Source code https://github.com/sammchardy/python-idex

Documentation https://python-idex.readthedocs.io/en/latest/

CHAPTER 1

Features

- Implementation of all REST endpoints except for deposit.
- Helper functions for your wallet address
- Response exception handling
- Websockets for Python 3.5+

CHAPTER 2

Quick Start

Register an account with IDEX.

```
pip install python-idex
```

CHAPTER 3

Synchronous Examples

```
api_key = 'api:jVXLd5h1bEYcKgZbQru2k'
address = '<address_string>'
private_key = '<private_key_string>'

from idex.client import Client
client = Client(api_key, address, private_key)

# get currencies
currencies = client.get_currencies()

# get market depth
depth = client.get_order_book('ETH_SENT')

# get your balances
balances = client.get_my_balances()

# get your open orders
orders = client.get_my_open_orders('ETH_SENT')

# create a limit order
order = client.create_order('SENT', 'ETH', '0.001', '10000')
```

CHAPTER 4

Async Examples for Python 3.5+

```
from idex.asyncio import AsyncClient, IdexSocketManager, SubscribeCategory

loop = None

async def main():
    global loop

    # Initialise the client
    client = await AsyncClient(api_key, address, private_key)

    # get currencies
    currencies = await client.get_currencies()

    # get market depth
    depth = await client.get_order_book('ETH_SENT')

    # get your balances
    balances = await client.get_my_balances()

    # get your open orders
    orders = await client.get_my_open_orders('ETH_SENT')

    # create a limit order
    order = await client.create_order('SENT', 'ETH', '0.001', '10000')

    # Coroutine to receive messages
    async def handle_evt(msg):
        print(f"event:{msg['event']} payload:{msg['payload']}")
        # do something with this event

    # Initialise the socket manager with the callback funciton
    ism = await IdexSocketManager.create(loop, handle_evt, api_key)

    # Subscribe to updates for the ETH_SENT, ETH_AURA and ETH_IDXM market for cancels,
↪ orders and trades
```

(continues on next page)

(continued from previous page)

```
    await ism.subscribe(
        SubscribeCategory.markets,
        ['ETH_SENT', 'ETH_AURA', 'ETH_IDXM'],
        ['market_cancels', 'market_orders', 'market_trades']
    )

    # keep the script running so we can retrieve websocket events
    while True:
        await asyncio.sleep(20, loop=loop)


if __name__ == "__main__":
    # get a loop and switch from synchronous to async
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

For more check out the documentation.

CHAPTER 5

TODO

- Deposit endpoints

CHAPTER 6

Donate

If this library helped you out feel free to donate.

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

CHAPTER 7

Other Exchanges

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

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 Quoinex or Qryptos check out my python-quoine library.

If you use Allcoin check out my python-allucoin library.

If you use Exx check out my python-exx library.

If you use BigONE check out my python-bigone library.

7.1 Contents

7.1.1 Getting Started

Installation

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

```
pip install python-idex
```

Register on IDEX

Firstly register an account with IDEX.

Make sure you save your private key as you will need it to sign trades.

Wallet Address

Your Wallet Address can be found in the top right under the account menu.

This is used to query the exchange for your balances, orders and trade history etc.

Some calls will throw an IdexException unless the wallet address and private key have been set.

Private Key

To perform and trading you will need both your wallet address and private key.

Note: Your private key is in the form '0x4efd9306gf134f9ee432d7415fb385029db50e7bce1682b2442beba24cf0a91f'

Initialise the client

Pass your Wallet Address and Private Key

```
from idex.client import Client
client = Client()

# add your wallet address later
client.set_wallet_address(address)

# change or add wallet address and private key
client.set_wallet_address(address, private_key)

# initialise the client with wallet address and private key
client = Client(address, private_key)
```

API Rate Limit

Unknown

Requests Settings

python-idex uses the requests library and the aiohttp library.

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

```
# for non-asyncio
client = Client(address, private_key, {"verify": False, "timeout": 20})

# for asyncio
client = Client(address, private_key, {"verify_ssl": 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.

```
# this would result in verify: False and timeout: 5 for the get_ticker call
client = Client(address, private_key, {"verify": False, "timeout": 20})
client.get_ticker('ETH_SAN', requests_params={'timeout': 5})
```

Check out the requests documentation for all options.

Proxy Settings

You can use the Requests Settings method above

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

# in the Client instantiation
client = Client(address, private_key, {'proxies': proxies})

# or on an individual call
client.get_ticker('ETH_SAN', 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
```

7.1.2 Currency Endpoints

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_currencies()

Get token data indexed by symbol

https://github.com/AuroraDAO/idex-api-docs#returncurrencies

```
currencies = client.get_currencies()
```

Returns API Response

```
{
    ETH: {
        decimals: 18,
        address: '0x0000000000000000000000000000000000000000',
        name: 'Ether'
    },
    REP: {
        decimals: 8,
        address: '0xc853ba17650d32daba343294998ea4e33e7a48b9',
        name: 'Reputation'
    },
    DVIP: {
        decimals: 8,
        address: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
```

(continues on next page)

(continued from previous page)

```
        name: 'Aurora'
    }
}
```

Raises IdexResponseException, IdexAPIException

get_currency(*currency*)

Get the details for a particular currency using it's token name or address

Parameters **currency** (*string or hex string*) – Name of the currency e.g. EOS or '0x7c5a0ce9267ed19b22f8cae653f198e3e8daf098'

```
# using token name
currency = client.get_currency('REP')

# using the address string
currency = client.get_currency('0xc853ba17650d32daba343294998ea4e33e7a48b9')
```

Returns

```
{
    decimals: 8,
    address: '0xc853ba17650d32daba343294998ea4e33e7a48b9',
    name: 'Reputation'
}
```

Raises IdexCurrencyNotFoundException, IdexResponseException, IdexAPIException

7.1.3 Market Endpoints

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_tickers()

Get all market tickers

Please note: If any field is unavailable due to a lack of trade history or a lack of 24hr data, the field will be set to 'N/A'. percentChange, baseVolume, and quoteVolume will never be 'N/A' but may be 0.

https://github.com/AuroraDAO/idex-api-docs#returnticker

```
tickers = client.get_tickers()
```

Returns API Response

```
{
    ETH_SAN:  {
        last: '0.000981',
        high: '0.0010763',
        low: '0.0009777',
        lowestAsk: '0.00098151',
        highestBid: '0.0007853',
        percentChange: '-1.83619353',
```

(continues on next page)

(continued from previous page)

```
        baseVolume: '7.3922603247161',
        quoteVolume: '7462.998433'
    },
    ETH_LINK: {
        last: '0.001',
        high: '0.0014',
        low: '0.001',
        lowestAsk: '0.002',
        highestBid: '0.001',
        percentChange: '-28.57142857',
        baseVolume: '13.651606265667369466',
        quoteVolume: '9765.891979953083752189'
    }
    # all possible markets follow ...
}
```

Raises IdexResponseException, IdexAPIException

get_ticker(*market*)

Get ticker for selected market

Please note: If any field is unavailable due to a lack of trade history or a lack of 24hr data, the field will be set to 'N/A'. percentChange, baseVolume, and quoteVolume will never be 'N/A' but may be 0.

https://github.com/AuroraDAO/idex-api-docs#returnticker

Parameters **market** (*string*) – Name of market e.g. ETH_SAN

```
ticker = client.get_ticker('ETH_SAN')
```

Returns API Response

```
{
    last: '0.000981',
    high: '0.0010763',
    low: '0.0009777',
    lowestAsk: '0.00098151',
    highestBid: '0.0007853',
    percentChange: '-1.83619353',
    baseVolume: '7.3922603247161',
    quoteVolume: '7462.998433'
}
```

Raises IdexResponseException, IdexAPIException

get_24hr_volume()

Get all market tickers

https://github.com/AuroraDAO/idex-api-docs#return24volume

```
volume = client.get_24hr_volume()
```

Returns API Response

```
{
    ETH_REP: {
        ETH: '1.3429046745',
        REP: '105.29046745'
    },
    ETH_DVIP: {
        ETH: '4',
        DVIP: '4'
    },
    totalETH: '5.3429046745'
}
```

Raises IdexResponseException, IdexAPIException

get_order_book(*market*, *count=1*)

Get order book for selected market

Each market returned will have an asks and bids property containing all the sell orders and buy orders sorted by best price. Order objects will contain a price amount total and orderHash property but also a params property which will contain additional data about the order useful for filling or verifying it.

https://github.com/AuroraDAO/idex-api-docs#returnorderbook

Parameters

- **market** (*string*) – Name of market e.g. ETH_SAN
- **count** (*int*) – Number of items to return

```
orderbook = client.get_order_book('ETH_SAN')
```

Returns API Response

```
{
    asks: [
        {
            price: '2',
            amount: '1',
            total: '2',
            orderHash:
↪'0x6aee6591def621a435dd86eafa32dfc534d4baa38d715988d6f23f3e2f20a29a',
            params: {
                tokenBuy: '0x0000000000000000000000000000000000000000',
                buySymbol: 'ETH',
                buyPrecision: 18,
                amountBuy: '2000000000000000000',
                tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
                sellSymbol: 'DVIP',
                sellPrecision: 8,
                amountSell: '100000000',
                expires: 190000,
                nonce: 164,
                user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
            }
        }
    ],
    bids: [
```

(continues on next page)

(continued from previous page)

```
        {
            price: '1',
            amount: '2',
            total: '2',
            orderHash:
↪'0x9ba97cfc6d8e0f9a72e9d26c377be6632f79eaf4d87ac52a2b3d715003b6536e',
            params: {
                tokenBuy: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
                buySymbol: 'DVIP',
                buyPrecision: 8,
                amountBuy: '200000000',
                tokenSell: '0x0000000000000000000000000000000000000000',
                sellSymbol: 'ETH',
                sellPrecision: 18,
                amountSell: '2000000000000000000',
                expires: 190000,
                nonce: 151,
                user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
            }
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

7.1.4 Exchange Endpoints

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_contract_address()

Get the contract address used for depositing, withdrawing, and posting orders

https://github.com/AuroraDAO/idex-api-docs#returncontractaddress

```
trades = client.get_contract_address()
```

Returns API Response

```
{
    address: '0x2a0c0dbecc7e4d658f48e01e3fa353f44050c208'
}
```

Raises IdexResponseException, IdexAPIException

7.1.5 Order Endpoints

These functions use the wallet address passed in the constructor.

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_order_trades(*order_hash*)

Get all trades involving a given order hash, specified by the order_hash

https://github.com/AuroraDAO/idex-api-docs#returnordertrades

Parameters **order_hash** (*256-bit hex string*) – The order hash to query for associated trades

```
trades = client.get_order_trades(
↪'0x62748b55e1106f3f453d51f9b95282593ef5ce03c22f3235536cf63a1476d5e4')
```

Returns API Response

```
[
    {
        date: '2017-10-11 21:41:15',
        amount: '0.3',
        type: 'buy',
        total: '1',
        price: '0.3',
        uuid: 'e8719a10-aecc-11e7-9535-3b8451fd4699',
        transactionHash:
↪'0x28b945b586a5929c69337929533e04794d488c2d6e1122b7b915705d0dff8bb6'
    }
]
```

Raises IdexResponseException, IdexAPIException

These functions take an address, typically you would only use them to fetch from your own address.

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_open_orders(*market*, *address*, *count=10*, *cursor=None*)

Get the open orders for a given market and address

Output is similar to the output for get_order_book() except that orders are not sorted by type or price, but are rather displayed in the order of insertion. As is the case with get_order_book(there is a params property of the response value that contains details on the order which can help with verifying its authenticity.

https://github.com/AuroraDAO/idex-api-docs#returnopenorders

Parameters

- **market** (*string*) – Name of market e.g. ETH_SAN
- **address** (*address string*) – Address to return open orders associated with
- **count** (*int*) – amount of results to return
- **cursor** (*str*) – For pagination. Provide the value returned in the idex-next-cursor HTTP header to request the next slice (or page)

```
orders = client.get_open_orders(
    'ETH_SAN',
    '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
[
    {
        orderNumber: 1412,
        orderHash:
↪'0xf1bbc500af8d411b0096ac62bc9b60e97024ad8b9ea170340ff0ecfa03536417',
        price: '2.3',
        amount: '1.2',
        total: '2.76',
        type: 'sell',
        params: {
            tokenBuy: '0x0000000000000000000000000000000000000000',
            buySymbol: 'ETH',
            buyPrecision: 18,
            amountBuy: '2760000000000000000',
            tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
            sellSymbol: 'DVIP',
            sellPrecision: 8,
            amountSell: '120000000',
            expires: 190000,
            nonce: 166,
            user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
        }
    },
    {
        orderNumber: 1413,
        orderHash:
↪'0x62748b55e1106f3f453d51f9b95282593ef5ce03c22f3235536cf63a1476d5e4',
        price: '2.98',
        amount: '1.2',
        total: '3.576',
        type: 'sell',
        params:{
            tokenBuy: '0x0000000000000000000000000000000000000000',
            buySymbol: 'ETH',
            buyPrecision: 18,
            amountBuy: '3576000000000000000',
            tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
            sellSymbol: 'DVIP',
            sellPrecision: 8,
            amountSell: '120000000',
            expires: 190000,
            nonce: 168,
            user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
        }
    }
]
```

Raises IdexResponseException, IdexAPIException

get_trade_history(*market=None*, *address=None*, *start=None*, *end=None*, *count=10*, *sort='desc'*, *cursor=None*)

Get the past 200 trades for a given market and address, or up to 10000 trades between a range specified in UNIX timetsamps by the "start" and "end" properties of your JSON input.

https://github.com/AuroraDAO/idex-api-docs#returntradehistory

Parameters

- **market** (*string*) – optional - will return an array of trade objects for the market, if

omitted, will return an object of arrays of trade objects keyed by each market

- **address** (*address string*) – optional - If specified, return value will only include trades that involve the address as the maker or taker.
- **start** (*int*) – optional - The inclusive UNIX timestamp (seconds since epoch) marking the earliest trade that will be returned in the response, (Default - 0)
- **end** (*int*) – optional - The inclusive UNIX timestamp marking the latest trade that will be returned in the response. (Default - current timestamp)
- **count** (*int*) – optional - Number of records to be returned per request. Valid range: 1 .. 100
- **sort** (*string*) – optional - Possible values are asc (oldest first) and desc (newest first). Defaults to desc.
- **cursor** (*string*) – optional - For pagination. Provide the value returned in the idex-next-cursor HTTP header to request the next slice (or page). This endpoint uses the tid property of a record for the cursor.

```
trades = client.get_trade_history()

# get trades for the last 2 hours for ETH EOS market
start = int(time.time()) - (60 * 2) # 2 hours ago
trades = client.get_trade_history(market='ETH_EOS', start=start)
```

Returns API Response

```
{
    ETH_REP: [
        {
            date: '2017-10-11 21:41:15',
            amount: '0.3',
            type: 'buy',
            total: '1',
            price: '0.3',
            orderHash:
↪'0x600c405c44d30086771ac0bd9b455de08813127ff0c56017202c95df190169ae',
            uuid: 'e8719a10-aecc-11e7-9535-3b8451fd4699',
            transactionHash:
↪'0x28b945b586a5929c69337929533e04794d488c2d6e1122b7b915705d0dff8bb6'
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

7.1.6 Account Endpoints

These functions use the wallet address passed in the constructor.

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

These functions take an address, typically it's simpler to use the above functions.

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

get_balances(*address*, *complete=False*)

Get available balances for an address (total deposited minus amount in open orders) indexed by token symbol.

https://github.com/AuroraDAO/idex-api-docs#returnbalances

Parameters

- **address** (*address string*) – Address to query balances of
- **complete** – Include available balances along with the amount you have in open orders for each token (Default False)
- **complete** – bool

```
balances = client.get_balances('0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
# Without complete details
{
    REP: '25.55306545',
    DVIP: '200000000.31012358'
}

# With complete details
{
    REP: {
        available: '25.55306545',
        onOrders: '0'
    },
    DVIP: {
        available: '200000000.31012358',
        onOrders: '0'
    }
}
```

Raises IdexResponseException, IdexAPIException

get_transfers(*address*, *start=None*, *end=None*)

Returns the deposit and withdrawal history for an address within a range, specified by the "start" and "end" properties of the JSON input, both of which must be UNIX timestamps. Withdrawals can be marked as "PENDING" if they are queued for dispatch, "PROCESSING" if the transaction has been dispatched, and "COMPLETE" if the transaction has been mined.

https://github.com/AuroraDAO/idex-api-docs#returndepositswithdrawals

Parameters

- **address** (*address string*) – Address to query deposit/withdrawal history for
- **start** (*int*) – optional - Inclusive starting UNIX timestamp of returned results (Default - 0)
- **end** (*int*) – optional - Inclusive ending UNIX timestamp of returned results (Default - current timestamp)

```
transfers = client.get_transfers('0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
{
    deposits: [
        {
            depositNumber: 265,
            currency: 'ETH',
            amount: '4.5',
            timestamp: 1506550595,
            transactionHash:
↪'0x52897291dba0a7b255ee7a27a8ca44a9e8d6919ca14f917616444bf974c48897'
        }
    ],
    withdrawals: [
        {
            withdrawalNumber: 174,
            currency: 'ETH',
            amount: '4.5',
            timestamp: 1506552152,
            transactionHash:
↪'0xe52e9c569fe659556d1e56d8cca2084db0b452cd889f55ec3b4e2f3af61faa57',
            status: 'COMPLETE'
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

get_next_nonce(*address*)

Get the lowest nonce that you can use from the given address in one of the trade functions

https://github.com/AuroraDAO/idex-api-docs#returnnextnonce

Parameters **address** (*address string*) – The address to query for the next nonce to use

```
nonce = client.get_next_nonce('0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c')
```

Returns API Response

```
{
    nonce: 2650
}
```

Raises IdexResponseException, IdexAPIException

7.1.7 Withdraw Endpoints

These functions use the wallet address passed in the constructor.

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

7.1.8 Async

In v0.3.0 async functionality was added for all REST API methods. This is only supported on Python 3.5+

Example

7.1.9 Websockets

Websockets are only available for Python 3.5+.

The IDEX Websocket documentation describes the type of messages that can be returned from the websocket.

These include

- orders added to the order book
- orders removed from the order book
- orders modified in the order book
- new trades

So far only new trade events are received.

Example

7.1.10 Exceptions

IdexWalletAddressNotFoundException

Raised if a wallet address has not been set.

IdexPrivateKeyNotFoundException

Raised if the private key has not been set.

IdexCurrencyNotFoundException

Raised if a requested currency is not found.

IdexResponseException

Raised if a non JSON response is returned.

IdexAPIException

On an API call error a idex.exceptions.IdexAPIException will be raised.

The exception provides access to the

- *status_code* - response status code
- *response* - response object
- *message* - IDEX error message
- *request* - request object if available

```
try:
    client.get_currencies()
except IdexAPIException as e:
    print(e.status_code)
    print(e.message)
```

7.1.11 Changelog

v1.0.0 - 2019-08-14

Breaking Change

- added mandatory api_key parameter to Client, AsyncClient and IdexSocketManager
- remove deprecated *get_order_books* function, use *get_order_book* with market param instead

v0.3.6 - 2019-06-29

Updated

- update rlp dependency for python 3

v0.3.5 - 2019-02-11

Updated

- websockets to use new datastream

v0.3.4 - 2019-01-15

Added

- new endpoint *get_order_status*
- new params for *get_order_book*, *get_open_orders*, *get_my_open_orders*, *get_trade_history*

Fixed

- bug in _convert_to_currency_quantity for large values

v0.3.3 - 2018-08-08

Added

- override params for requests and aiohttp libraries

v0.3.2 - 2018-06-27

Fixed

- rlp version requirement

v0.3.1 - 2018-06-27

Fixed

- setup.py requirements

v0.3.0 - 2018-06-21

Added

- async versions of REST API
- async websocket interface

Fixed

- extracted dependencies from pyethereum to utils to fix Windows installs

Removed

- Python 3.3 and 3.4 support

v0.2.7 - 2018-01-30

Fixed

- revert *get_currency* call

v0.2.6 - 2018-01-29

Fixed

- *get_currency* call with address for token

v0.2.5 - 2018-01-29

Fixed

- cancel order signature. . . again

Added

- token name parameter to *get_balances* call

v0.2.4 - 2018-01-23

Fixed

- set wallet address lowercase

Added

- validation of private key format

v0.2.3 - 2018-01-20

Fixed

- order of hash params in *create_trade* function

v0.2.2 - 2018-01-20

Fixed

- issue with hashed data in *cancel_order* function

v0.2.1 - 2018-01-19

Added

- Withdraw endpoint

Fixed

- issue with Nonce value being too high

v0.2.0 - 2017-11-16

Added

- Trading endpoints
- Better exception handling
- Reference currency by address as well as name

v0.1.0 - 2017-11-15

Added

- Implementation of all non trading REST endpoints.
- Helper functions for your wallet address
- Response exception handling

7.1.12 Idex API

client module

class idex.client.**BaseClient**(*api_key=None*, *requests_params=None*)
Bases: object

API_URL = 'https://api.idex.market'

__init__(*api_key=None*, *requests_params=None*)
IDEX API Client constructor

https://github.com/AuroraDAO/idex-api-docs

Parameters

- **api_key** (*address string*) – optional - Wallet address
- **requests_params** (*dict.*) – optional - Dictionary of requests params to use for all calls

set_wallet_address(*address*, *private_key=None*)

Set the wallet address. Optionally add the private_key, this is only required for trading.

Parameters

- **address** (*address string*) – Address of the wallet to use
- **private_key** (*string*) – optional - The private key for the address

```
client.set_wallet_address('0x925cfc20de3fcbdba2d6e7c75dbb1d0a3f93b8a3', 'priv_
↪key...')
```

Returns nothing

get_wallet_address()

Get the wallet address

```
address = client.get_wallet_address()
```

Returns address string

get_last_response()

Get the last response object for inspection

```
response = client.get_last_response()
```

Returns response objects

class idex.client.**Client**(*api_key*, *address=None*, *private_key=None*)

Bases: *idex.client.BaseClient*

__init__(*api_key*, *address=None*, *private_key=None*)

Parameters

- **api_key** (*string*) –
- **address** (*address string*) – optional - Wallet address
- **private_key** (*string*) – optional - The private key for the address

```
api_key = 'kjdfiaadmad'
client = Client(api_key=api_key)

# with wallet address and private key
address = '0x925cfc20de3fcbdba2d6e7c75dbb1d0a3f93b8a3'
private_key = 'priv_key...'
client = Client(api_key=api_key, address=address, private_key=private_key)
```

get_tickers()

Get all market tickers

Please note: If any field is unavailable due to a lack of trade history or a lack of 24hr data, the field will be set to 'N/A'. percentChange, baseVolume, and quoteVolume will never be 'N/A' but may be 0.

https://github.com/AuroraDAO/idex-api-docs#returnticker

```
tickers = client.get_tickers()
```

Returns API Response

```
{
    ETH_SAN:  {
        last: '0.000981',
        high: '0.0010763',
        low: '0.0009777',
        lowestAsk: '0.00098151',
        highestBid: '0.0007853',
        percentChange: '-1.83619353',
        baseVolume: '7.3922603247161',
        quoteVolume: '7462.998433'
    },
    ETH_LINK: {
        last: '0.001',
        high: '0.0014',
        low: '0.001',
        lowestAsk: '0.002',
        highestBid: '0.001',
        percentChange: '-28.57142857',
        baseVolume: '13.651606265667369466',
        quoteVolume: '9765.891979953083752189'
    }
    # all possible markets follow ...
}
```

Raises IdexResponseException, IdexAPIException

get_ticker(*market*)

Get ticker for selected market

Please note: If any field is unavailable due to a lack of trade history or a lack of 24hr data, the field will be set to 'N/A'. percentChange, baseVolume, and quoteVolume will never be 'N/A' but may be 0.

https://github.com/AuroraDAO/idex-api-docs#returnticker

Parameters **market** (*string*) – Name of market e.g. ETH_SAN

```
ticker = client.get_ticker('ETH_SAN')
```

Returns API Response

```
{
    last: '0.000981',
    high: '0.0010763',
    low: '0.0009777',
    lowestAsk: '0.00098151',
    highestBid: '0.0007853',
    percentChange: '-1.83619353',
    baseVolume: '7.3922603247161',
    quoteVolume: '7462.998433'
}
```

Raises IdexResponseException, IdexAPIException

get_24hr_volume()
Get all market tickers

https://github.com/AuroraDAO/idex-api-docs#return24volume

```
volume = client.get_24hr_volume()
```

Returns API Response

```
{
    ETH_REP: {
        ETH: '1.3429046745',
        REP: '105.29046745'
    },
    ETH_DVIP: {
        ETH: '4',
        DVIP: '4'
    },
    totalETH: '5.3429046745'
}
```

Raises IdexResponseException, IdexAPIException

get_order_book(*market*, *count=1*)
Get order book for selected market

Each market returned will have an asks and bids property containing all the sell orders and buy orders sorted by best price. Order objects will contain a price amount total and orderHash property but also a params property which will contain additional data about the order useful for filling or verifying it.

https://github.com/AuroraDAO/idex-api-docs#returnorderbook

Parameters

- **market** (*string*) – Name of market e.g. ETH_SAN
- **count** (*int*) – Number of items to return

```
orderbook = client.get_order_book('ETH_SAN')
```

Returns API Response

```
{
    asks: [
        {
            price: '2',
            amount: '1',
            total: '2',
            orderHash:
↪'0x6aee6591def621a435dd86eafa32dfc534d4baa38d715988d6f23f3e2f20a29a',
            params: {
                tokenBuy: '0x0000000000000000000000000000000000000000',
                buySymbol: 'ETH',
                buyPrecision: 18,
                amountBuy: '200000000000000000',
```

(continues on next page)

(continued from previous page)

```
                tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
                sellSymbol: 'DVIP',
                sellPrecision: 8,
                amountSell: '100000000',
                expires: 190000,
                nonce: 164,
                user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
            }
        }
    ],
    bids: [
        {
            price: '1',
            amount: '2',
            total: '2',
            orderHash:
↪'0x9ba97cfc6d8e0f9a72e9d26c377be6632f79eaf4d87ac52a2b3d715003b6536e',
            params: {
                tokenBuy: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
                buySymbol: 'DVIP',
                buyPrecision: 8,
                amountBuy: '200000000',
                tokenSell: '0x0000000000000000000000000000000000000000',
                sellSymbol: 'ETH',
                sellPrecision: 18,
                amountSell: '2000000000000000000',
                expires: 190000,
                nonce: 151,
                user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
            }
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

get_open_orders(*market*, *address*, *count=10*, *cursor=None*)

Get the open orders for a given market and address

Output is similar to the output for get_order_book() except that orders are not sorted by type or price, but are rather displayed in the order of insertion. As is the case with get_order_book(there is a params property of the response value that contains details on the order which can help with verifying its authenticity.

https://github.com/AuroraDAO/idex-api-docs#returnopenorders

Parameters

- **market** (*string*) – Name of market e.g. ETH_SAN
- **address** (*address string*) – Address to return open orders associated with
- **count** (*int*) – amount of results to return
- **cursor** (*str*) – For pagination. Provide the value returned in the idex-next-cursor HTTP header to request the next slice (or page)

```
orders = client.get_open_orders(
    'ETH_SAN',
    '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
[
    {
        orderNumber: 1412,
        orderHash:
↪'0xf1bbc500af8d411b0096ac62bc9b60e97024ad8b9ea170340ff0ecfa03536417',
        price: '2.3',
        amount: '1.2',
        total: '2.76',
        type: 'sell',
        params: {
            tokenBuy: '0x0000000000000000000000000000000000000000',
            buySymbol: 'ETH',
            buyPrecision: 18,
            amountBuy: '2760000000000000000',
            tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
            sellSymbol: 'DVIP',
            sellPrecision: 8,
            amountSell: '120000000',
            expires: 190000,
            nonce: 166,
            user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
        }
    },
    {
        orderNumber: 1413,
        orderHash:
↪'0x62748b55e1106f3f453d51f9b95282593ef5ce03c22f3235536cf63a1476d5e4',
        price: '2.98',
        amount: '1.2',
        total: '3.576',
        type: 'sell',
        params:{
            tokenBuy: '0x0000000000000000000000000000000000000000',
            buySymbol: 'ETH',
            buyPrecision: 18,
            amountBuy: '3576000000000000000',
            tokenSell: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
            sellSymbol: 'DVIP',
            sellPrecision: 8,
            amountSell: '120000000',
            expires: 190000,
            nonce: 168,
            user: '0xca82b7b95604f70b3ff5c6ede797a28b11b47d63'
        }
    }
]
```

Raises IdexResponseException, IdexAPIException

get_my_open_orders(**args*, ***kwargs*)

get_order_status(*order_hash*)
Returns a single order

https://docs.idex.market/#operation/returnOrderStatus

Parameters **order_hash** (*256-bit hex string*) – The order hash to query for associated trades

```
status = client.get_order_status('0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
{
    "timestamp": 1516415000,
    "market": "ETH_AURA",
    "orderNumber": 2101,
    "orderHash":
↪"0x3fe808be7b5df3747e5534056e9ff45ead5b1fcace430d7b4092e5fcd7161e21",
    "price": "0.000129032258064516",
    "amount": "3100",
    "total": "0.4",
    "type": "buy",
    "params": {
        "tokenBuy": "0x7c5a0ce9267ed19b22f8cae653f198e3e8daf098",
        "buyPrecision": 18,
        "amountBuy": "3100000000000000000000",
        "tokenSell": "0x0000000000000000000000000000000000000000",
        "sellPrecision": 18,
        "amountSell": "400000000000000000",
        "expires": 100000,
        "nonce": "1",
        "user": "0x57b080554ebafc8b17f4a6fd090c18fc8c9188a0"
    },
    "filled": "1900",
    "initialAmount": "5000",
    "status": "open"
}
```

Raises IdexResponseException, IdexAPIException

get_trade_history(*market=None*, *address=None*, *start=None*, *end=None*, *count=10*, *sort='desc'*, *cursor=None*)
Get the past 200 trades for a given market and address, or up to 10000 trades between a range specified in UNIX timetsamps by the "start" and "end" properties of your JSON input.

https://github.com/AuroraDAO/idex-api-docs#returntradehistory

Parameters

- **market** (*string*) – optional - will return an array of trade objects for the market, if omitted, will return an object of arrays of trade objects keyed by each market
- **address** (*address string*) – optional - If specified, return value will only include trades that involve the address as the maker or taker.
- **start** (*int*) – optional - The inclusive UNIX timestamp (seconds since epoch) marking the earliest trade that will be returned in the response, (Default - 0)

- **end** (*int*) – optional - The inclusive UNIX timestamp marking the latest trade that will be returned in the response. (Default - current timestamp)
- **count** (*int*) – optional - Number of records to be returned per request. Valid range: 1 .. 100
- **sort** (*string*) – optional - Possible values are asc (oldest first) and desc (newest first). Defaults to desc.
- **cursor** (*string*) – optional - For pagination. Provide the value returned in the idex-next-cursor HTTP header to request the next slice (or page). This endpoint uses the tid property of a record for the cursor.

```
trades = client.get_trade_history()

# get trades for the last 2 hours for ETH EOS market
start = int(time.time()) - (60 * 2) # 2 hours ago
trades = client.get_trade_history(market='ETH_EOS', start=start)
```

Returns API Response

```
{
    ETH_REP: [
        {
            date: '2017-10-11 21:41:15',
            amount: '0.3',
            type: 'buy',
            total: '1',
            price: '0.3',
            orderHash:
↪'0x600c405c44d30086771ac0bd9b455de08813127ff0c56017202c95df190169ae',
            uuid: 'e8719a10-aecc-11e7-9535-3b8451fd4699',
            transactionHash:
↪'0x28b945b586a5929c69337929533e04794d488c2d6e1122b7b915705d0dff8bb6'
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

get_my_trade_history(**args*, ***kwargs*)

get_currencies()

Get token data indexed by symbol

https://github.com/AuroraDAO/idex-api-docs#returncurrencies

```
currencies = client.get_currencies()
```

Returns API Response

```
{
    ETH: {
        decimals: 18,
        address: '0x0000000000000000000000000000000000000000',
        name: 'Ether'
```

(continues on next page)

(continued from previous page)

```
    },
    REP: {
        decimals: 8,
        address: '0xc853ba17650d32daba343294998ea4e33e7a48b9',
        name: 'Reputation'
    },
    DVIP: {
        decimals: 8,
        address: '0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c',
        name: 'Aurora'
    }
}
```

Raises IdexResponseException, IdexAPIException

get_currency(*currency*)

Get the details for a particular currency using it's token name or address

Parameters **currency** (*string or hex string*) – Name of the currency e.g. EOS or '0x7c5a0ce9267ed19b22f8cae653f198e3e8daf098'

```
# using token name
currency = client.get_currency('REP')

# using the address string
currency = client.get_currency('0xc853ba17650d32daba343294998ea4e33e7a48b9')
```

Returns

```
{
    decimals: 8,
    address: '0xc853ba17650d32daba343294998ea4e33e7a48b9',
    name: 'Reputation'
}
```

Raises IdexCurrencyNotFoundException, IdexResponseException, IdexAPIException

get_balances(*address*, *complete=False*)

Get available balances for an address (total deposited minus amount in open orders) indexed by token symbol.

https://github.com/AuroraDAO/idex-api-docs#returnbalances

Parameters

- **address** (*address string*) – Address to query balances of
- **complete** – Include available balances along with the amount you have in open orders for each token (Default False)
- **complete** – bool

```
balances = client.get_balances('0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
# Without complete details
{
    REP: '25.55306545',
    DVIP: '200000000.31012358'
}

# With complete details
{
    REP: {
        available: '25.55306545',
        onOrders: '0'
    },
    DVIP: {
        available: '200000000.31012358',
        onOrders: '0'
    }
}
```

Raises IdexResponseException, IdexAPIException

get_my_balances(**args*, ***kwargs*)

get_transfers(*address*, *start=None*, *end=None*)

Returns the deposit and withdrawal history for an address within a range, specified by the "start" and "end" properties of the JSON input, both of which must be UNIX timestamps. Withdrawals can be marked as "PENDING" if they are queued for dispatch, "PROCESSING" if the transaction has been dispatched, and "COMPLETE" if the transaction has been mined.

https://github.com/AuroraDAO/idex-api-docs#returndepositswithdrawals

Parameters

- **address** (*address string*) – Address to query deposit/withdrawal history for
- **start** (*int*) – optional - Inclusive starting UNIX timestamp of returned results (Default - 0)
- **end** (*int*) – optional - Inclusive ending UNIX timestamp of returned results (Default - current timestamp)

```
transfers = client.get_transfers('0xca82b7b95604f70b3ff5c6ede797a28b11b47d63')
```

Returns API Response

```
{
    deposits: [
        {
            depositNumber: 265,
            currency: 'ETH',
            amount: '4.5',
            timestamp: 1506550595,
            transactionHash:
↪'0x52897291dba0a7b255ee7a27a8ca44a9e8d6919ca14f917616444bf974c48897'
        }
    ],
    withdrawals: [
        {
```

(continues on next page)

(continued from previous page)

```
            withdrawalNumber: 174,
            currency: 'ETH',
            amount: '4.5',
            timestamp: 1506552152,
            transactionHash:
↪'0xe52e9c569fe659556d1e56d8cca2084db0b452cd889f55ec3b4e2f3af61faa57',
            status: 'COMPLETE'
        }
    ]
}
```

Raises IdexResponseException, IdexAPIException

get_my_transfers(**args*, ***kwargs*)

get_order_trades(*order_hash*)

Get all trades involving a given order hash, specified by the order_hash

https://github.com/AuroraDAO/idex-api-docs#returnordertrades

Parameters **order_hash** (*256-bit hex string*) – The order hash to query for associated trades

```
trades = client.get_order_trades(
↪'0x62748b55e1106f3f453d51f9b95282593ef5ce03c22f3235536cf63a1476d5e4')
```

Returns API Response

```
[
    {
        date: '2017-10-11 21:41:15',
        amount: '0.3',
        type: 'buy',
        total: '1',
        price: '0.3',
        uuid: 'e8719a10-aecc-11e7-9535-3b8451fd4699',
        transactionHash:
↪'0x28b945b586a5929c69337929533e04794d488c2d6e1122b7b915705d0dff8bb6'
    }
]
```

Raises IdexResponseException, IdexAPIException

get_next_nonce(*address*)

Get the lowest nonce that you can use from the given address in one of the trade functions

https://github.com/AuroraDAO/idex-api-docs#returnnextnonce

Parameters **address** (*address string*) – The address to query for the next nonce to use

```
nonce = client.get_next_nonce('0xf59fad2879fb8380ffa6049a48abf9c9959b3b5c')
```

Returns API Response

```
{
    nonce: 2650
}
```

Raises IdexResponseException, IdexAPIException

get_my_next_nonce(**args*, ***kwargs*)

get_contract_address()

Get the contract address used for depositing, withdrawing, and posting orders

https://github.com/AuroraDAO/idex-api-docs#returncontractaddress

```
trades = client.get_contract_address()
```

Returns API Response

```
{
    address: '0x2a0c0dbecc7e4d658f48e01e3fa353f44050c208'
}
```

Raises IdexResponseException, IdexAPIException

parse_from_currency_quantity(*currency*, *quantity*)

Convert a quantity string to a float

Parameters

- **currency** (*string*) – Name of currency e.g EOS
- **quantity** (*string*) – Quantity value as string '310000000000000000000'

Returns decimal

convert_to_currency_quantity(*currency*, *quantity*)

Convert a float quantity to the correct decimal places

Parameters

- **currency** (*string*) – Name or address of currency e.g EOS or '0x7c5a0ce9267ed19b22f8cae653f198e3e8daf098'
- **quantity** (*Decimal, string, int, float*) – Quantity value 4.234298924 prefer Decimal or string, int or float should work

create_order(**args*, ***kwargs*)

create_order_wei(**args*, ***kwargs*)

create_trade(**args*, ***kwargs*)

cancel_order(**args*, ***kwargs*)

withdraw(**args*, ***kwargs*)

exceptions module

exception idex.exceptions.**IdexException**(*message*)
Bases: exceptions.Exception

__init__(*message*)
x.__init__(...) initializes x; see help(type(x)) for signature

exception idex.exceptions.**IdexAPIException**(*response*, *status_code*, *text*)
Bases: exceptions.Exception

Exception class to handle general API Exceptions

code values

message format

__init__(*response*, *status_code*, *text*)
x.__init__(...) initializes x; see help(type(x)) for signature

exception idex.exceptions.**IdexRequestException**(*message*)
Bases: exceptions.Exception

__init__(*message*)
x.__init__(...) initializes x; see help(type(x)) for signature

exception idex.exceptions.**IdexCurrencyNotFoundException**(*message*)
Bases: *idex.exceptions.IdexException*

exception idex.exceptions.**IdexWalletAddressNotFoundException**
Bases: exceptions.Exception

exception idex.exceptions.**IdexPrivateKeyNotFoundException**
Bases: exceptions.Exception

7.2 Index

- genindex

Python Module Index

i

Index

Symbols

A

B

C

G

I

P

S

W