Privex Python Exchange Library Documentation

Privex Inc., Chris (Someguy123)

Apr 13, 2020

MAIN:

Welcome to the documentation for Privex's Python Exchange library - a python package built for querying cryptocurrency exchanges and price aggregators such as CoinGecko

This documentation is automatically kept up to date by ReadTheDocs, as it is automatically re-built each time a new commit is pushed to the Github Project

CHAPTER

ONE

CONTENTS

1.1 Installation

1.1.1 Download and install from PyPi using pip (recommended)

```
pip3 install privex-exchange
```

1.1.2 (Alternative) Manual install from Git

Option 1 - Use pip to install straight from Github

```
pip3 install git+https://github.com/Privex/python-exchange
```

Option 2 - Clone and install manually

```
# Clone the repository from Github
git clone https://github.com/Privex/python-exchange
cd python-exchange

# RECOMMENDED MANUAL INSTALL METHOD
# Use pip to install the source code
pip3 install .

# ALTERNATIVE MANUAL INSTALL METHOD
# If you don't have pip, or have issues with installing using it, then you can use␣
↪setuptools instead.
python3 setup.py install
```

privex.exchange	
privex.exchange.base	
privex.exchange.Binance([extra_provides, ...])	
privex.exchange.Bittrex([extra_provides, ...])	
privex.exchange.CoinGecko([extra_provides, ...])	
privex.exchange.Kraken([extra_provides, ...])	
tests	

1.2 privex.exchange

1.3 privex.exchange.base

Functions

empty_decimal(obj)	

1.3.1 empty_decimal

empty_decimal(*obj: Optional[decimal.Decimal]*)

Classes

ExchangeAdapter([extra_provides, ...])	Abstract base class used by all exchange adapters for interoperability
ExchangeManager	Basic usage.
PriceData(from_coin, to_coin, last[, bid, ...])	Exchange price data object returned by exchange adapters

1.3.2 ExchangeAdapter

class ExchangeAdapter(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

Abstract base class used by all exchange adapters for interoperability

__init__(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

Initialize self. See help(type(self)) for accurate signature.

1.3.2.1 Methods

Methods

__init__([extra_provides, validate_provides])	Initialize self.
get_pair(from_coin, to_coin)	**Instantiate an exchange class and get a ticker asynchronously**.
has_pair(from_coin, to_coin)	

1.3.2.1.1 __init__

ExchangeAdapter.**__init__**(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

Initialize self. See help(type(self)) for accurate signature.

1.3.2.1.2 get_pair

async ExchangeAdapter.**get_pair**(*from_coin: str*, *to_coin: str*) → *privex.exchange.base.PriceData*

Instantiate an exchange class and get a ticker asynchronously:

```
>>> from privex.exchange import Binance
>>> b = Binance()
>>> ticker = await b.get_pair('LTC', 'BTC')
>>> ticker.last
Decimal('0.00607100')
>>> ticker.volume
Decimal('89557.57000000')
```

Can use synchronously in non-async applications:

```
>>> from privex.exchange import Binance
>>> b = Binance()
>>> ticker = b.get_pair('LTC', 'BTC')
>>> ticker.last
Decimal('0.00607100')
```

Parameters

- **from_coin** –
- **to_coin** –

Returns

1.3.2.1.3 has_pair

abstract async ExchangeAdapter.**has_pair**(*from_coin: str*, *to_coin: str*) → bool

1.3.2.2 Attributes

Attributes

cache_timeout
code
name
provides

1.3.2.2.1 cache_timeout

ExchangeAdapter.**cache_timeout: int = 120**

1.3.2.2.2 code

abstract property ExchangeAdapter.**code**

1.3.2.2.3 name

abstract property ExchangeAdapter.**name**

1.3.2.2.4 provides

abstract property ExchangeAdapter.**provides**

1.3.3 ExchangeManager

class ExchangeManager

Basic usage:

```
>>> from privex.exchange import ExchangeManager
>>> exm = ExchangeManager()
>>> await exm.get_pair('btc', 'usd')
Decimal('6694.53000000')
>>> await exm.get_pair('ltc', 'usd')
Decimal('40.15000000')
```

Converting arbitrary cryptos between each other, seamlessly:

```
>>> await exm.get_pair('eos', 'ltc')     # LTC per 1 EOS
Decimal('0.059573048699132755517011340894')
>>> await exm.get_pair('hive', 'eos')    # EOS per 1 HIVE
Decimal('0.043253079507278835386338185 90')
```

__init__()

Initialize self. See help(type(self)) for accurate signature.

1.3.3.1 Methods

Methods

exchange_by_code(code)	
exchange_by_name(name)	
exchange_by_path(obj_path)	
get_pair(from_coin, to_coin[, rate, use_proxy])	
get_tickers(from_coin, to_coin[, rate])	
list_pairs_from(frm_coin)	

continues on next page

Table 6 – continued from previous page

list_pairs_to(to_coin)	
load_exchange(package, name)	
load_exchanges()	
pair_exists(from_coin, to_coin[, should_raise])	
try_proxy(from_coin, to_coin[, proxy, rate, ...])	

```
>>> exm = ExchangeManager()
```

1.3.3.1.1 exchange_by_code

classmethod ExchangeManager.**exchange_by_code**(*code: str*) → *privex.exchange.base.ExchangeAdapter*

1.3.3.1.2 exchange_by_name

classmethod ExchangeManager.**exchange_by_name**(*name: str*) → *privex.exchange.base.ExchangeAdapter*

1.3.3.1.3 exchange_by_path

classmethod ExchangeManager.**exchange_by_path**(*obj_path: str*) → *privex.exchange.base.ExchangeAdapter*

1.3.3.1.4 get_pair

async ExchangeManager.**get_pair**(*from_coin: str*, *to_coin: str*, *rate='last'*, *use_proxy: bool = True*)

1.3.3.1.5 get_tickers

async classmethod ExchangeManager.**get_tickers**(*from_coin: str*, *to_coin: str*, *rate='last'*) → Dict[str, decimal.Decimal]

1.3.3.1.6 list_pairs_from

classmethod ExchangeManager.**list_pairs_from**(*frm_coin: str*) → Dict[str, *privex.exchange.base.ExchangeAdapter*]

1.3.3.1.7 list_pairs_to

classmethod ExchangeManager.**list_pairs_to**(*to_coin: str*) → Dict[str, *privex.exchange.base.ExchangeAdapter*]

1.3.3.1.8 load_exchange

async classmethod ExchangeManager.**load_exchange**(*package: str*, *name: str*) → *privex.exchange.base.ExchangeAdapter*

1.3.3.1.9 load_exchanges

async classmethod ExchangeManager.**load_exchanges**()

1.3.3.1.10 pair_exists

classmethod ExchangeManager.**pair_exists**(*from_coin: str*, *to_coin: str*, *should_raise=False*) → bool

1.3.3.1.11 try_proxy

async classmethod ExchangeManager.**try_proxy**(*from_coin: str*, *to_coin: str*, *proxy: str = 'BTC'*, *rate='last'*, *adapter: privex.exchange.base.ExchangeAdapter = None*)

```
>>> exm = ExchangeManager()
>>> await exm.load_exchanges()
>>> await exm.try_proxy('HIVE', 'USD', proxy='BTC')
```

Parameters

- **from_coin** –
- **to_coin** –
- **proxy** –
- **rate** –
- **adapter** –

Returns

1.3.3.2 Attributes

Attributes

available_adapters
ex_code_map
ex_instances
ex_name_map
ex_pair_map
ex_pair_map_inv
loaded
map_tether_real
proxy_coins
tether_map

1.3.3.2.1 available_adapters

ExchangeManager.**available_adapters: List[Tuple[str, str]] = [('privex.exchange.Binance',**

1.3.3.2.2 ex_code_map

ExchangeManager.**ex_code_map: Dict[str, str] = {}**

1.3.3.2.3 ex_instances

ExchangeManager.**ex_instances: Dict[str, privex.exchange.base.ExchangeAdapter] = {}**

1.3.3.2.4 ex_name_map

ExchangeManager.**ex_name_map: Dict[str, str] = {}**

1.3.3.2.5 ex_pair_map

ExchangeManager.**ex_pair_map: Dict[str, List[privex.exchange.base.ExchangeAdapter]] = {}**

1.3.3.2.6 ex_pair_map_inv

ExchangeManager.**ex_pair_map_inv: Dict[privex.exchange.base.ExchangeAdapter, Set[Tuple[str,**

1.3.3.2.7 loaded

ExchangeManager.**loaded = False**

1.3.3.2.8 map_tether_real

ExchangeManager.**map_tether_real: bool = True**

1.3.3.2.9 proxy_coins

ExchangeManager.**proxy_coins = ['BTC', 'USD', 'USDT']**

1.3.3.2.10 tether_map

ExchangeManager.**tether_map = {'USDC': 'USD', 'USDT': 'USD'}**

1.3.4 PriceData

class PriceData(*from_coin: str*, *to_coin: str*, *last*, *bid=None*, *ask=None*, *open=None*, *close=None*, *high=None*, *low=None*, *volume=None*)

Exchange price data object returned by exchange adapters

__init__(*from_coin: str*, *to_coin: str*, *last*, *bid=None*, *ask=None*, *open=None*, *close=None*, *high=None*, *low=None*, *volume=None*) → None

Initialize self. See help(type(self)) for accurate signature.

1.3.4.1 Methods

Methods

__init__(from_coin, to_coin, last[, bid, ...])	Initialize self.

1.3.4.1.1 __init__

PriceData.**__init__**(*from_coin: str*, *to_coin: str*, *last*, *bid=None*, *ask=None*, *open=None*, *close=None*, *high=None*, *low=None*, *volume=None*) → None

Initialize self. See help(type(self)) for accurate signature.

Exceptions

`ExchangeDown`	Raised when an exchange appears to be down, e.g.
`ExchangeException`	Base exception for all exceptions that are part of *privex.exchange*
`ExchangeRateLimited`	Raised when an exchange adapter encounters a rate limit while querying an exchange
`PairNotFound`	Raised when a requested coin pair isn't supported by this exchange adapter
`ProxyMissingPair`	Raised when a requested coin pair required for try_proxy does not exist.

1.4 Binance

class Binance(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

__init__(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)
Initialize self. See help(type(self)) for accurate signature.

1.4.1 Methods

Methods

—

1.4.2 Attributes

Attributes

—

1.5 Bittrex

class Bittrex(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

__init__(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)
Initialize self. See help(type(self)) for accurate signature.

1.5.1 Methods

Methods

—

1.5.2 Attributes

Attributes

—

1.6 CoinGecko

class CoinGecko(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

__init__(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)
Initialize self. See help(type(self)) for accurate signature.

1.6.1 Methods

Methods

—

1.6.2 Attributes

Attributes

—

1.7 Kraken

class Kraken(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)

__init__(*extra_provides: Union[List[Tuple[str, str]], Set[List[Tuple[str, str]]]] = None*, *validate_provides: bool = True*, ***ex_settings*)
Initialize self. See help(type(self)) for accurate signature.

1.7.1 Methods

Methods

—

1.7.2 Attributes

Attributes

—

1.8 tests

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

t

INDEX

Symbols

A

B

C

E

G

H

K

L

M

N

P

T