
crix Documentation

Release 1.0

Baryshnikov Aleksandr, Blockwise LTD

Dec 10, 2019

Contents:

1	Installation	3
2	Rate limit	5
3	Sample usage	7
3.1	Unauthorized (public) access	7
3.2	Authorized (clients-only) access	7
3.3	Authorized asynchronous access - get balance	8
4	Indices and tables	21
	Python Module Index	23
	Index	25

This official client of CRIX.io crypto exchange.

Environment requirements:

- python 3.5+
- requests 2.*

For several operations like create/cancel orders you should also be registered in the exchange and got BOT API token and secret.

To access historical data you should get explicit permission by exchange support.

CHAPTER 1

Installation

- over pip: `pip install crix`
- manually: `pip install git+https://github.com/blockwise/crix-client-py.git#egg=crix`

CHAPTER 2

Rate limit

Currently for BOT API there is a rate limit about 100 requests/second, however several functions in the library can use multiple requests inside as noted in their documentation.

3.1 Unauthorized (public) access

```
import crix

client = crix.Client(env='prod')

# get all symbols
for symbol in client.fetch_markets():
    print(symbol)

# get some order book
depth = client.fetch_order_book('BTC_BCH')
print(depth)
```

3.2 Authorized (clients-only) access

Warning: BOT API token and secret are required and should be obtained for each client from the exchange

```
import crix
from crix.models import NewOrder

client = crix.AuthorizedClient(
    env='prod',
    token='xxyyzz',
    secret='aabbcc'
) # replace token and secret value for your personal API credentials
```

(continues on next page)

(continued from previous page)

```

# list all open orders
for order in client.fetch_open_orders('BTC_BCH'):
    print(order)

# prepare order
new_order = NewOrder.market('BTC_BCH', is_buy=True, quantity=0.1) # or use NewOrder_
↳ constructor
# place order
order = client.create_order(new_order)
print(order)

```

3.3 Authorized asynchronous access - get balance

```

import crix
from crix.models import NewOrder
from aiohttp import ClientSession

async def run():
    # initialize HTTP(s) session
    async with ClientSession() as session:
        client = crix.AsyncAuthorizedClient(token='xxyyzz',
                                           secret='aabbcc',
                                           env='prod',
                                           session=session) # replace token and secret_
↳ value for your personal API credentials
        # list opened and closed orders
        async for x in client.fetch_orders('ETH_BTC'):
            print(x)

asyncio.get_event_loop().run_until_complete(run())

```

3.3.1 API clients

exception `crix.client.APIError` (*operation, code, text*)

General exception for API calls

static `async_ensure` (*operation, req*)

Ensure status code of HTTP request and raise exception if needed (asyncio version)

Parameters

- **operation** (`str`) – logical operation name
- **req** (`ClientResponse`) – request's response object

code = `None`

HTTP response code

static `ensure` (*operation, req*)

Ensure status code of HTTP request and raise exception if needed

Parameters

- **operation** (`str`) – logical operation name
- **req** (`Response`) – request's response object

operation = None
operation name

text = None
error description

class `crix.client.AuthorizedClient` (*token, secret, *, env='mvp', cache_market=True*)
HTTP client to the exchange for non-authorized and authorized requests.

Supported environments:

- 'mvp' - testnet sandbox with full-wipe each 2nd week (usually)
- 'prod' - mainnet, production environment with real currency

Expects API token and API secret provided by CRIX.IO exchange as part of bot API.

cancel_order (*order_id, symbol*)
Cancel placed order

Parameters

- **order_id** (`int`) – order id generated by the exchange
- **symbol** (`str`) – symbol names same as in placed order

Return type `Order`

Returns order definition with filled field (also includes filled quantity)

create_order (*new_order*)
Create and place order to the exchange

Parameters **new_order** (`NewOrder`) – order parameters

Return type `Order`

Returns order definition with filled fields from the exchange

fetch_balance ()
Get all balances for the user

Return type `List[Account]`

Returns list of all accounts

fetch_closed_orders (**symbols, limit=1000*)
Get complete (filled, canceled) orders for user

Note: One request per each symbol will be made plus additional request to query all supported symbols if symbols parameter not specified.

Parameters

- **symbols** (`str`) – filter orders by symbols. if not specified - all symbols queried and used
- **limit** (`int`) – maximum number of orders for each symbol

Return type `Iterator[Order]`

Returns iterator of orders definitions

fetch_history (*begin, end, currency*)

Get historical minute tickers for specified time range and currency There are several caveats:

- it requires additional permission
- end param should be not more then server time, otherwise error returned
- maximum difference between earliest and latest date should be no more then 366 days
- it could be slow for a long time range
- mostly all points have 1 minute tick however in a very few cases gap can be a bit bigger

Parameters

- **begin** (*datetime*) – earliest interesting time
- **end** (*datetime*) – latest interesting time
- **currency** (*str*) – currency name in upper case

Return type `Iterator[Ticker]`

Returns iterator of parsed tickers

fetch_my_trades (**symbols, limit=1000*)

Get all trades for the user. There is some gap (a few ms) between time when trade is actually created and time when it becomes visible for the user.

Note: One request per each symbol will be made plus additional request to query all supported symbols if symbols parameter not specified.

Parameters

- **symbols** (*str*) – filter trades by symbols. if not specified - used all symbols
- **limit** (*int*) – maximum number of trades for each symbol

Return type `Iterator[Trade]`

Returns iterator of trade definition

fetch_open_orders (**symbols, limit=1000*)

Get all open orders for the user.

Note: One request per each symbol will be made plus additional request to query all supported symbols if symbols parameter not specified.

Parameters

- **symbols** (*str*) – filter orders by symbols. if not specified - all symbols queried and used
- **limit** (*int*) – maximum number of orders for each symbol

Return type `Iterator[Order]`

Returns iterator of orders definitions

fetch_order (*order_id, symbol_name*)

Fetch single open order info

Parameters

- **order_id** (int) – order id generated by server during ‘create_order’ phase
- **symbol_name** (str) – symbol name same as in order

Return type Optional[Order]

Returns order definition or None if nothing found

fetch_orders (*symbols, limit=1000)

Get opened and closed orders filtered by symbols. If no symbols specified - all symbols are used. Basically the function acts as union of fetch_open_orders and fetch_closed_orders.

Note: Two requests per each symbol will be made plus additional request to query all supported symbols if symbols parameter not specified.

Parameters

- **symbols** (str) – symbols: filter orders by symbols. if not specified - used all symbols
- **limit** (int) – maximum number of orders for each symbol for each state (open, close)

Return type Iterator[Order]

Returns iterator of orders definitions sorted from open to close

class crix.client.Client (*, env='mvp', cache_market=True)

HTTP client to the exchange for non-authorized requests.

Supported environments:

- ‘mvp’ - testnet sandbox with full-wipe each 2nd week (usually)
- ‘prod’ - mainnet, production environment with real currency

Disable *cache_market* if latest symbols info are always required

fetch_currency_codes ()

Get list of currencies codes in quote_base format (ex. btc_bch)

Return type List[str]

Returns list of formatted currencies codes

fetch_markets (force=False)

Get list of all symbols on the exchange. Also includes symbol details like precision, quote, base and e.t.c. It’s a good idea to cache result of this function after first invoke

Parameters **force** (bool) – don’t use cached symbols

Return type Tuple[Symbol]

Returns list of supported symbols

fetch_ohlcv (symbol, utc_start_time, utc_end_time, resolution=<Resolution.one_minute: '1'>, limit=10)

Get K-Lines for specific symbol in a time frame.

Latest OHLCV ticks representing interval up to current minute (ex: now: 10:15:32, then latest OHLCV with minute resolution will be from 10:14:00 to 10:15:00).

Parameters

- **symbol** (str) – K-Line symbol name

- **utc_start_time** (datetime) – earliest interesting time
- **utc_end_time** (datetime) – latest interesting time
- **resolution** (*Resolution*) – K-line resolution (by default 1-minute)
- **limit** (int) – maximum number of entries in a response

Return type List[*Ticker*]

Returns list of ticker

fetch_order_book (*symbol*, *level_aggregation=None*)

Get order book for specific symbol and level aggregation

```
import os
import crix

client = crix.AuthorizedClient(token=os.getenv('TOKEN'),
                               secret=os.getenv('SECRET'),
                               env='mvp')

# get all symbols
symbols = client.fetch_markets()
for symbol in symbols:
    # get order book for symbol
    order_book = client.fetch_order_book(symbol.name)
```

Parameters

- **symbol** (str) – interesting symbol name
- **level_aggregation** (Optional[int]) – aggregate by rounding numbers (if not defined - no aggregation)

Return type *Depth*

Returns order depth book

fetch_ticker ()

Get tickers for all symbols for the last 24 hours

Return type List[*Ticker24*]

Returns list of tickers

fetch_trades (*symbol*, *limit=100*)

Get last trades for specified symbol name. OrderID, UserID, Fee, FeeCurrency will be empty (or 0)

Parameters

- **symbol** (str) – symbol name
- **limit** (int) – maximum number of trades (could not be more than 1000)

Return type List[*Trade*]

Returns list of trades

3.3.2 API models

class crix.models.**Account** (*id*, *user_id*, *balance*, *locked_balance*, *currency_name*, *deposit_address*)

balance

Return type Decimal

currency_name

Return type str

deposit_address

Return type str

static from_json (*info*)

Construct object from dictionary

Return type *Account*

id

Return type int

locked_balance

Return type Decimal

user_id

Return type int

class `crix.models.Depth` (*symbol_name, is_aggregated, last_update_id, level_aggregation, asks, bids*)

asks

Return type `typing.List[crix.models.Offer]`

bids

Return type `typing.List[crix.models.Offer]`

static from_json (*info*)

Construct object from dictionary

Return type *Depth*

is_aggregated

Return type bool

last_update_id

Return type int

level_aggregation

Return type int

symbol_name

Return type str

class `crix.models.NewOrder` (*type, symbol, price, quantity, is_buy, time_in_force, stop_price, expire_time*)

expire_time

Return type `typing.Union[datetime.datetime, NoneType]`

is_buy

Return type bool

static limit (*symbol, is_buy, price, quantity, **args*)

Helper to create basic limit order

Parameters

- **symbol** (*str*) – symbol name as defined by the exchange
- **is_buy** (*bool*) – order direction
- **price** (*Union[Decimal, float, str]*) – order price
- **quantity** (*Union[Decimal, float, str]*) – number of items in the order
- **args** – additional parameters proxied to the `NewOrder` constructor

Return type *NewOrder*

Returns new order

static market (*symbol, is_buy, quantity, **args*)

Helper to create basic market order

Parameters

- **symbol** (*str*) – symbol name as defined by the exchange
- **is_buy** (*bool*) – order direction
- **quantity** (*Union[Decimal, float, str]*) – number of items
- **args** – additional parameters proxied to the `NewOrder` constructor

Return type *NewOrder*

Returns new order

price

Return type Decimal

quantity

Return type Decimal

stop_price

Return type *typing.Union[decimal.Decimal, NoneType]*

symbol

Return type str

time_in_force

Return type *TimeInForce*

to_json ()

Build JSON package ready to send to the API endpoint

Return type dict

type

Return type *OrderType*

class `crix.models.Offer` (*count, price, quantity*)

count
Return type int

static from_json (*info*)
Construct object from dictionary
Return type *Offer*

price
Return type Decimal

quantity
Return type Decimal

class `crix.models.Order` (*id, user_id, type, symbol_name, is_buy, quantity, price, stop_price, filled_quantity, time_in_force, expire_time, status, created_at, last_updated_at*)

created_at
Return type datetime

expire_time
Return type typing.Union[datetime.datetime, NoneType]

filled_quantity
Return type Decimal

static from_json (*info*)
Construct object from dictionary
Return type *Order*

id
Return type int

is_buy
Return type bool

last_updated_at
Return type datetime

price
Return type Decimal

quantity
Return type Decimal

status
Return type *OrderStatus*

stop_price
Return type Decimal

symbol_name
Return type str

`time_in_force`

Return type *TimeInForce*

`type`

Return type *OrderType*

`user_id`

Return type int

class `crix.models.OrderStatus`

An enumeration.

`cancel = 2`

`complete = 1`

`new = 0`

class `crix.models.OrderType`

An enumeration.

`limit = 0`

`market = 1`

`stop_loss = 2`

`stop_loss_limit = 3`

`stop_loss_range = 4`

`take_profit = 5`

`take_profit_limit = 6`

class `crix.models.Resolution`

An enumeration.

`day = 'D'`

`fifteen_minutes = '15'`

`five_minutes = '5'`

`four_hours = '240'`

`half_an_hour = '30'`

`hour = '60'`

`one_minute = '1'`

`two_hours = '120'`

`week = 'W'`

class `crix.models.Symbol`(*name, base, base_precision, quote, quote_precision, description, level_aggregation, maker_fee, taker_fee, min_lot, max_lot, min_price, max_price, min_notional, tick_lot, tick_price, is_trading*)

base

Return type str

base_precision

Return type int

description

Return type `str`

static from_json (*info*)

Construct object from dictionary

Return type *Symbol*

is_trading

Return type `bool`

level_aggregation

Return type `typing.List[int]`

maker_fee

Return type `Decimal`

max_lot

Return type `Decimal`

max_price

Return type `Decimal`

min_lot

Return type `Decimal`

min_notional

Return type `Decimal`

min_price

Return type `Decimal`

name

Return type `str`

quote

Return type `str`

quote_precision

Return type `int`

taker_fee

Return type `Decimal`

tick_lot

Return type `Decimal`

tick_price

Return type `Decimal`

class `crix.models.Ticker` (*symbol_name, open_time, open, close, high, low, volume, resolution*)

close

Return type `Decimal`

static `from_json` (*info*)
Construct object from dictionary

Return type *Ticker*

static `from_json_history` (*info*)
Construct object from dictionary (for a fixed resolution)

Return type *Ticker*

high
Return type Decimal

low
Return type Decimal

open
Return type Decimal

open_time
Return type datetime

resolution
Return type str

symbol_name
Return type str

volume
Return type Decimal

class `crix.models.Ticker24` (*symbol_name, open_time, open, close, high, low, volume, resolution, first_id, last_id, prev_close_price, price_change, price_change_percent*)

close
Return type Decimal

first_id
Return type int

static `from_json` (*info*)
Construct object from dictionary

Return type *Ticker24*

high
Return type Decimal

last_id
Return type int

low
Return type Decimal

open
Return type Decimal

open_time

Return type datetime

prev_close_price

Return type Decimal

price_change

Return type Decimal

price_change_percent

Return type Decimal

resolution

Return type str

symbol_name

Return type str

volume

Return type Decimal

class `crix.models.TimeInForce`

An enumeration.

fill_or_kill = 2

good_till_cancel = 0

good_till_date = 3

immediate_or_cancel = 1

class `crix.models.Trade`(*id, user_id, created_at, order_filled, is_buy, order_id, price, quantity, fee, fee_currency, symbol_name*)

created_at

Return type datetime

fee

Return type Decimal

fee_currency

Return type str

static `from_json`(*info*)

Construct object from dictionary

Return type *Trade*

id

Return type int

is_buy

Return type bool

order_filled

Return type bool

`order_id`

Return type int

`price`

Return type Decimal

`quantity`

Return type Decimal

`symbol_name`

Return type str

`user_id`

Return type int

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

C

`crix.client`, 8
`crix.models`, 12

A

Account (*class in crix.models*), 12
APIError, 8
asks (*crix.models.Depth attribute*), 13
async_ensure() (*crix.client.APIError static method*), 8
AuthorizedClient (*class in crix.client*), 9

B

balance (*crix.models.Account attribute*), 12
base (*crix.models.Symbol attribute*), 16
base_precision (*crix.models.Symbol attribute*), 16
bids (*crix.models.Depth attribute*), 13

C

cancel (*crix.models.OrderStatus attribute*), 16
cancel_order() (*crix.client.AuthorizedClient method*), 9
Client (*class in crix.client*), 11
close (*crix.models.Ticker attribute*), 17
close (*crix.models.Ticker24 attribute*), 18
code (*crix.client.APIError attribute*), 8
complete (*crix.models.OrderStatus attribute*), 16
count (*crix.models.Offer attribute*), 14
create_order() (*crix.client.AuthorizedClient method*), 9
created_at (*crix.models.Order attribute*), 15
created_at (*crix.models.Trade attribute*), 19
crix.client (*module*), 8
crix.models (*module*), 12
currency_name (*crix.models.Account attribute*), 13

D

day (*crix.models.Resolution attribute*), 16
deposit_address (*crix.models.Account attribute*), 13
Depth (*class in crix.models*), 13
description (*crix.models.Symbol attribute*), 17

E

ensure() (*crix.client.APIError static method*), 8
expire_time (*crix.models.NewOrder attribute*), 13
expire_time (*crix.models.Order attribute*), 15

F

fee (*crix.models.Trade attribute*), 19
fee_currency (*crix.models.Trade attribute*), 19
fetch_balance() (*crix.client.AuthorizedClient method*), 9
fetch_closed_orders() (*crix.client.AuthorizedClient method*), 9
fetch_currency_codes() (*crix.client.Client method*), 11
fetch_history() (*crix.client.AuthorizedClient method*), 9
fetch_markets() (*crix.client.Client method*), 11
fetch_my_trades() (*crix.client.AuthorizedClient method*), 10
fetch_ohlcv() (*crix.client.Client method*), 11
fetch_open_orders() (*crix.client.AuthorizedClient method*), 10
fetch_order() (*crix.client.AuthorizedClient method*), 10
fetch_order_book() (*crix.client.Client method*), 12
fetch_orders() (*crix.client.AuthorizedClient method*), 11
fetch_ticker() (*crix.client.Client method*), 12
fetch_trades() (*crix.client.Client method*), 12
fifteen_minutes (*crix.models.Resolution attribute*), 16
fill_or_kill (*crix.models.TimeInForce attribute*), 19
filled_quantity (*crix.models.Order attribute*), 15
first_id (*crix.models.Ticker24 attribute*), 18
five_minutes (*crix.models.Resolution attribute*), 16
four_hours (*crix.models.Resolution attribute*), 16
from_json() (*crix.models.Account static method*), 13

from_json() (*crix.models.Depth* static method), 13
 from_json() (*crix.models.Offer* static method), 15
 from_json() (*crix.models.Order* static method), 15
 from_json() (*crix.models.Symbol* static method), 17
 from_json() (*crix.models.Ticker* static method), 17
 from_json() (*crix.models.Ticker24* static method), 18
 from_json() (*crix.models.Trade* static method), 19
 from_json_history() (*crix.models.Ticker* static method), 18

G

good_till_cancel (*crix.models.TimeInForce* attribute), 19
 good_till_date (*crix.models.TimeInForce* attribute), 19

H

half_an_hour (*crix.models.Resolution* attribute), 16
 high (*crix.models.Ticker* attribute), 18
 high (*crix.models.Ticker24* attribute), 18
 hour (*crix.models.Resolution* attribute), 16

I

id (*crix.models.Account* attribute), 13
 id (*crix.models.Order* attribute), 15
 id (*crix.models.Trade* attribute), 19
 immediate_or_cancel (*crix.models.TimeInForce* attribute), 19
 is_aggregated (*crix.models.Depth* attribute), 13
 is_buy (*crix.models.NewOrder* attribute), 13
 is_buy (*crix.models.Order* attribute), 15
 is_buy (*crix.models.Trade* attribute), 19
 is_trading (*crix.models.Symbol* attribute), 17

L

last_id (*crix.models.Ticker24* attribute), 18
 last_update_id (*crix.models.Depth* attribute), 13
 last_updated_at (*crix.models.Order* attribute), 15
 level_aggregation (*crix.models.Depth* attribute), 13
 level_aggregation (*crix.models.Symbol* attribute), 17
 limit (*crix.models.OrderType* attribute), 16
 limit() (*crix.models.NewOrder* static method), 14
 locked_balance (*crix.models.Account* attribute), 13
 low (*crix.models.Ticker* attribute), 18
 low (*crix.models.Ticker24* attribute), 18

M

maker_fee (*crix.models.Symbol* attribute), 17
 market (*crix.models.OrderType* attribute), 16
 market() (*crix.models.NewOrder* static method), 14
 max_lot (*crix.models.Symbol* attribute), 17

max_price (*crix.models.Symbol* attribute), 17
 min_lot (*crix.models.Symbol* attribute), 17
 min_notional (*crix.models.Symbol* attribute), 17
 min_price (*crix.models.Symbol* attribute), 17

N

name (*crix.models.Symbol* attribute), 17
 new (*crix.models.OrderStatus* attribute), 16
 NewOrder (class in *crix.models*), 13

O

Offer (class in *crix.models*), 14
 one_minute (*crix.models.Resolution* attribute), 16
 open (*crix.models.Ticker* attribute), 18
 open (*crix.models.Ticker24* attribute), 18
 open_time (*crix.models.Ticker* attribute), 18
 open_time (*crix.models.Ticker24* attribute), 19
 operation (*crix.client.APIError* attribute), 9
 Order (class in *crix.models*), 15
 order_filled (*crix.models.Trade* attribute), 19
 order_id (*crix.models.Trade* attribute), 19
 OrderStatus (class in *crix.models*), 16
 OrderType (class in *crix.models*), 16

P

prev_close_price (*crix.models.Ticker24* attribute), 19
 price (*crix.models.NewOrder* attribute), 14
 price (*crix.models.Offer* attribute), 15
 price (*crix.models.Order* attribute), 15
 price (*crix.models.Trade* attribute), 20
 price_change (*crix.models.Ticker24* attribute), 19
 price_change_percent (*crix.models.Ticker24* attribute), 19

Q

quantity (*crix.models.NewOrder* attribute), 14
 quantity (*crix.models.Offer* attribute), 15
 quantity (*crix.models.Order* attribute), 15
 quantity (*crix.models.Trade* attribute), 20
 quote (*crix.models.Symbol* attribute), 17
 quote_precision (*crix.models.Symbol* attribute), 17

R

Resolution (class in *crix.models*), 16
 resolution (*crix.models.Ticker* attribute), 18
 resolution (*crix.models.Ticker24* attribute), 19

S

status (*crix.models.Order* attribute), 15
 stop_loss (*crix.models.OrderType* attribute), 16
 stop_loss_limit (*crix.models.OrderType* attribute), 16

stop_loss_range (*crix.models.OrderType* attribute), 16
 stop_price (*crix.models.NewOrder* attribute), 14
 stop_price (*crix.models.Order* attribute), 15
 Symbol (*class in crix.models*), 16
 symbol (*crix.models.NewOrder* attribute), 14
 symbol_name (*crix.models.Depth* attribute), 13
 symbol_name (*crix.models.Order* attribute), 15
 symbol_name (*crix.models.Ticker* attribute), 18
 symbol_name (*crix.models.Ticker24* attribute), 19
 symbol_name (*crix.models.Trade* attribute), 20

T

take_profit (*crix.models.OrderType* attribute), 16
 take_profit_limit (*crix.models.OrderType* attribute), 16
 taker_fee (*crix.models.Symbol* attribute), 17
 text (*crix.client.APIError* attribute), 9
 tick_lot (*crix.models.Symbol* attribute), 17
 tick_price (*crix.models.Symbol* attribute), 17
 Ticker (*class in crix.models*), 17
 Ticker24 (*class in crix.models*), 18
 time_in_force (*crix.models.NewOrder* attribute), 14
 time_in_force (*crix.models.Order* attribute), 15
 TimeInForce (*class in crix.models*), 19
 to_json () (*crix.models.NewOrder* method), 14
 Trade (*class in crix.models*), 19
 two_hours (*crix.models.Resolution* attribute), 16
 type (*crix.models.NewOrder* attribute), 14
 type (*crix.models.Order* attribute), 16

U

user_id (*crix.models.Account* attribute), 13
 user_id (*crix.models.Order* attribute), 16
 user_id (*crix.models.Trade* attribute), 20

V

volume (*crix.models.Ticker* attribute), 18
 volume (*crix.models.Ticker24* attribute), 19

W

week (*crix.models.Resolution* attribute), 16