Introduction

Welcome to Blockchain.com's Exchange API and developer documentation. These documents detail and give examples of various functionality offered by the API such as receiving real time market data, requesting balance information and performing trades.

View the full REST API documentation here.

To Get Started

- Create or log into your existing Blockchain.com Exchange account
- Select API from the drop down menu
- Fill out form and click "Create New API Key Now"
- Once generated you can view your keys under API Settings

Websocket API

The Websocket API can be used to receive market data and to interact with the trading system in real time. Every message is in a JSON format and trading messages use the FIX standard for naming of fields, and message types.

Each type of data is provided over a dedicated channel. Clients need to subscribe to all relevant channels they wish to receive real time updates from.

Authentication

```
# Simple python websocket client
# https://github.com/websocket-client/websocket-client
from websocket import create_connection
options = {}
```

```
options['origin'] = 'https://exchange.blockchain.com'
url = "wss://ws.blockchain.info/mercury-gateway/v1/ws"
ws = create_connection(url, **options)
msg = '{"token": "{API_SECRET}", "action": "subscribe", "channel": "auth"}'
ws.send(msg)
result =  ws.recv()
print(result)
# { "seqnum":0,
#   "event":"subscribed",
#   "channel":"auth",
#   "readOnly":false }
ws.close()
```

Interaction with the API will require an API key. To generate one, go to the API section of your Blockchain.com Exchange user settings. Each API key generated will be linked to your account. The only required information is a key name. Optionally you can set trading access, and IP address whitelisting.

Once your key is created you'll be shown an **API Key** and **API Secret**. Store the secret in a safe place because it will only be shown once.

Connection

The websocket endpoint is,

Environment	URI
prod	`wss://ws.blockchain.info/mercury-gateway/v1/ws`

In order to connect you have to add the following headers to the connection request

Environment	Headers
prod	`Origin: https://exchange.blockchain.com`

Requests

Each Message sent to the server will have at least 2 fields: action and channel.

CHANNEL

A channel provides context about the type of data being communicated between the client and server. There are multiple channels available:

Channel	Visibility	Description
heartbeat	anonymous	Receive heartbeat messages
l2	anonymous	Receive level 2 order book data (aggregated)
l3	anonymous	Receive level 3 order book data (aggregated)
prices	anonymous	Receive candlestick market data

Channel	Visibility	Description
symbols	anonymous	Receive symbol messages
ticker	anonymous	Receive ticker messages
trades	anonymous	Receive trade execution messages
auth	authenticated	To authenticate a web socket connection
balances	authenticated	To receive balance updates
trading	authenticated	Submit & cancel orders, receive order snapshots and updates

ACTION

`action` describes what action to take for the provided channel. The following standard action's are supported by all channels (with the exception of the auth channel):

Action	Description
subscribe	Subscribe to the provided channel and attributes
unsubscribe	Unsubscribe from the provided channel and attributes

A channel may expose other bespoke actions.

Response

SEQUENCE NUMBERS

Each message sent from the server will contain a sequence number `seqnum` which will be incremented by 1 with each message. If the client receives a `seqnum` which has skipped one or more sequences, it indicates that a message was missed and the client is recommended to restart the websocket connection.

EVENT

In addition, each response field will contain an event field with the corresponding channel to indicate the purpose of the message. The following events are supported:

Event	Type	Description
subscribed	admin	The channel was successfully subscribed to
unsubscribed	admin	The channel was successfully unsubscribed to
rejected	admin	the last action for the channel was rejected. A text field will be provided about the reason for the reject
snapshot	app	A channel snapshot has been provided
updated	app	An update corresponding to the channel has occurred

Each time an `action` is applied to a channel, an administrative event is sent from the server to notify whether the `action` was applied successfully.

Data Types

All Messages use the standard JSON format. The following data types are supported:

Type	Description	Example
number	Signed decimal number	1234.45
string	UTF-8 encoded unicode string	"Authentication Failed"
timestamp	UTC Timestamps following the convention YYYY-MM-DDTHH:MM:SS.ssssssZ or YYYY-MM-DDTHH:MM:SS.000ssssssZ or YYYY-MM-DDT00:00:00Z	

Fair usage and rate limits

```
{
  "event": "rejected",
  "text": "Connection throttling enabled, your messages will be ignored."
}
```

Currently there is a limit of 1200 messages per minute. If the limit is exceeded, you will receive a `rejected` event. After waiting a minute, normal functionality will resume and you will be able to send messages again.

Response data throttling and delays

The response and update messages are never delayed, clients will get every update as soon as it happens.

Anonymous Channels

Heartbeat

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "heartbeat"
}
```

A heartbeat can be sent by the server by subscribing to the `heartbeat` channel.

Server response:

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "heartbeat"
}
```

The server will send confirmation of the subscription.

Snapshot Example:

```
{
  "seqnum": 1,
  "event": "updated",
  "channel": "heartbeat",
  "timestamp": "2019-05-31T08:36:45.666753Z"
}
```

Snapshot messages are sent every 5 seconds and have the following format:

L2 Order Book

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "l2",
  "symbol": "BTC-USD"
}
```

Level 2 Order Book data is available through the `l2` channel. This channel returns the volume available at each price. All the price levels are retrieved with this channel. Subscribing is done per symbol. Each entry in bids and asks arrays is a price level, along with its price (`px`), quantity (`qty`) and number of orders (`num`) attributes.

Server response:

```
{
  "seqnum": 1,
  "event": "subscribed",
  "channel": "l2",
  "symbol": "BTC-USD"
}
```

Snapshots:

```
{
  "seqnum": 2,
  "event": "snapshot",
  "channel": "l2",
  "symbol": "BTC-USD",
  "bids": [
    {
      "px": 8723.45,
      "qty": 1.45,
      "num": 2
    },
    {
      "px": 8124.45,
      "qty": 123.45,
      "num": 1
    }
  ],
  "asks": [
    {
      "px": 8730.0,
      "qty": 1.55,
      "num": 2
    },
    {
      "px": 8904.45,
      "qty": 13.66,
      "num": 2
    }
  ]
}
```

There is no ordering guarantee in bids and asks entries.

```
{
  "seqnum": 3,
  "event": "updated",
  "channel": "l2",
  "symbol": "BTC-USD",
  "bids": [
    {
      "px": 8723.45,
      "qty": 1.1,
      "num": 1
    }
  ],
  "asks": []
}
```

An update with qty equal to 0, means the price should be removed.

```
{
  "seqnum": 4,
  "event": "updated",
  "channel": "l2",
  "symbol": "BTC-USD",
  "bids": [
    {
      "px": 8723.45,
      "qty": 0.0,
      "num": 0
    }
  ],
  "asks": []
}
```

L3 Order Book

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "l3",
  "symbol": "BTC-USD"
}
```

Level 3 Order Book data is available through the `l3` channel. This channel returns all the order updates reaching the exchange; by applying the updates to the snapshot you can recreate the full state of the orderbook. Subscribing is done per symbol. Each entry in bids and asks arrays is an order, along with its id (`id`), price (`px`) and quantity (`qty`) attributes.

Server response:

```
{
  "seqnum": 1,
  "event": "subscribed",
  "channel": "l3",
  "symbol": "BTC-USD"
}
```

Example snapshot:

```
{
  "seqnum": 2,
  "event": "snapshot",
  "channel": "l3",
  "symbol": "BTC-USD",
  "bids": [
    {
      "id": "1234",
      "px": 8723.45,
      "qty": 1.1
    },
    {
      "id": "1235",
      "px": 8723.45,
      "qty": 0.35
    },
    {
      "id": "234",
      "px": 8124.45,
      "qty": 123.45
    }
  ],
  "asks": [
    {
      "id": "2222",
      "px": 8730.0,
      "qty": 0.65
    },
    {
      "id": "2225",
      "px": 8730.0,
      "qty": 0.9
    },
    {
      "id": "2343",
      "px": 8904.45,
      "qty": 8.66
    },
    {
      "id": "2353",
      "px": 8904.45,
      "qty": 5.0
    }
  ]
}
```

Updates will follow. An update with qty equal to 0, means the order should be removed.

```
{
  "seqnum": 3,
  "event": "updated",
  "channel": "l3",
  "symbol": "BTC-USD",
  "bids": [
    {
      "id": "1234",
      "px": 8723.45,
      "qty": 0
    }
  ],
  "asks": []
}
```

Prices

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "prices",
  "symbol": "BTC-USD",
  "granularity": 60
}
```

To receive candlestick market data you can subscribe to the prices channel. Subscriptions are per `symbol` and `granularity` (in seconds) has to be specified. Supported granularity values are: `60, 300, 900, 3600, 21600, 86400`

You can subscribe for multiple symbols but not for multiple granularities per symbol.

Server response:

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "prices",
  "symbol": "BTC-USD"
}
```

The server will send confirmation of the subscription.

Updated event on the prices channel:

```
{
  "seqnum": 2,
  "event": "updated",
  "channel": "prices",
  "symbol": "BTC-USD",
  "price": [1559039640, 8697.24, 8700.98, 8697.27, 8700.98, 0.431]
}
```

The price data is an array consisting of [`timestamp`, `open`, `high`, `low`, `close`, `volume`]

Symbols

To receive symbol updates, subscribe to the `symbols` channel. The server will send confirmation of the subscription. The next message on this channel will be a snapshot of the current `symbol` status.

When the `symbol` is not halted the auction data in the message may be blank.

When a `symbol` is in a halt state the auction data will populate as the book builds. When an opening time has been chosen, the `auction-time` field will show the opening time. Subsequent updates will be sent only if the `symbol` status changes in any way.

FIELD DESCRIPTION

Field	Description
auction-price	If the symbol is halted and will open on an auction, this will be the opening price.
auction-size	Opening size
auction-time	Opening time in HHMM format.
imbalance	Auction imbalance. If > 0 then there will be buy orders left over at the auction price. If < 0 then there will be sell orders left over at the auction price.
status	Symbol status; open, close, suspend, halt, halt-freeze.
base_currency	The currency quantities are expressed in
base_currency_scale	The number of decimals the currency can be split in
counter_currency	The currency prices are expressed in
counter_currency_scale	The number of decimals the currency can be split in
min_price_increment	The price of the instrument must be a multiple of min_price_increment * (10^-min_price_increment_scale)
min_price_increment_scale	The price of the instrument must be a multiple of min_price_increment * (10^-min_price_increment_scale)
min_order_size	The minimum quantity for an order for this instrument must be min_order_size*(10^-min_order_size_scale)
min_order_size_scale	The minimum quantity for an order for this instrument must be min_order_size*(10^-min_order_size_scale)
max_order_size	The maximum quantity for an order for this instrument is max_order_size*(10^-max_order_size_scale). If this equal to zero, there is no limit
max_order_size_scale	The maximum quantity for an order for this instrument is max_order_size*(10^-max_order_size_scale). If this equal to zero, there is no limit

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "symbols"
}
```

Server response:

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "symbols"
}
```

Example symbol status:

```
{
  "seqnum": 1,
  "event": "snapshot",
  "channel": "symbols",
  "symbols": {
    "BTC-USD": {
      "base_currency": "BTC",
      "base_currency_scale": 8,
      "counter_currency": "USD",
      "counter_currency_scale": 2,
      "min_price_increment": 10,
      "min_price_increment_scale": 0,
      "min_order_size": 50,
      "min_order_size_scale": 2,
      "max_order_size": 0,
      "max_order_size_scale": 8,
      "lot_size": 5,
      "lot_size_scale": 2,
      "status": "halt",
      "id": 1,
      "auction_price": 0.0,
      "auction_size": 0.0,
      "auction_time": "",
      "imbalance": 0.0
    },
    "ETH-BTC": {
      "base_currency": "ETH",
      "base_currency_scale": 8,
      "counter_currency": "BTC",
      "counter_currency_scale": 8,
      "min_price_increment": 100,
      "min_price_increment_scale": 8,
      "min_order_size": 220001,
      "min_order_size_scale": 8,
      "max_order_size": 0,
      "max_order_size_scale": 8,
      "lot_size": 0,
      "lot_size_scale": 0,
      "status": "open",
      "id": 3,
      "auction_price": 0.0,
      "auction_size": 0.0,
      "auction_time": "",
      "imbalance": 0.0
    },
    "BTC-EUR": {
      "base_currency": "BTC",
      "base_currency_scale": 8,
      "counter_currency": "EUR",
      "counter_currency_scale": 2,
      "min_price_increment": 10,
      "min_price_increment_scale": 0,
      "min_order_size": 50,
      "min_order_size_scale": 2,
      "max_order_size": 0,
      "max_order_size_scale": 0,
      "lot_size": 0,
      "lot_size_scale": 0,
      "status": "closed",
      "id": 4,
      "auction_price": 0.0,
      "auction_size": 0.0,
      "auction_time": "",
      "imbalance": 0.0
    },
    "ETH-EUR": {
      "base_currency": "ETH",
      "base_currency_scale": 8,
      "counter_currency": "EUR",
      "counter_currency_scale": 2,
      "min_price_increment": 10,
      "min_price_increment_scale": 0,
      "min_order_size": 50,
      "min_order_size_scale": 2,
      "max_order_size": 500,
      "max_order_size_scale": 2,
      "lot_size": 0,
      "lot_size_scale": 0,
      "status": "open",
      "id": 5,
      "auction_price": 0.0,
      "auction_size": 0.0,
      "auction_time": "",
      "imbalance": 0.0
    },
    "ETH-USD": {
      "base_currency": "ETH",
      "base_currency_scale": 8,
      "counter_currency": "USD",
      "counter_currency_scale": 2,
      "min_price_increment": 5,
      "min_price_increment_scale": 0,
      "min_order_size": 50,
      "min_order_size_scale": 2,
      "max_order_size": 0,
      "max_order_size_scale": 0,
      "lot_size": 0,
      "lot_size_scale": 0,
      "status": "open",
      "id": 2,
      "auction_price": 0.0,
      "auction_size": 0.0,
      "auction_time": "",
      "imbalance": 0.0
    }
  }
}
```

Example not halted symbol:

```
{
  "seqnum": 1,
  "event": "updated",
  "channel": "symbols",
  "symbol": "BTC-USD",
  "auction-price": 0,
  "auction-size": 0,
  "auction-time": "",
  "imbalance": 0,
  "status": "open"
}
```

Example halted symbol:

```
{
  "seqnum": 1,
  "event": "updated",
  "channel": "symbols",
  "symbol": "BTC-USD",
  "auction-price": 4500.5,
  "auction-size": 220.125,
  "auction-time": "2230",
  "imbalance": -0.5,
  "status": "halt"
}
```

Ticker

Subscribe to channel:

```
{
  "action": "subscribe",
  "channel": "ticker",
  "symbol": "BTC-USD"
}
```

In order to receive ticker updates, `ticker` channel is available. Subscriptions are again per symbol. The server will send confirmation of the subscription.

Server response:

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "ticker"
}
```

Example ticker snapshot:

```
{
  "seqnum": 8,
  "event": "snapshot",
  "channel": "ticker",
  "symbol": "BTC-USD",
  "price_24h": 4988.0,
  "volume_24h": 0.3015,
  "last_trade_price": 5000.0
}
```

Trades

In order to receive trade updates on executions within each market across the Exchange, you can subscribe to the `trades` channel.

> Subscriptions are again per symbol.

```
{
  "action": "subscribe",
  "channel": "trades",
  "symbol": "ETH-USD"
}
```

> The server will send confirmation of the subscription.

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "trades",
  "symbol": "ETH-USD"
}
```

> Example of a trade update:

```
{
  "seqnum": 21,
  "event": "updated",
  "channel": "trades",
  "symbol": "BTC-USD",
  "timestamp": "2019-08-13T11:30:06.100140Z",
  "side": "sell",
  "qty": 8.5e-5,
  "price": 11252.4,
  "trade_id": "12884909920"
}
```

Authenticated Channels

Authenticated

To authenticate a web socket connection, the client must subscribe to the `auth` channel passing a token field. Alternatively a client can provide the header cookie `auth_token` on a new connection and then authentication will take place immediately without the need to subscribing to `auth` channel.

> If authentication is successful, the server will send an initial notification.

```
{
  "seqnum": 0,
  "event": "subscribed",
  "channel": "auth"
}
```

If authentication was unsuccessful, the server will send the following notification:

```
{
  "seqnum": 0,
  "event": "rejected",
  "channel": "auth",
  "text": "Authentication Failed"
}
```

Trading

The client can submit and cancel orders, as well as receive order executions, through the `trading` channel. The messages are in JSON format and the attribute names are using standardised FIX 4.2 Field names.

Below is a table showing the permitted FIX fields used as a JSON key for cancelling / creating an order:

Tag	Field	Type	Mandatory	Description	Example
11	clOrdID	string	YES	Reference field provided by client and cannot exceed 20 characters	"ABC"
55	symbol	string	YES	Blockchain symbol identifier	"BTC-USD"
40	ordType	string	YES	"market" for market "limit" for limit, "stop" for stop, "stopLimit" for stopLimit	"limit"
59	timeInForce	string	YES	"GTC" for Good Till Cancel, "IOC" for Immediate or Cancel, "FOK" for Fill or Kill, "GTD" Good Till Date	"GTC"
54	side	string	YES	"buy" for Buy, "sell" for Sell	"buy"
38	orderQty	number	YES	The order size in the terms of the base currency	10.23
44	price	number	required for limit and stopLimit order types	The limit price for the order	0.12345
432	expireDate	number	required for GTD orders	expiry date in the format YYYYMMDD	20190318
99	stopPx	number	required for limit and stopLimit order types	Price to trigger the stop order	3500.12
110	minQty	number	Optional for all market orders and for limit orders with IOC timeInForce	The minimum quantity required for an IOC fill	10.0
18	execInst	string	Optional for Limit Orders	The order is placed with Add Liquidity Only (aka Post Only): it will not match liquidity immediately. It will be rejected instead of matching liquidity in the market.	"ALO"

Below is a table representing the different permutations permitted for the FIX fields, with X indicating what can be added and where O is not required:

	Market	Limit	Stop	Stop Limit
clOrdID	X	X	X	X

	Market	Limit	Stop	Stop Limit
symbol	X	X	X	X
ordType	X	X	X	X
timeInForce	O	X	X	X
side	X	X	X	X
orderQty	X	X	X	X
price		X		X
expireDate		O		
stopPx			X	X
minQty	O	O		

The server can include the following additional FIX fields when notifying of an order update:

Tag	Field	Type	Description	Example
35	msgType	string	"8" for ExecutionReport, "9" for OrderCancelRejected	"8"
150	execType	string	"0" for New, "4" for Canceled, "C" for Expired, "8" for Rejected, "F" for partial fill, "A" for Pending, "H" for Trade Break, "I" Order Status	"O"
39	ordStatus	string	‘cancelled’, ‘expired’, ...	
37	orderID	string	The unique order id assigned by the exchange	"11111111"
37	execID	string	Unique identifier for the execution	"123456"
32	lastShares	number	The executed quantity for the order's last fill	0.5678
31	lastPx	number	The executed price for the last fill	3500.12
151	leavesQty	string	For Open and Partially Filled orders this is the remaining quantity open for execution. For Cancelled and Expired orders this is the quantity than was still open before cancellation/expiration. For Rejected order this is equal to orderQty. For other states this is always zero.	10.0
14	cumQty	number	The quantity of the order which has been filled	0.123345
60	transactTime	string	The time the transaction occurred	"2019-08-13T13:15:35.000955868Z"
6	avgPx	number	Calculated the Volume Weighted Average Price of all fills for this order	345.33
1004	tradeId	string	The unique ID assigned to the order fill, also known as trade	"77309432596"

STATE

Enumerated fields are defined as follows:

Name	Description	Example
pending	Order is pending acceptance. Only applicable to stop and stop-limit orders	
open	Order has been accepted	
rejected	Order has been rejected	Limit and market orders can get rejected if you have no balance to fill the order even partially.
cancelled	Order has been cancelled	A market order might get in state cancelled if you don't have enough balance to fill it at market price. Both market orders and limit orders with IOC can have ordStatus 'cancelled' if there is no market for them, even without the user requesting the cancellation.
filled	Order has been filled	A limit order get in state cancelled after the user requested a cancellation.
partial	Order has been partially filled	
expired	Order has been expired	

TYPE

Name	Description
limit	order which has a price limit
market	order that will match at any price available in the market, starting from the best prices and filling up to the available balance
stop	order which has a stop/trigger price, and when that price is reached, it triggers a market order
stopLimit	order which has a stop price and limit price, and when the stop price is reached, it triggers a limit order at the limit price

TIMEINFORCE

To subscribe:

```
{
  "action": "subscribe",
  "channel": "trading"
}
```

To submit & cancel orders as well as receive live order updates, subscribe to the authenticated `trading` channel.

Server response:

```
{
  "seqnum": 1,
  "event": "subscribed",
  "channel": "trading"
}
```

The next message will be a snapshot of live orders for the logged on user"

```
{
  "seqnum": 3,
  "event": "snapshot",
  "channel": "trading",
  "orders": [
    {
      "orderID": "12891851020",
      "clOrdID": "78502a08-c8f1-4eff-b",
      "symbol": "BTC-USD",
      "side": "sell",
      "ordType": "limit",
      "orderQty": 5.0e-4,
      "leavesQty": 5.0e-4,
      "cumQty": 0.0,
      "avgPx": 0.0,
      "ordStatus": "open",
      "timeInForce": "GTC",
      "text": "New order",
      "execType": "0",
      "execID": "11321871",
      "transactTime": "2019-08-13T11:30:03.000593290Z",
      "msgType": 8,
      "lastPx": 0.0,
      "lastShares": 0.0,
      "tradeId": "0",
      "price": 15000.0
    }
  ]
}
```

Name	Description
GTC	Good Till Cancel. The order will rest on the order book until it is cancelled or filled
GTD	Good Till Date. The order will reset on the order book until it is cancelled, filled, or expired
FOK	Fill or Kill. The order is either completely filled or cancelled. No Partial Fills are permitted.
IOC	Immediate or Cancel. The order is either a) completely filled, b) partially filled and the remaining quantity canceled, or c) the order is canceled.

Which TIF are supported. X for supported

	Market	Limit	Stop	Stop Limit
GTC		X	X	X
GTD		X	X	X
IOC		X		
FOK		X		

The `trading` channel supports the following additional actions:

Action	Description
NewOrderSingle	Creates an order
CancelOrderRequest	Cancels an order
OrderMassCancelRequest	Cancel multiple orders
OrderMassStatusRequest	Snapshot of live orders

CANCEL ON DISCONNECT

When subscribing to this channel, users can enable cancel on disconnect. This ensures that when the connection is disconnected, all live orders of the user will be cancelled.

To subscribe:

```
{
  "action": "subscribe",
  "channel": "trading",
  "cancelOnDisconnect": true
}
```

Server response:

```
{
  "seqnum": 1,
  "event": "subscribed",
  "channel": "trading",
  "cancelOnDisconnect": true
}
```

Once enabled, cancel on disconnect cannot be turned off for this connection. Even unsubscribing from trading channel will trigger a cancellation of all live orders.

The next message will be a snapshot of live orders for the logged on user"

```
{
  "seqnum": 3,
  "event": "snapshot",
  "channel": "trading",
  "orders": [
    {
      "orderID": "12891851020",
      "clOrdID": "78502a08-c8f1-4eff-b",
      "symbol": "BTC-USD",
      "side": "sell",
      "ordType": "limit",
      "orderQty": 5.0e-4,
      "leavesQty": 5.0e-4,
      "cumQty": 0.0,
      "avgPx": 0.0,
      "ordStatus": "open",
      "timeInForce": "GTC",
      "text": "New order",
      "execType": "0",
      "execID": "11321871",
      "transactTime": "2019-08-13T11:30:03.000593290Z",
      "msgType": 8,
      "lastPx": 0.0,
      "lastShares": 0.0,
      "tradeId": "0",
      "price": 15000.0
    }
  ]
}
```

Create a new order (NewOrderSingle)

Example creating a GTC limit order:

```
{
  "action": "NewOrderSingle",
  "channel": "trading",
  "clOrdID": "Client ID 3",
  "symbol": "BTC-USD",
  "ordType": "limit",
  "timeInForce": "GTC",
  "side": "sell",
  "orderQty": 10.0,
  "price": 3400.0,
  "execInst": "ALO"
}
```

This action creates an order using the provided fields as described above.

Each time an order changes state or has a match, an event will be sent from the server:

```
{
  "seqnum": 3,
  "event": "updated",
  "channel": "trading",
  "msgType": "8",
  "clOrdID": "Client ID 3",
  "orderID": "999999878",
  "ordStatus": "open",
  "execType": "0",
  "symbol": "BTC-USD",
  "side": "sell",
  "orderQty": 10.0,
  "ordType": "limit",
  "price": 3400.0,
  "transactTime": "2019-08-13T13:09:34.000659345Z",
  "leavesQty": 10.0,
  "cumQty": 0.0,
  "avgPx": 0.0
}
```

The next message will be a snapshot of live orders for the logged on user:

```
{
  "seqnum": 3,
  "event": "snapshot",
  "channel": "trading",
  "orders": [
    {
      "orderID": "12891851020",
      "clOrdID": "78502a08-c8f1-4eff-b",
      "symbol": "BTC-USD",
      "side": "sell",
      "ordType": "limit",
      "orderQty": 5.0e-4,
      "leavesQty": 5.0e-4,
      "cumQty": 0.0,
      "avgPx": 0.0,
      "ordStatus": "open",
      "timeInForce": "GTC",
      "text": "New order",
      "execType": "0",
      "execID": "11321871",
      "transactTime": "2019-08-13T11:30:03.000593290Z",
      "msgType": 8,
      "lastPx": 0.0,
      "lastShares": 0.0,
      "tradeId": "0",
      "price": 15000.0
    }
  ]
}
```

EXAMPLES

Example of a reply if your market order get filled:

```
{
  "seqnum": 5,
  "event": "updated",
  "channel": "trading",
  "orderID": "12891915594",
  "clOrdID": "b50112a2-9851-43ce-a",
  "symbol": "BTC-USD",
  "side": "sell",
  "ordType": "market",
  "orderQty": 0.001,
  "leavesQty": 0.0,
  "cumQty": 0.001,
  "avgPx": 11142.7,
  "ordStatus": "filled",
  "timeInForce": "GTC",
  "text": "Fill",
  "execType": "F",
  "execID": "11451022",
  "transactTime": "2019-08-13T13:50:02.000027480Z",
  "msgType": 8,
  "lastPx": 11142.7,
  "lastShares": 0.001,
  "tradeId": "12884910084"
}
```

Example of a response where your limit order is accepted and resting in the orderbook without fills:

```
{
  "seqnum": 3,
  "event": "updated",
  "channel": "trading",
  "orderID": "12895385711",
  "clOrdID": "ac3f50b0-ec1c-456f-9",
  "symbol": "BTC-USD",
  "side": "buy",
  "ordType": "limit",
  "orderQty": 0.001,
  "leavesQty": 0.001,
  "cumQty": 0.0,
  "avgPx": 0.0,
  "ordStatus": "open",
  "timeInForce": "GTC",
  "text": "New order",
  "execType": "0",
  "execID": "18389438",
  "transactTime": "2019-08-16T11:07:55.000648119Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "price": 10065.0
}
```

If your limit order get rejected you will get:

```
{
  "seqnum": 5,
  "event": "rejected",
  "channel": "trading",
  "text": "Invalid price",
  "clOrdID": "Client ID 3",
  "ordStatus": "rejected",
  "action": "NewOrderSingle"
}
```

or:

```
{
  "seqnum": 5,
  "event": "updated",
  "channel": "trading",
  "orderID": "-1",
  "clOrdID": "71bf645a-6619-499f-9",
  "symbol": "BTC-USD",
  "side": "sell",
  "ordType": "limit",
  "orderQty": 100.0,
  "leavesQty": 0.0,
  "cumQty": 100.0,
  "avgPx": 0.0,
  "ordStatus": "rejected",
  "timeInForce": "GTC",
  "text": "Insufficient Balance",
  "execType": "8",
  "execID": "0",
  "transactTime": "1970-01-01T00:00:00Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "price": 16000.3
}
```

or:

```
{
  "seqnum": 5,
  "event": "updated",
  "channel": "trading",
  "orderID": "0",
  "clOrdID": "12345678901234567890",
  "symbol": "BTC-USD",
  "side": "sell",
  "ordType": "limit",
  "orderQty": 0.001,
  "leavesQty": 0.001,
  "cumQty": 0.0,
  "avgPx": 0.0,
  "ordStatus": "rejected",
  "timeInForce": "GTC",
  "text": "Marketable AOL order",
  "execType": "8",
  "execID": "18331458",
  "transactTime": "2019-08-16T10:39:19.000525614Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "price": 100.3
}
```

If your market order get rejected you will get:

```
{
  "seqnum": 3,
  "event": "updated",
  "channel": "trading",
  "orderID": "-1",
  "clOrdID": "Client ID 3",
  "symbol": "BTC-USD",
  "side": "sell",
  "ordType": "market",
  "orderQty": 100.0,
  "leavesQty": 0.0,
  "cumQty": 100.0,
  "avgPx": 0.0,
  "ordStatus": "rejected",
  "timeInForce": "IOC",
  "text": "Insufficient Balance",
  "execType": "8",
  "execID": "0",
  "transactTime": "1970-01-01T00:00:00Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "minQty": 0.0
}
```

or:

```
{
  "seqnum": 1,
  "event": "updated",
  "channel": "trading",
  "orderID": "4561237891",
  "clOrdID": "Client ID 3",
  "symbol": "BTC-USD",
  "side": "buy",
  "ordType": "market",
  "orderQty": 3.0,
  "leavesQty": 3.0,
  "cumQty": 0.0,
  "avgPx": 0.0,
  "ordStatus": "cancelled",
  "timeInForce": "GTC",
  "text": "Met cash limit",
  "execType": "4",
  "execID": "1111111111",
  "transactTime": "2019-01-01T08:08:08.000888888Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "fee": 0.0
}
```

Cancel an order (CancelOrderRequest)

To cancel order:

```
{
  "action": "CancelOrderRequest",
  "channel": "trading",
  "orderID": "999999878"
}
```

Server response:

```
{
  "seqnum": 18,
  "event": "updated",
  "channel": "trading",
  "orderID": "999999878",
  "clOrdID": "Client ID 3",
  "symbol": "BTC-USD",
  "side": "sell",
  "ordType": "limit",
  "orderQty": 10.0,
  "leavesQty": 10.0,
  "cumQty": 0.0,
  "avgPx": 0.0,
  "ordStatus": "cancelled",
  "timeInForce": "GTC",
  "text": "Canceled by User",
  "execType": "4",
  "execID": "11397697",
  "transactTime": "2019-08-13T13:15:35.000955868Z",
  "msgType": 8,
  "lastPx": 0.0,
  "lastShares": 0.0,
  "tradeId": "0",
  "price": 3400.0
}
```

If the client gives an invalid orderId it will return a rejected cancellation:

```
{
  "seqnum": 18,
  "event": "rejected",
  "channel": "trading",
  "text": "Internal server error"
}
```

Mass order cancel request (OrderMassCancelRequest)

To cancel order:

```
{
  "action": "OrderMassCancelRequest",
  "channel": "trading"
}
```

Server response:

```
{
  "action": "OrderMassCancelRequest",
  "channel": "trading",
  "symbol": "BTC-USD"
}
```

Users have the ability to cancel all of their live orders at once by using this action. A symbol can be optionally specified to reduce the scope of this action. After requesting a mass cancel, execution reports for the affected orders will follow.

Mass order status request (OrderMassStatusRequest)

To cancel order:

```
{
  "action": "OrderMassStatusRequest",
  "channel": "trading"
}
```

Live orders can be listed at any point in time with this request. The subsequent response will contain a snapshot similar to the one received when subscribing to this channel.

Balances

To receive balances for a user, subscribe to the authenticated balances channel:

```
{
  "action": "subscribe",
  "channel": "balances"
}
```

Server response:

```
{
  "seqnum": 1,
  "event": "subscribed",
  "channel": "balances"
}
```

Snapshot of user balances (Zero balances are not sent in the initial snapshot):

```
{
  "seqnum": 2,
  "event": "snapshot",
  "channel": "balances",
  "balances": [
    {
      "currency": "BTC",
      "balance": 0.00366963,
      "available": 0.00266963,
      "balance_local": 38.746779155,
      "available_local": 28.188009155,
      "rate": 10558.77
    },
    {
      "currency": "USD",
      "balance": 11.66,
      "available": 0.0,
      "balance_local": 11.66,
      "available_local": 0.0,
      "rate": 1.0
    },
    {
      "currency": "ETH",
      "balance": 0.18115942,
      "available": 0.18115942,
      "balance_local": 37.289855013,
      "available_local": 37.289855013,
      "rate": 205.84
    }
  ],
  "total_available_local": 65.477864168,
  "total_balance_local": 87.696634168
}
```

Each time the balance changes, a new snapshot event will be sent from the server; this channel has only 'snapshot' event, not 'updates'

```
{
  "seqnum": 19,
  "event": "snapshot",
  "channel": "balances",
  "balances": [
    {
      "currency": "BTC",
      "balance": 0.00266963,
      "available": 0.00166963,
      "balance_local": 28.188009155,
      "available_local": 17.629239155,
      "rate": 10558.77
    },
    {
      "currency": "USD",
      "balance": 22.18,
      "available": 10.52,
      "balance_local": 22.18,
      "available_local": 10.52,
      "rate": 1.0
    },
    {
      "currency": "ETH",
      "balance": 0.18115942,
      "available": 0.18115942,
      "balance_local": 37.289855013,
      "available_local": 37.289855013,
      "rate": 205.84
    }
  ],
  "total_available_local": 65.439094168,
  "total_balance_local": 87.657864168
}
```

REST Endpoints

Query single order status

Example request:

```
GET https://api.blockchain.com/exchange/order/21745988181
```

Response:

```
{
  "orderId": 21745988181,
  "gwOrderId": 2789221636,
  "clOrdId": "d7402f75314a4cf1webd",
  "symbol": "ETH-BTC",
  "ordType": "limit",
  "timeInForce": "GTC",
  "side": "buy",
  "orderQty": 1.0,
  "minQty": 0.0,
  "cumQty": 0.0,
  "price": 0.015797,
  "stopPx": 0.0,
  "ordStatus": "cancelled",
  "expireDate": 20200414,
  "execID": "537374697",
  "avgPx": 0.0
}
```

In order to look up the status of a single order, users can utilize the following endpoint.

Item	Description
URL	https://api.blockchain.com/exchange/order/{orderId}
HTTP Method	GET
Required headers	Cookie: auth_token={apiKey}

Users can authenticate using API keys, by providing a cookie named auth_token with the value set to their API key.

View whitelisted addresses

Example request:

```
GET https://api.blockchain.com/exchange/beneficiaries
```

Response:

```
{
  "capabilities": [
    {
      "currency": "BTC",
      "address": true,
      "xpub": false,
      "existingBeneficiaryOnly": false,
      "fiat": false
    },
    {
      "currency": "ETH",
      "address": true,
      "xpub": false,
      "existingBeneficiaryOnly": false,
      "fiat": false
    }
  ],
  "enabled": true,
  "enabledAt": "2020-05-08T11:41:32.664Z",
  "whitelisted": [
    {
      "id": "298907ac-07df-47ab-93d5-5594e019813b",
      "address": "******************************2v7b",
      "agent": {
        "account": "",
        "address": null,
        "code": null,
        "country": null,
        "name": null,
        "recipient": null,
        "routingNumber": null
      },
      "currency": "BTC",
      "state": "ACTIVE",
      "name": "My Ledger Device",
      "whitelisted": true,
      "fiat": false
    }
  ]
}
```

In order to see the list of whitelisted beneficiaries, users can utilize the following endpoint.

Item	Description
URL	https://api.blockchain.com/exchange/beneficiaries
HTTP Method	GET
Required headers	Cookie: auth_token={apiKey}

Users can authenticate using API keys, by providing a cookie named auth_token with the value set to their API key.

Create A Withdrawal

Example request:

```
POST https://api.blockchain.com/exchange/withdrawals
```

```
{
    "currency": "BTC",
    "amount": "0.0001",
    "beneficiary": "3ff5bc82-b118-45b3-b468-9697be208bdf"
}
```

Response:

```
{
    "id": "1941fcdd-d16a-4b5a-998e-f58f5862af88",
    "user": "e9307d88-bd92-4fdf-840c-7e30fbb0bbd7",
    "product": "MERCURY",
    "amount": {
        "symbol": "BTC",
        "value": "0.0001"
    },
    "fee": {
        "symbol": "BTC",
        "value": "0.0005"
    },
    "state": "NONE"
}
```

In order to create a withdrawal via api, users can utilize the following endpoint.

Item	Description
URL	https://api.blockchain.com/exchange/withdrawals
HTTP Method	POST
Required headers	Cookie: auth_token={apiKey}

Users can authenticate using API keys, by providing a cookie named auth_token with the value set to their API key.

Request Parameters

Parameter	Description
currency	The ticker symbol for the withdrawal currency e.g. "BTC
amount	The amount of the withdrawal as a decimal e.g. 1.231
beneficiary	The id of your beneficiary for this currency, which can be retrieved by using the /beneficiaries endpoint