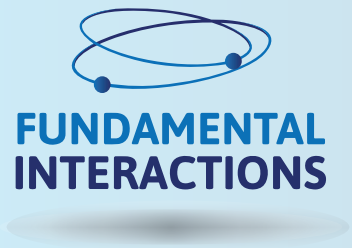

End User Websocket

2022 Jan

TABLE OF CONTENTS

Contents

TABLE OF CONTENTS

Contents

INTRODUCTION

Overview

Welcome to API documentation. These documents outline trading platform functionality, market details, and APIs.

APIs are separated into two categories: Market data feed and trading. Feed APIs provide market data and are public. Trading APIs require authentication and provide access to placing orders and other account information.

Rate Limits

The API is rate limited to prevent abuse that would degrade our ability to maintain consistent API performance for all users.

WebSocket API

Public Endpoints

We throttle public endpoints by IP: 3 requests per second, up to 6 requests per second in bursts. Some endpoints may have custom rate limits.

Private Endpoints

We throttle private endpoints by user ID: 5 requests per second, up to 10 requests per second in bursts. Some endpoints may have custom rate limits.

Time Format

The API timestamp fields us the Epoch/POSIX format for WebSocket API Requests. Unix and POSIX measure time as the number of seconds that have passed since 1 January 1970 00:00:00 UTC.

Do not provide miliseconds. '1614331055' or '1614331055000' are supported. '1614331055123' is not supported.

Websocket API

Connectivity

The WebSocket market data feed provides real-time market data updates for orders and trades.

wss://ws. FI.com:8081

For testing, use:

wss://ws- sandbox.FI.com:8081/

The WebSocket feed uses a bidirectional protocol, which encodes all messages as JSON objects. All messages have a type attribute that can be used to handle the message appropriately.

Please note that new message types can be added at any point in time. Clients are expected to ignore messages they do not support.

In the case of an initial connection failure, ensure to add the required certificates. Please follow the steps detailed in this knowledge base article - Adding the required SSL certificates.

Example Structure: JAVA Script

```
const WebSocket = require('ws');
process.env['NODE_TLS_REJECT_UNAUTHORIZED'] = 0
const ws = new WebSocket("wss://ws-sandbox.FI.com:8081/")

ws.onopen = function() {
    ws.send(JSON.stringify(
      {
        "type": "TYPE",
        "request": "REQUEST"
      }
    ))

    ws.onmessage = function(response) {
      let websocket_response = JSON.parse(response.data);

      console.log(websocket_response)
    }
     ws.close()
  }
}
```

Market Data

Get Symbols

Category: Market Data

Permissions: Public

Endpoint: getsymbols

Retrieves the current assets and trading pairs on the platform.

Response

Key	Value	
security	string.	The trading pair or asset.
fractionbase	string.	The number of decimals supported. e.g 100 = 1.00
tradingsymbol	string.	true if a trading pair, false if an asset
pair_first	string.	Specifies what the first asset is. e.g. LTC vs USD
fiat	string.	true for fiat, false for virtual assets
pair	string.	true if a trading pair, false if an asset

Example Structure: JAVA Script

```
# Request
  {
    "type": "getsymbols"
  }
# Response
  {
    "result":"OK",
    "data":[
      {
        "security":"USD",
        "fractionbase":100,
        "tradingsymbol":false,
        "fiat":false,
        "pair":false
```

```
},
{
  "security":"LTCUSD",
  "fractionbase":10000,
  "pair_first":"LTC",
  "tradingsymbol":true,
  "fiat":false,
  "pair_second":"USD",
  "pair":true
},
{
  "security":"LTC",
  "fractionbase":10000,
  "tradingsymbol":false,
  "fiat":false,
  "pair":false
},
{
  "security":"ETHUSD",
  "fractionbase":100000,
  "pair_first":"ETH",
  "tradingsymbol":true,
  "fiat":false,
  "pair_second":"USD",
  "pair":true
},
{
  "security":"ETH",
  "fractionbase":100000,
  "tradingsymbol":false,
  "fiat":false,
  "pair":false
},
{
  "security":"BTCUSD",
  "fractionbase":100000,
  "pair_first":"BTC",
  "tradingsymbol":true,
  "fiat":false,
  "pair_second":"USD",
  "pair":true
```

```
    },
    {
      "security":"BTC",
      "fractionbase":100000,
      "tradingsymbol":false,
      "fiat":false,
      "pair":false
    },
    {
      "security":"BCHUSD",
      "fractionbase":100000,
      "pair_first":"BCH",
      "tradingsymbol":true,
      "fiat":false,
      "pair_second":"USD",
      "pair":true
    },
    {
      "security":"BCH",
      "fractionbase":100000,
      "tradingsymbol":false,
      "fiat":false,
      "pair":false
    }
  ],
    "type":"getsymbols"
}
```


Subscribe Order Book

Category: Market Data

Permissions: Public

Endpoint: subscribe

Retrieves the latest order book information and then subscribes the user to ongoing market data event updates for the trading pairs specified.

The Subscribe call responds with the response shown below. Messages are then periodically sent in the same format as

this response UpdateEvent information when best-bid/best-offer issue, until you close the connection.

Request

Key	Value		Required
msg	string.	Specify the type of data to subscribe to. use 'book' for order book data.	Yes
security	string.	Specify the trading pair you wish to subscribe to.	Yes
dest	string.	This value will always be set to CROX.	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
side	string.	The order side.
act	string.	The act to Update or Remove orders
price	string.	The price of the order.
qty	string.	The size of the order.
time	string.	The time of the order, in POSIX format.

You can identify multiple trading pairs (security) by using '*'.

Example Structure: JAVA Script

```
# Request

 {
   "type": "subscribe",
   "request": [
     {
       "msg":"book",
       "security":"BCHUSD",
       "dest":"CROX"
     }
   ]
 }
```

```
# Response

# order added

 {
   “security”:”BCHUSD”,
   “books”:[
    {
      “side”:”B”,
      “act”:”U”,
      “src”:”CROX”,
      “price”:”472.33”,
      “qty”:”0.46”,
      “id”:120947223715360,
      “time”:”1626338416747”,
      “mpid”:”ANON”,
      “key”:”NA”
    }
    ],
    “type”:”book”
 }

 # order removed

 {
   “security”:”BCHUSD”,
   “books”:[
    {
      “act”:”R”,
      “id”:120947223715360,
      “time”:”1626338444101”
    }
      ],
       “type”:”book”
 }
```

Subscribe Live Trades

Category: Live Trades

Permissions: Public

Endpoint: subscribe

Retrieves the latest trade information and then subscribes the user to ongoing trade event updates for the specified trading pair.

The Subscribe call responds with the response shown below. Messages are then periodically sent in the same format as this response until you close the connection.

Request

Key	Value		Required
msg	string.	Specify the type of data to subscribe to. use 'trade' for trade history data.	Yes
security	string.	Specify the trading pair you wish to subscribe to.	Yes
dest	string.	This value will always be set to CROX.	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
src	string.	The value will always be set to CROX.
price	string.	The price of the order.
qty	string.	The size of the order.
time	string.	The time of the order, in POSIX format.
type	string.	The nature of record.
Matched	string.	The exchange reference ID of the trade.

You must specify a single trading pair for live trades.

Example Structure: JAVA Script

```
# Request

 {
   “type”: “subscribe”,
   “request”: [
    {
      “msg”:”trade”,
      “security”:”BCHUSD”,
      “dest”:”CROX”
    }
  ]
 }

# Response

 {
   “security”:”BCHUSD”,
   “src”:”CROX”,
   “price”:”471.43”,
   “qty”:”0.06969”,
   “time”:”1626338546538”,
   “type”:”trade”,
   “matchid”:”62YNMDOQ2SPR”
 }
```

Query Trade History

Category: Trade History

Permissions: Public

Endpoint: lshistory

The query returns trade history data will return data from the specified start time. The “total_rec” tells how many records there should be, and “rec_no” tell you the current number of record.

Request

Key	Value		Required
type	string.	Specify the type of data to subscribe to using 'tradehistory' for history data.	Yes
security	string.	Specify the trading pair you wish to subscribe to.	Yes
starttime	string.	Specify the start time in POSIX format	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
execprice	string.	The price of the order.
execqty	string.	The size of the order.
time	string.	The time of the order, in POSIX format.
rec_no int.		The record number.
starttime	string.	The start time of the window.
total_rec	string.	The total records displayed.

You must specify a single trading pair for execution history.

Example Structure: JAVA Script

```
# Request

  {
    "type":"lshistory",
    "security":"ETHUSD",
    "starttime":"1630530000000"
  }
```

```
# Response

{
    “result”: “OK”,
    “security”: “LTCUSD”,
    “data”:
      [
          {
            “security”: “LTCUSD”,
            “rec_no”: 1,
            “execprice”: “182.51”,
            “time”: “1630561535942”,
            “execqty”: “0.5702”
          }
      ],
        “total_rec”: “1”,
        “starttime”: “1630530000000”,
        “type”: “lshistory”
  }
```

Query Aggregate Trade History

Category: Trade History

Permissions: Public

Endpoint: tradehistory

The query returns the trade history in 1-minute block. If that minute has a trade, a corresponding block will be return with summary of that minute high/low/first/last trade price as well as volume and number of trades information.

There can be multiple responses to one query if there are more than 100 entries in response. Each response will carry maximum of 100 entries. “total_rec” tells how many records there should be, and “rec_no” tell you current number of record

Request

Key	Value		Required
type	string.	Specify the type of data to subscribe to using 'tradehistory' for trade history data.	Yes
security	string.	Specify the trading pair you wish to subscribe to.	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
price	string.	The price of the order.
qty	string.	The size of the order.
time	string.	The time of the order, in POSIX format.
numoftrades	string.	The number of trades.
lastttime	string.	The last transaction time.
blocktime	string.	
high price	string.	The highest price of the window.
lowprice	string.	The Lowest price of the window.
rec_no int.		The record number.
starttime	string.	The start time of the window.
lastprice	string.	The last price of the window.
startprice	string.	The start price of the window.
total_rec	string.	The total records displayed.

You must specify a single trading pair for execution history.

Example Structure: JAVA Script

```
# Request

  {
    “type”: “tradehistory”,
    “security”: “LTCUSD”
  }


# Response

  {
    “result”:”OK”,
    “security”:”LTCUSD”,
    “data”:[
      {
        “volume”:”0.9375”,
        “numoftrades”:”1”,
        “lastttime”:”1630497372665”,
        “blocktime”:”1630497360000”,
        “highprice”:”174.69”,
        “lowprice”:”174.69”,
        “rec_no”:1,
        “starttime”:”1630497372665”,
        “lastprice”:”174.69”,
        “startprice”:”174.69”
      }
    ],
      “total_rec”:”1”,
      “type”:”tradehistory”
  }
```

Subscribe Trading Pair Status

Category: Trading Pair Status

Permissions: Public

Endpoint: subscribesymbolstatus

Retrieves any changes to trading pair status. Values can be Halt, PreOpening or Normal.

The subscribesymbolstatus call responds with the responses shown below. Messages are sent anytime the status is updated.

Request

Key	Value		Required
type	string.	Specify the type of data subscribe to using 'subscribesymbolstatus' for symbol status data.	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
time	string.	The time of the order, in POSIX format.
status	string.	The new status for the trading pair.

Example Structure: JAVA Script

```
# Request

  {
    "type": "subscribesymbolstatus"
  }

# Response

  {
    "result":"OK",
    "type":"subscribesymbolstatus"
  }

 # OR
```

```
{
  “data”: [
    {
      “security”: “BCHUSD”,
      “time”: “1623155345213”,
      “status”: “Halt”
    }
  ],
    “type”: “symbolstatus”
}

# OR

{
  “data”: [
    {
      “security”: “BCHUSD”,
      “time”: “1623155345213”,
      “status”: “PreOpening”
    }
  ],
    “type”: “symbolstatus”
}

# OR

{
  “data”: [
    {
      “security”: “BCHUSD”,
      “time”: “1623155345213”,
      “status”: “Normal”
    }
  ],
    “type”: “symbolstatus”
}
```

Subscribe High and Low Price

Category: Market Data

Permissions: Public

Endpoint: subscribehighlow

Subscribes the user to ongoing market data event updates when there is a new high or low for the trading pairs specified. The time window 01:00 AM GST - 12:59 AM GST + 1

The subscribehighlow call responds with the response shown below. Messages are then periodically sent if there is a new high or low price.

Request

Key	Value		Required
security	string.	Specify the trading pair you wish to subscribe to.	Yes

Response

Key	Value	
security	string.	The trading pair being subscribed to.
openingprice	string.	The price at 01:01 AM or price after an opening price calculation.
highprice	string.	The highest execution price.
lowprice	string.	The lowest execution price.
prevcloseprice	string.	The execution price at 01:00 AM

You can identify multiple trading pairs (security) by using "*".

Example Structure: JAVA Script

```
# Request

  {
    “type”: “subscribehighlow”,
    “security”: “*”
  }

# Response

  {
    “security”: “BTCUSD”,
    “prevcloseprice”: “35739.34”,
    “type”: “highlow”
  }

# OR

  {
    “openingprice”: “2388.1”,
    “security”: “ETHUSD”,
    “highprice”: “2388.1”,
    “lowprice”: “2388.1”,
    “prevcloseprice”: “2611.75”,
    “type”: “highlow”
  }

# OR

  {
    “result”: “OK”,
    “security”: “*”,
    “type”: “subscribehighlow”
  }
```

Authentication

Category: Authentication

Permissions: Trading

Once client establishes the WebSocket connection, the client has 30 seconds to complete the login process. Otherwise, the socket will be closed by CoinSquare.

Challenge

First client needs to send a challenge request to the system.

Response

Key	Value	
Key	string.	The string used to encrypt the password

The response contains a "key" field, which contains a public key string the client must use to encrypt the password field. The "key" field is a 64 base ASCII print out of the binary byte array for the corresponding public key of asymmetric public/private key pair. CoinSquare will hold the private key while sending clients the public key. Client will first convert the ASCII back to binary byte array. Then use that binary byte array to create a "RSA" public key instance. Afterwards, client will use that RSA public key instance to encrypt info required in this order entry API, e.g. password encryption.

You need to use npm package to install jsencrypt library if you are using JavaScript.

Example Structure: JAVA Script

```
# Request

  {
    "type": "challenge"
  }

# Response

  {
    "result":"OK",
    "type":"challenge",
```

```
"key":"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCvBXTe278Lwg2MoI7iGKolSYuF
+sNFKrsZplxCN9x0kItU3KIf8+1q60ILLwLewCEf7foxzpWp32j9YYU9vNBghuJ7BHcDYTffTRcv
+QdNno491j701Hq7DIw13AGCQQTRcnfclvblnytIEWoQsiUvPJcdiWgqJIX3IQGA47a+uwIDAQAB"
  }

global.window = {};
  const JSEncrypt = require("jsencrypt");

  key = response['key'];
  const encrypt = new JSEncrypt();
  encrypt.setPublicKey(key);
  let sign = encrypt.encrypt('YOUR PASSPHRASE');
```

Login

After encryption of password, the next step is to send in the login request. The string in the "pass" field is from the output result in sample above.

Key	Value		Required
userid	string.	The username of the user.	Yes
pass	string.	The pass phrase generated from the challenge and steps above.	Yes

Response

Key	Value	
result	string.	The response will either contain OK if successful or a rejection reason.

CoinSquare will send current open orders and position information upon login. If a user places a new order, CoinSquare will send an order update automatically. If the user trades, they will receive a trade update and corresponding position updates.

Example Structure: JAVA Script

```
# Request
 {
   “type”:”login”,
   “userid”:”USERNAME”,

“pass”:”s7UW26iGE/iVfk2ihPYIcyzRqZRi/Ztb23UNMomf3xrBzGKUHKzfNwZe5PIR/0zvfevYvkJnKLQVhR4U9/
kObD/Ir0z6mBfLLgFw
EcRm08jYI/nk7lDU+W32PqduTOCThlkXYueQslK54vR9rKvMs=”
 }

# Response - You will get a ‘result’:’OK’

 {
   “result”:”OK”,
   “firm”:”COIN”,
   “need2FA”:true,
   “roles”:”OOOOO”,
   “active”:”Y”,
   “secondary_account”:”TESEGY0101PGHA12345”,
   “type”:”login”,
   “attr”:
     {
       “country”:””,
       “tax_code”:”SR”,
       “last_name”:”USER”,
       “first_name”:”USER “,
       “email”:”user@mail.com”
     },
       “use2fa”:”Y”,
       “userid”:”user@mail.com”,
 }

# Any open orders will be returned as follows

 {
   “refno”:”62YNMDOQ2SPSC0”,
   “side”:”B”,
```

```
  "orderstatus":"Open",
  "type":"order",
  "userid":"user@mail.com",
  "execamount":"0",
  "tif":"GTC",
  "firm":"COIN",
  "security":"BCHUSD",
  "price":"471.43",
  "liveqty":"0.49031",
  "qty":"0.49031",
  "updtime":"1626342626369",
  "enttime":"1626342626366",
  "category":"STAGE",
  "execqty":"0",
  "account":"user@mail.com",
  "ordertype":"LMT"
}

# Your current positions will also be returned

{
  "firm":"COIN",
  "security":"USD",
  "curpos":"1013057.59",
  "cost":"-841663670.1899977",
  "type":"position",
  "account":"user@mail.com"
}

{
  "firm":"COIN",
  "security":"BCH",
  "curpos":"128.25376",
  "cost":"12008212.15",
  "type":"position",
  "account":"user@mail.com"
}
```

```
  {
    “firm”:”COIN”,
    “security”:”LTC”,
    “curpos”:”121.7144”,
    “cost”:”1199785.55”,
    “type”:”position”,
    “account”:”user@mail.com”
  }

# OR

  {
    “result”:”invalid user/password”,
    “type”:”login”
  }
```


Logout

This is for clean logout. Once CoinSquare receives this message, it will immediately terminate the WebSocket connection.

Example Structure: JAVA Script

```
# Request
  {
    “type”:”logout”,
  }

# Response

  {
    “result”:”OK”,
    “type”:”logout”
  }
```

Order Placement

Category: Trading

Permissions: Trading

You can place two types of orders: limit and market. Orders can only be placed if your account has sufficient funds.

Addorder

You can place two types of orders: limit and market. Orders can only be placed if your account has sufficient funds.

Request

Key	Value		Required
security	string.	The trading pair you wish to place the order in.	Yes
side	string.	The order side. Options are 'B' or sell 'S'.	Yes
price	string.	Specify the order price. Required for limit orders Conditionally	
amount	string.	Specify the amount in fiat for the order. Required for market buy orders.	Conditionally
ordertype	string.	Specify if sending a limit or market order. Options are 'LMT' or 'MKT'. Default is 'LMT'.	
tif	string.	Options are GTC, IOC, FOK, GTT. GTC is the default	No

Response

Key	Value		Required
security	string.	The trading pair you wish to place the order in.	Yes
side	string.	The order side. Options are buy 'B' or sell 'S'.	Yes
qty	string.	Specify the quantity of order. Required for all orders except market buy orders.	
price	string.	Specify the order price. Required for limit orders	
amount	string.	Specify the amount in fiat for the order. Required for market buy orders.	
ordertype	string.	Specify if sending a limit or market order. Options are 'LMT' or 'MKT'. Default is 'LMT'.	
tif	string.	Options are GTC, IOC, FOK, GTT. GTC is the default	No
stoptype	string.	Used to place a stop limit ot stop market order by setting value to 'STOP'. Default is not set.	No

Key	Value		Required
stopprice	string.	Specifiy the stop price if stop type is set.	Conditionally
exptime	string.	The expiry time of the order, in POSIX format. Required if TIF is set to GTT.	
alo	boolean.	The alo (Add Liquidity Only) field can be set to true for post/maker only orders	No
clientorderid	string.	You can pass your own client order id which will be echoed back in corresponding “order”,” sale” updates.	

Rejected orders will be displayed with reasons. Please refer to the Reject Messages section for the various types of error messages returned when triggered.

Example Structure: JAVA Script

```
# Request - LTCUSD Buy Market Order IOC order for 180 dollars.

 {
   “type”:”addorder”,
   “security”:”LTCUSD”,
   “side”:”B”,
   “ordertype”:”MKT”,
   “tif”:”IOC”,
  “amount”:”180”
 }


# Response

  # If the order is places successfully you will as “result”:”OK” response , followed by any
    order , execution and position updates.


 {
   “tif”:”IOC”,
   “result”:”OK”,
   “firm”:”COIN”,
   “refno”:”62YNMDOQ2SPIC0”,
   “security”:”LTCUSD”,
   “side”:”B”,
   “amount”:180,
   “type”:”addorder”,
   “userid”:”user@mail.com”,
   “ordertype”:”MKT”
 }
```

```
{
  “refno”:”62YNMDOQ2SPIC0”,
  “side”:”B”,
  “amount”:”180”,
  “orderstatus”:”Open”,
  “type”:”order”,
  “userid”:”user@mail.com”,
  “execamount”:”0”,”tif”:”IOC”,
  “firm”:”COIN”,
  “security”:”BCHUSD”,
  “price”:”0”,
  “liveqty”:”0.00001”,
  “qty”:”0.00001”,
  “updtime”:”1626332766266”,
  “enttime”:”1626332766266”,
  “category”:”STAGE”,
  “execqty”:”0”,
  “account”:”user@mail.com”,
  “ordertype”:”MKT”
}

{
  “firm”:”COIN”,
  “security”:”LTC”,
  “curpos”:”0.38031”,
  “cost”:”17999691.99”,
  “type”:”position”,
  “account”:”user@mail.com”
}

{
  “traderefno”:”62YNMDOQ2SPIC02_62YNMDOQ2SPJ”,
  “refno”:”62YNMDOQ2SPIC0”,
  “trdtime”:”1626332766360”,
  “side”:”B”,
 “ismaker”:false,
  “execprice”:”183.29”,
  “type”:”sale”,
  “userid”:”user@mail.com”,
  “firm”:”COIN”,
  “security”:”LTCUSD”,
  “exchangeref”:”62YNMDOQ2SPJ”,
  “category”:”CROX”,
  “execqty”:”0.38031”,
```

```
  "account":"user@mail.com"
}

{
  "firm":"COIN",
  "security":"USD",
  "curpos":"622.09",
  "cost":"-11216742.005",
  "type":"position",
  "account":"user@mail.com"
}

{
  "refno":"62YNMDOQ2SPIC0",
  "side":"B",
  "amount":"180",
  "orderstatus":"Executed",
  "type":"order",
  "userid":"user@mail.com",
  "execamount":"179.9969199",
  "tif":"IOC",
  "firm":"COIN",
  "security":"LTCUSD",
  "price":"0",
  "liveqty":"0",
  "qty":"0.38031",
  "updtime":"1626332766920",
  "enttime":"1626332766266",
  "category":"STAGE",
  "execqty":"0.38031",
  "account":"user@mail.com",
  "ordertype":"MKT"
}
```


If the order is not submitted successfully, you will see the reject reason

in the result field.

Post/Maker order entered at price that will execute

```
{
    “tif”:”GTC”,
    “result”:”order marketable to existing quote/book”,
    “firm”:”COIN”,
    “security”:”LTCUSD”,
    “side”:”B”,
    “alo”:true,
    “price”:”173.29”,
    “qty”:”0.29969”,
    “type”:”addorder”,
    “userid”:”user@mail.com”,
    “ordertype”:”LMT”,
    “clientorderi”: “7599bb88-1d48-48b1-bf90-ed20d81b344d”
  }

  {
    “tif”:”GTC”,
    “result”:”order marketable to existing quote/book”,
    “firm”:”COIN”,
    “security”:”LTCUSD”,
    “side”:”B”,
    “alo”:true,
    “price”:”173.29”,
    “qty”:”0.29969”,
    “type”:”addorder”,
    “userid”:”user@mail.com”,
    “ordertype”:”LMT”
  }
```


Passing incorrect field


```
  {
    “result”:”cannot use qty for market buy order”,
    “tif”:”IOC”,
    “ordertype”:”MKT”,
    “security”:”LTCUSD”,
    “side”:”B”,
    “qty”:”180”,
    “type”:”addorder”
  }
```

Cancelorder

To cancel an open order, you can use the cancelorder endpoint. You must pass the Exchange assigned 'refno' to specify what order to cancel.

Request

Key	Value		Required
security	string.	The trading pair you wish to cancel the order for.	Yes
refno	string.	The Exchange generated order id.	Yes

Response

Key	Value		Required
result	string.	The response will either contain OK if successful or a rejection reason.	
security	string.	The trading pair you wish to cancel the order for.	-
refno	string.	The Exchange generated order id.	-

Order & Trade Query

Example Structure: JAVA Script

```
# Request

  {
    "type":"cancelorder",
    "security":"BTCUSD",
    "refno":"5FSDIAGCE768A0"
  }

# Response

  {
    "result":"OK",
    "refno":"5FSDIAGCE768A0",
    "security":"BTCUSD",
    "type":"cancelorder"
  }
```

```
# OR

  {
    “result”:”order not found”,
    “refno”:”5FSDIAGCE768A0”,
    “security”:”BTCUSD”,
    “type”:”cancelorder”
  }

# OR

  {
    “result”:”failed”,
    “refno”:”5FSDIAGCE768A0”,
    “security”:”BTCUSD”,
    “type”:”cancelorder”
  }
```

Order & Trade Query

Category: Trading

Permissions: Trading

Users can retrieve open and executed order information

Queryorder

Users can retrieve open and executed order information.

Request

Key	Value		Required
security	string.	The trading pair you wish to query orders for.	Yes
refno	string.	The Exchange generated order id.	Yes

Response

Key	Value		Required
updtime	string.	Time of the last order update, in POSIX format.	-

Key	Value		Required
tif	string.	The order's time in force.	-
execamount	string.	The amount of the order that has executed in fiat value.	-
qty	string.	The order's original quantity.	-
security	string.	The trading pair.	-
liveqty	string.	The order's remaining open quantity.	-
refno	string.	The Exchange generated order id.	-
execqty	string.	The order quantity that has been executed.	-
side	string.	The order side	-

Example Structure: JAVA Script

```
# Request

  {
    "type":"queryorder",
    "security":"BTCUSD",
    "refno":"5GS08WTIE8CWA0"
  }

# Response

  {
    "result":"OK",
    "refno":"5GS08WTIE8CWA0",
    "data": {
      "updtime":"1557505235085",
      "tif":"GTC",
      "execamount":"10000",
      "qty":"1000",
      "security":"BTCUSD",
      "type":"order",
      "clientorderid":"ID1",
      "liveqty":"900",
      "category":"CROX",
      "refno":"5GS08WTIE8CWA0",
      "price":"100",
      "execqty":"100",
      "side":"S"
    },
    "security":"BTCUSD",
    "type":"queryorder"
  }
```

```
# OR

  {
    "result":"order not found",
    "refno":"5GS08WTIE8CWA0",
    "security":"BTCUSD",
    "type":"queryorder"
  }
```

Openorderquery

This query for getting open orders for all Assets all or certain Asset.

Request

Key	Value		Required
Security	string.	The trading pair you wish to query trades for.	No

Response

"total_rec" tells you how many records there are available.

If there are more than 100 records, the system will send you multiple response, each will have 100 records.

Key	Value		Required
Security	string.	The trading pair.	-
Refno	string.	The Exchange generated order id.	-
Execqty	string.	The order quantity that has been executed.	-
Rec_no	int.	The record number of the response.	-
Side	string.	The order side	-
price	string.	The price of the order.	-
qty	string.	The size of the order.	-
ordertype	string.	Specify if sending a limit or market order. Options are 'LMT' or 'MKT'. Default	
orderstatus	string.	The status of the order (Open, Executed, Rejected, Cancelled)	-
commrate	string.	The comission rate (E.G If the comission rate is "0 42" this meaning 0 for maker , 42 basis points for taker).	
liveqty	string.	The order's remaining open quantity.	-

Key	Value		Required
updtime	string.	Time of the last order update, in POSIX format.	-
enttime	string.	The time order was entered in POSIX format.	

Example Structure: JAVA Script

```
# Request

  {
    "type":"openorderquery",
    "security":"ETH"
  }

  # OR to Query all open orders for all Assets

  {
    "type":"openorderquery",
    "security":"*"
  }

# Response

  {
   "result": "OK",
    "security": "*",
    "data": [
      {
          "refno": "66ZHSAPJYL3DC1",
          "side": "B",
          "orderstatus": "Open",
          "type": "order",
          "userid": "user@mail.com",
          "execamount": "0",
          "tif": "GTC",
          "firm": "COIN",
          "commrate": "0|42",
          "security": "ETHUSD",
          "price": "4000",
          "liveqty": "1",
          "rec_no": 1,
          "qty": "1",
          "updtime": "1638860161388",
          "enttime": "1638860161388",
          "category": "STAGE",
          "execqty": "0",
          "account": "user@mail.com",
          "ordertype": "LMT"
      },
```

```
        {
            “refno”:”66ZHSAPJYPOFC0”,
            “side”: “B”,
            “orderstatus”: “Open”,
            “type”: “order”,
            “userid”: “user@mail.com”,
            “execamount”: “0”,
            “tif”: “GTC”,
            “firm”: “COIN”,
            “commrate”: “0|42”,
            “security”: “BTCUSD”,
            “price”: “50000”,
            “liveqty”: “1”,
            “rec_no”: 2,
            “qty”: “1”,
            “updtime”: “1638860142071”,
            “enttime”: “1638860142071”,
            “category”: “STAGE”,
            “execqty”: “0”,
            “account”: “user@mail.com”,
            “ordertype”: “LMT”
        }
      ],
      “total_rec”: 2,
      “type”: “openorderquery”
    }

    # OR if there are no records

    {
     “result”: “OK”,
     “total_rec”: 0,
     “type”: “openorderquery”
    }
```


Querytrade

This is to query the trades for a user, or specify certain amount of order query you wish to get and from specific time.

Request

Key	Value		Required
Security	string.	The trading pair you wish to query trades for.	No
maxreturn	string.	Specify the maximum amount of orders returned.	No
lastfirst	string.	Specify that you wish to return last orders.	No

Key	Value		Required
userid	string.	Specify the user id	No
Fromtime	string.	Specify the cut off time in POSIX format.	No

Response

"total_rec" tells you how many records there are available.

If there are more than 100 records, the system will send you multiple response, each will have 100 records.

Key	Value		Required
Trdtime	string.	The execution time, in POSIX format.	-
Execprice	string.	The execution price of the trade.	-
Security	string.	The trading pair.	-
Refno	string.	The Exchange generated order id.	-
Traderefno	string.	The Exchange generated execution id.	-
Execqty	string.	The order quantity that has been executed.	-
Rec_no	int.	The record number of the response.	-
Side	string.	The order side	-

Example Structure: JAVA Script

```
# Request

  {
    "type":"querytrade",
    "security":"LTCUSD",
    "fromtime":"1557323619303"
  }

  # OR

  {
    "type": "querytrade",
    "maxreturn": 1000,
    "lastfirst": true,
    "userid": "*",
    "fromtime": "1633305600000"
  }
```

Response


```
{
  "result":"OK",
  "data":[
    {
      "account": "user@mail.com",
      "refno":"5GQBE9BCZOO0A3",
      "category":"CROX",
      "comm": "0.00001",
      "commsecurity": "LTC",
      "exchangeref": "648KTXPYN46U",
      "execqty":"10",
      "execprice":"30",
      "ismaker": false,
      "traderefno":"5GQBE9BCZOO0A32_5GQBE9BCZOO2",
      "side":"B",
      "trdtime":"1557323619303",
      "rec_no":1,
      "security":"LTCUSD",
      "type":"sale"
    },
    {
      "account": "user@mail.com",
      "refno":"5GQBE9BCZN4GA7",
      "category":"CROX",
      "exchangeref": "648KTXPYN46Z",
      "execqty":"10",
      "execprice":"100.5",
      "ismaker": true,
      "traderefno":"5GQBE9BCZN4GA72_5GQBE9BCZN4I",
      "side":"B",
      "trdtime":"1557323603427",
      "rec_no":2,
      "security":"LTCUSD",
      "type":"sale"
    }
  ],
  "type":"querytrade",
  "total_rec":2
}
```

Trading Transactions

Category: Tracking Transactions

Permissions: Trading

Querydeposit

This is to query the transactions history for a user.

Request

Key	Value		Required
maxreturn	string.	Specify the maximum amount of orders returned.	No
lastfirst	string.	Specify that you wish to return last orders.	No
userid	string.	Specify the user id	No
Fromtime	string.	Specify the cut off time in POSIX format.	No

Response

"total_rec" tells you how many records there are available.

If there are more than 100 records, the system will send you multiple response, each will have 100 records.

Key	Value		Required
time	string.	The time of transaction.	-
security	string.	The trading pair.	-
type	string.	The type of Transaction	-
traderefno	string.	The Exchange generated execution id.	-
amount	string.	the amount of Cryptocurrency transferred.	-
hash	int.	The transaction id.	-
Rec_no	int.	The record number of the response.	

Example Structure: JAVA Script

```
# Request

 {
   “type”:”querydeposit”,
   “maxreturn”: 1000,
   “lastfirst”: “true”,
   “fromtime”: “1639406640393”,
   “userid”: “user@mail.com”
 }

# Response

 {
   “result”:”OK”,
   “data”:
   [
     {
       “firm”:”TFF1”,
       “traderefno”:”STAGEWEBSOCKET:user@mail.com:1637313057457:1052”,
       “security”:”BTC”,
       “amount”:”0.00001”,
       “rec_no”:1,
       “time”:”1637313057457”,
       “type”:”withdraw”,
       “userid”:”user@mail.com”,
       “account”:”user@mail.com”,
       “hash”:”STAGEWEBSOCKET:user@mail.com”
     }
   ],
   “total_rec”: 1,
   “type”: “querydeposit”
 }
```

Account Balances

Category: Balances and Positions

Permissions: Trading

Querypos

Users can receive balance information / positions and continuous updates afterwards.

Request

Key	Value		Required
security	string.	The virtual asset or currency to query for.	No

Response

Key	Value		Required
curpos	string.	The amount of that Virtual Asset or Viat in the users balance	-
security	string.	The virtual asset or currency to query for.	

Example Structure: JAVA Script

```
# Request

  {
    “type”:”querypos”,
    “security”:”LTC”
  }

# Resposne

  {
    “result”:”OK”,
    “firm”:”COIN”,
    “data”: [
      {
        “curpos”:”300”,
        “rec_no”:1,
        “security”:”LTC”,
        “type”:”position”
      },
      {
        “curpos”:”300”,
        “rec_no”:2,
        “security”:”USD”,
        “type”:”position”
```

```
        }
      ],
      "userid":"USER",
      "type":"querypos",
      "total_rec":2
    }
  # OR

    {
      "result":"no position",
      "type":"querypos"
    }
```

Querypendingpos

Note: This endpoint does not auto update and has to be manually queried. Only querypos will continuously update when balance changes.

Request

Key	Value		Required
security	string.	The virtual asset or currency to query for.	No

Response

Key	Value		Required
curpos	string.	The amount of that Virtual Asset or Viat in the users balance. Includes managedpos	-
security	string.	The virtual asset or currency to query for.	-
rec_no	int.	The record number.	-
managedpos	string.	The withheld balance. The amount withheld in open orders/	-
openpos	string.	The field can be ignored and will always be zero.	-

Reject Messages

Example Structure: JAVA Script

```
# Request

 {
   “type”:”querypendingpos”
 }

# Resposne

 {
   “result”: “OK”,
   “data”: [
     {
       “security”: “ETH”,
       “curpos”: “1.13361”,
       “rec_no”: 1,
       “managedpos”: “0”,
       “type”: “position”,
       “openpos”: “0”
     },
     {
       “security”: “BCH”,
       “curpos”: “0.14721”,
       “rec_no”: 2,
       “managedpos”: “0.1”,
       “type”: “position”,
       “openpos”: “0”
     },
     {
       “security”: “USD”,
       “curpos”: “1269.59”,
       “rec_no”: 3,
       “managedpos”: “0”,
       “type”: “position”,
       “openpos”: “0”
     }
   ],
   “total_rec”: 3,
   “type”: “querypendingpos”
 }

# OR

 {
   “result”:”no position”,
   “type”:”querypendingpos”
 }
```

Reject Messages

You may receive different reject messages while attempting to place orders. The following is a list of error messages that are returned by the system when risk rejects are triggered.

Order marketable to existing quote/book

This error message indicates that you are attempting to place orders that will execute against other orders in the order book in the case of post only orders (add only orders).

Sample

{“tif”:”GTC”,”result”:”order marketable to existing quote/book”,”firm”:”COIN”,”security”:”BCHUSD”,”side”:”B”,”alo”:true,”price”:”473.29”,”qty”:”0.29969”,”type“:”addorder”,”userid”:”user@mail.com”,”ordertype”:”LMT”}

Exceed order/sec limit

This error message indicates that you are attempting to place orders beyond the maximum order quantity per second (more than 4 orders per second).

Sample

{‘tif’: ‘GTT’, ‘result’: ‘exceed order/sec limit’, ‘exptime’: ‘1631004134000’, ‘security’: ‘ETHUSD’, ‘side’: ‘B’, ‘alo’: ‘true’, ‘price’: ‘3782.01’, ‘qty’: ‘0.0825’, ‘type’: ‘addorder’, ‘clientorderid’: ‘7599bb88-1d48-48b1-bf90-ed20d81b344d’}

No crypto pos

This error message indicates that you are attempting to place orders with insufficient virtual asset balance.

Sample

{“tif”:”IOC”,”result”:”no crypto pos”,”security”:”BTCUSD”,”side”:”B”,”amount”:”5100”,”type”:”addorder”,”ordertype”:”MKT”}
{“refno”:”64GSC7W0XIY8C0”,”side”:”B”,”amount”:”5100”,”orderstatus”:”Rejected”,”type”:”order”,”userid”:”user@mail.com”,”execamount”:”0”,”tif”:”IOC”,”firm”:”9737ARECYP”,”security”:”BTCUSD”,”price”:”0”,”liveqty”:”0”,”qty”:”0”,”updtime”:”1631017300225”,”text”:”no crypto pos”,”enttime”:”1631017300225”,”category”:”STAGE”,”execqty”:”0”,”account”:”user@mail.com”,”ordertype”:”MKT”}

Wash sale prohibited

This error message indicates that you are attempting to place an order that will execute against another order you have placed on the opposite side of the order book.

Sample

{“refno”:”64GSC7W0XHR8C0”,”side”:”B”,”amount”:”510”,”orderstatus”:”Rejected”,”type”:”order”,”userid”:”user@mail.com”,”execamount”:”0”,”tif”:”IOC”,”firm”:”9737ARECYP”,”security”:”BTCUSD”,”price”:”0”,”liveqty”:”0”,“qty”:”0”,”updtime”:”1631016705668”,”text”:”wash sale prohibited”,”enttime”:”1631016705668”,”category”:”STAGE”,”execqty”:”0”,”account”:”user@mail.com”,”ordertype”:”MKT”}

Exceeds margin limit Exceed total Amount

This error message indicates that you are attempting to place an order that is above the maximum daily notional value.

Sample

{“tif”:”GTC”,”result”:”Exceeds margin limit Exceed total Amount”,”security”:”BTCUSD”,”side”:”B”,”price”:”46500”,”qty”:”100”,”type”:”addorder”,”ordertype”:”LIMIT”}
{“refno”:”64IEBHMP914LC0”,”side”:”B”,”orderstatus”:”Rejected”,”type”:”order”,”userid”:”user@mail.com”,”execamount”:”0”,”tif”:”GTC”,”firm”:”COIN”,”security”:”BTCUSD”,”price”:”46500”,”liveqty”:”0”,”qty”:”100”,”updtime”:”1631175016204”,”text”:”Exceeds margin limit Exceed total Amount”,”enttime”:”1631175016204”,”category”:”STAGE”,”execqty”:”0”,”account”:”user@mail com”,”ordertype”:”LMT”}

Error parsing json

This error message indicates that there are typographical errors in your request.

Sample

```
{“result”:”error parsing json”}
```

Notes

FUNDAMENTAL INTERACTIONS

Sales & Development Office
147 W. 26th Street. 300
New York, NY 10001
Office: (212) 725-3509 | Corporate: 212-845-9077
sales@fundamentalinteractions.Com

Support Office
Hudson Street, Hoboken, NJ 07030
Support Phone: 888-851-1369
support@fundamentalinteractions.com

Development Office
Development Office
1500 District Avenue, 2nd Floor, Burlington,
MA 01803

Europe Development & Support Office
Kharkov, Ukraine
61000