ErisX Websocket API V3.4
Market Data and Order Entry

ErisX

Please contact ErisX sales or technical.onboarding@erisx.com for more information.

Contents

ErisX

1 Change History

Date	Message(s) or Section	Description
20190816		Version 1.0
20190930	Authentication	A number of changes to better describe the authentication method.
20191016	MarketDataSubscribe Trade Only	Trade Only flag format is boolean not string.
20191020	ReplaceLimitOrderSingleRequest and ReplaceStopLimitOrderSingleRequest	Change handlInst parameter from AutomatedExecutionOrderPublic to AutomatedExecutionOrderPrivate
20191021	MarkeDataSubscribe Trade Only	Upon connection a response will be included with the last trade information.
20191203		Version 3.0
	Futures Specific Functionality	Updated to Include details for Futures
20200316		Version 3.1
	API credentials, Authentication, Post-Only	Added new API permissions layout for Authentication Removed python2 example Added new Post-Only order type
20200519		Version 3.2
	Security Status	Added new SecurityStatus message workflow
	Market Data Response fields	Add new endFlag and numberOfOrders fields
20200617		Version 3.3
	Cancel All Orders	Add support for new message type to cancel all working orders for a partyID
	Order Cancellation and Order Modification	Separate the order modification and order cancellation section into two separate sections
20200721		Version 3.4
	Security List	Add productCode, securityGroup, cap and floor Add securityGroup field in SecurityList request
	Execution Reports	Add AvailableBalanceData component with AvailableBalance and AvailableBalanceCurrency
	Order History	Add AvailableBalanceData component with AvailableBalance and AvailableBalanceCurrency. VIEW DISCLAIMER

ErisX

2 General

This API service enables Clearing Members to subscribe to real-time market data, enter and manage orders through a WebSocket connection. All requests and responses are application/json content type.

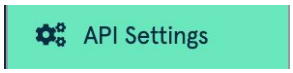
All Order Entry messages are private and every request needs to be signed using the authentication method described.

2.1 WebSocket API Endpoint URL

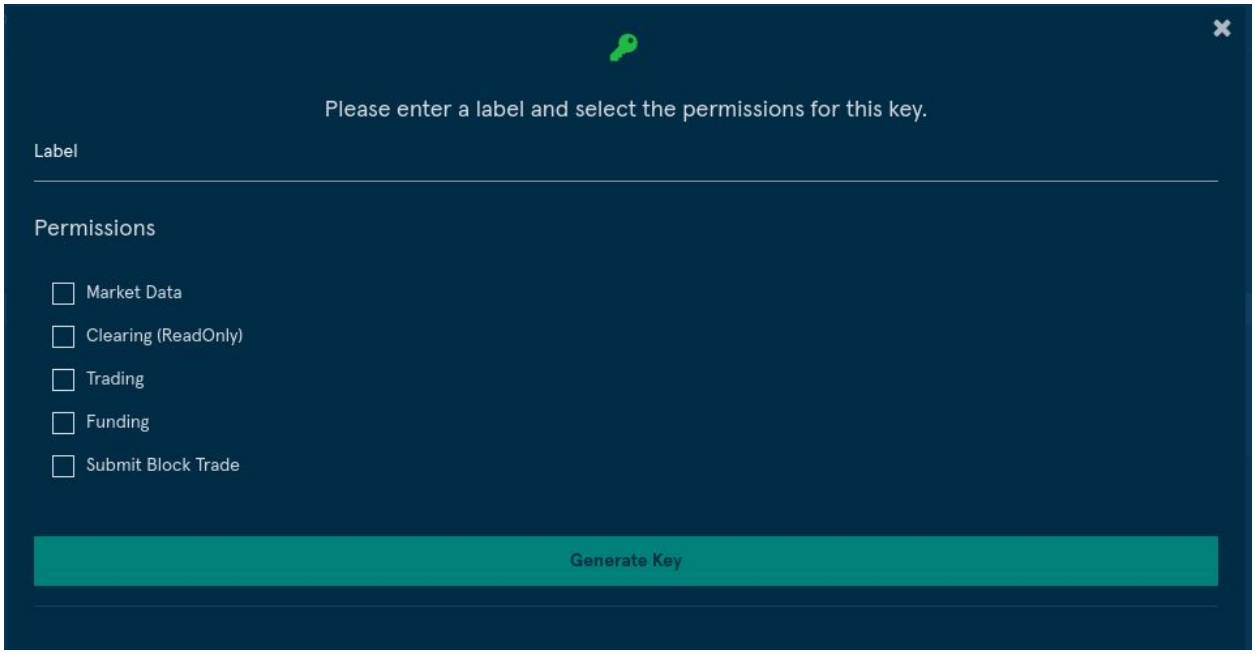
- **Testing - wss://trade-api.newrelease.erisx.com/**
- **Production - wss://trade-api.erisx.com/**

2.2 API Credentials

In order to sign your API requests, you will need to create a set of API Credentials.

From the Eris member Portal, navigate to the dropdown next to your username in the top right of the page and select 

. After clicking **Create New API Key** you will be asked to select the permissions you want to enable.



API Key permissions

- **Market Data:** An API key can query historical data or subscribe to real time data.
- **Trading:** Allows an API key to enter, modify and cancel orders.
- **Clearing (ReadOnly):** Allows an API key to query information about their clearing accounts (Documented Separately).

ErisX

- **Funding:** Allows an API key to initiate withdrawal requests (Documented Separately).
- **Submit Block Trade:** Allows an API key to submit Block Trades (Documented Separately).

When ready click **Generate Key** and you will be presented with two pieces of information that must be kept safe as they will be needed for authentication of calls to the end points and will not be shown again:

- **API key**
- **Secret**

2.3 Authentication

A JSON web token should be generated using the HS256 algorithm on the API key, secret and timestamp as described in the examples below. This token will be used in the authentication request message.

- **Timestamp:** The authentication token requires a Unix Epoch timestamp.
- **Token Age:** Each token will only be valid for 60 seconds after the timestamp.
- **Header:** The authentication token must include the header information describing the algorithm and token type. This header is automatically created in most jwt libraries. The following link Example: `{"typ":"JWT", "alg":"HS256"}`

Notes:

- In python use the **pyjwt** package to generate the token (https://pyjwt.readthedocs.io/en/latest/).
- In python 3 you will need to use the **decode('utf-8')** function to convert the token from a bytes like object to a string.

Javascript Example:

```
var jwt = require('jsonwebtoken');
var apiKey = '9106676d85f1163f.d1ba2efac8bc1e0a';
var secret = '31b6b61606588580';
var payload = {
 iat: Date.now(),
 sub: apiKey
};
var token = jwt.sign(payload, secret, { algorithm: 'HS256'});
```

Python 3 Example:

```
import jwt
import time

def gen_token(secret, api_key):
   unix_timestamp = int(round(time.time()))
   payload_dict = {'sub': api_key, 'iat': unix_timestamp}
   return jwt.encode(payload_dict, secret, algorithm='HS256').decode('utf-8')
```

ErisX

```
my_secret = '31b6b61606588580'
my_api_key = '9106676d85f1163f.d1ba2efac8bc1e0a'
token = gen_token(my_secret, my_api_key)
```

Upon creation of the connection to the websocket, an authentication request is required in order to enable the authorization to make any further requests or subscriptions for Market Data.

Only one active session per set of API credentials is allowed. If a second session authenticates with the same API credentials as an already existing session, the new session will take over the existing session and the initial session will receive a Logout message and will then be disconnected.

Example Logout message due to a second session authenticating with a set of API credentials already in use by another session:


```
{"correlation":"test123","type":"Logout","text":"Another session has connected with this apiKey. Closing session.","encodedTextLen":0,"encodedText":null}
```


After successfully creating the connection, the following message should be sent to authenticate it:

Field	Req	Value
correlation	Y	The provided correlation string will be returned on the response. Use this to map requests to responses. The response type will be different from the submitted request type. Only alphanumeric (a..z,A..Z,0..9) values are allowed with a max of 50.
type	Y	AuthenticationRequest
token	C	Jwt token generated using the method described above

Example request:


```
{"correlation":"test123","type":"AuthenticationRequest","token":"jwt-generated-token"}
```


Example response:


```
{"correlation":"md","type":"AuthenticationResult","success":true,"message":"Authentication successful"}
```


2.4 Unsolicited Messages

The WebAPI server will send an unsolicited message in only one scenario: when the exchange is marked as offline.


```
{"correlation":"unsolicited","type":"OFFLINE","message":"The exchange is now offline"}
```


It is recommended that Clearing Members disconnect and retry again later.

2.5 Connection Time-out

The websocket session will be disconnected after 66 minutes of idle connection. In order to determine if the websocket server is up or to keep idle websocket connections alive the standard websocket "Ping/Pong" control messages may be used as a heartbeat mechanism.

ErisX

2.6 Rate Limiting

Once the connection is established, it will be subject to a messaging rate limit. The limit is based on token usage. The maximum number of tokens that can be used per second is 40. Every second the number of available tokens refills by an amount of 10 tokens. Different request types have different token usage, see table below for more information.

If the limit is exceeded the user will get a response back informing them that the limit has been exceeded and the request has been ignored. Requests will be accepted again after the user has enough available tokens to make the appropriate request.

Request Type	Tokens	Request Type	Tokens
AuthenticationRequest	1	NewLimitOrderSingle	1
SecurityList	20	NewStopLimitOrderSingle	1
MarketDataSubscribe	1	CancelLimitOrderSingleRequest	1
MarketDataStatus	1	CancelStopLimitOrderSingleRequest	1
MarketDataUnsubscribe	1	ReplaceLimitOrderSingleRequest	1
TopOfBookMarketDataSubscribe	1	ReplaceStopLimitOrderSingleRequest	1
TopOfBookMarketDataUnsubscribe	1	OrderMassStatusRequest	20
NewLimitOrderSingle	1	PartyListRequest	20

Example response:


```
{"correlation":"15675211888790","type":"ERROR_MESSAGE","error":"Your request used 10 tokens, which exceeded the remaining amount of your allocated tokens per second, and was ignored. Please try again later.","details":"correlation=15675211888790"}
```


2.7 Table's Legend

Req	Explanation
Y	Field is always required.
N	Field is not required.
O	Field is optional.
C	Field is conditional upon the message type and/or other field values.
F	Field is required only for Futures.

ErisX

3 Real-time Market Data Service

This section describes a set of messages that allow a clearing member to subscribe to real-time market data.

3.1 Subscription Requests

Each subscription request must contain a correlation value, subscription type and symbol.

Field	Req	Value
correlation	Y	The provided correlation string will be returned on the response. Use this to map requests to responses. The response type will be different from the submitted request type. Only alphanumeric (a..z,A..Z,0..9) values are allowed with a max of 50.
type	Y	The data subscription type
symbol	C	Product code i.e. BTC/USD

Example:

```
"correlation": "123456789abcdefg", "type": "MarketDataSubscribe", "symbol": "BTC/USD"
```

3.2 Request Status

A subscription request will be responded to with a status message indicating whether or not the request was successful.

Example:

```
{
  "correlation": "abc123",
  "type": "STATUS",
  "message": "Subscribed to market data for BTC/USD."
}
```

3.3 Security Status

Following the response to a successful Market Data Subscription Request a message with **“type”: “SecurityStatus”** will also be sent to the client application. This message describes the current trading status of the given symbol.

A Security Status message will also be sent whenever there is a change to the securityTradingStatus for a given symbol.

Values for securityTradingStatus:

Values
NOT_AVAILABLE_FOR_TRADING_END_OF_SESSION
READY_TO_TRADE_START_OF_SESSION
TRADING_HALT
PRE_OPEN

Example:

```
{
  "correlation": "15849795388040",
  "type": "SecurityStatus",
  "security": {
    "currency": "BTC",
    "symbol": "BTCZ9",
    "symbolSfx": null,
    "securityDesc": "BTCZ9",
    "minTradeVol": 1,
    "maxTradeVol": 100000,
    "roundLot": 1,
    "minPriceIncrement": 1,
    "product": "COMMODITY",
    "cfiCode": "FCXXSX",
    "securityType": "FUT",
    "maturityMonthYear": "202007",
    "contractMultiplier": 0.1,
    "securityExchange": "HYDX",
    "activation": "20200323",
    "lastEligableTradeDate": "20200701",
    "maturityDate": "20200701",
    "lastTradeTime": "15:00:00Z",
    "expiryTime": "15:00:00Z"
  },
  "currency": "BTC",
  "securityTradingStatus": "NOT_AVAILABLE_FOR_TRADING_END_OF_SESSION",
  "sessionEnd": "CHANGE_OF_TRADING_SESSION",
  "sendingTime": "20200323-16:08:56.259",
  "transactTime": "20200323-16:45:10.036645101"
}
```


3.4 Market Status Messages

A JSON message should be submitted over the websocket client with **“type”: “MarketStatus”** in order to get a response with information on the Market Status.

Request:

```
 {
     "correlation": "abc123",
     "type": "MarketStatus",
 }
```

Response:


```
{
   "correlation":"abc123",
   "type":"STATUS",
   "message":"Exchange is open"
}
```

ErisX

3.5 Security List Messages

A JSON message should be submitted over the websocket client with **"type":"SecurityList"** in order to get all available symbols. The symbol list is updated periodically.

The Security List request message can include an optional field "**securityGroup**" to better filter the list of available symbols that will be sent in the Security List.

Field	Req	Value
securityGroup	N	**ALL**: ErisX will return all active instruments Other value, ErisX will return all active instruments where securityGroup matches the requested value If securityGroup is not specified, ErisX will only return a default subset of contracts

Security List Response

Field	Req	Value
correlation	Y	Value provided by the clearing member request for the subscription
symbol	Y	Instrument (E.g. BTC/USD)
symbolSfx	O	Instrument suffix
product	O	Product Type
cfiCode	O	FCXXSX for futures
securityType	O	FUT = Futures
contractMultiplier	O	The quantity of underlying units per 1 futures contract
maturityMonthYear	O	Specifies the month and year of maturity (YYYYMM)
maturityDate	O	Specifies date of maturity (YYYYMMDD)
lastEligibleTradeDate	O	Specifies last available trade date
activation	O	Specifies date when the contract becomes active
securityExchange	O	Market used to help identify a security = ERSX
minPriceIncrement	Y	Minimum price change for a given symbol
securityDesc	Y	Security description
minTradeVol	Y	The minimum order quantity that can be submitted for an order
maxTradeVol	Y	The maximum order quantity that can be submitted for an order
roundLot	Y	Trading lot size of security (minimum fill size)
currency	Y	This will be the Base currency
securityGroup	O	An exchange specific name assigned to a group of related securities which may be concurrently affected by market events and actions
productCode	O	Groups asset based on a common contract specification
cap	O	Upper Price Boundary of a contract
floor	O	Lower Price Boundary of a contract

ErisX

Request:

```
{
    "correlation": "abc123",
    "type": "SecurityList",
    "securityGroup": "ALL"
}
```

Response:


```
{
    "correlation": "12345abc",
    "securities": [{
        "currency": "BTC",
        "symbol": "BTC/USD",
        "symbolSfx": "SP",
        "securityDesc": "Bitcoin USD",
        "minTradeVol": 0.01,
        "maxTradeVol": 1000.0,
        "roundLot": 0.001,
        "minPriceIncrement": "0.01"
    },
       {
         "activation": "20191115",
         "cfiCode": "FCXXSX",
         "contractMultiplier": 1,
         "currency": "BTC",
         "lastEligableTradeDate": "20191125",
         "maturityDate": "20200223",
         "maturityMonthYear": "202002",
         "maxTradeVol": 100000,
         "minPriceIncrement": 1,
         "minTradeVol": 1,
         "product": "COMMODITY",
         "roundLot": 0.01,
         "securityDesc": "BTCZ9",
         "securityExchange": "ERSX",
         "securityType": "FUT",
         "symbol": "BTCZ9",
         "symbolSfx": null
    }, ...]
}
```


3.6 Market Data Subscriptions Types

There are a number of different market data subscription types paired with unsubscribe types.

Type	Description
MarketDataSubscribe	This is a subscription to the full order book. Upon a successful request a snapshot of the entire order book is provided followed by incremental data and trade updates for

	as long as the subscription is active.
MarketDataSubscribe with tradeOnly flag	A similar subscription using the 'tradeOnly' flag will provide a stream of updates for only trades within the given symbol.
MarketDataUnsubscribe	This request is used to unsubscribe from a full order book subscription or the 'tradeOnly' equivalent for a given symbol.
TopOfBookMarketDataSubscribe	This subscription allows the user to request an aggregated order book with up to 20 levels of depth using the topOfBookDepth field. Upon a successful request a snapshot of the requested levels is provided followed by incremental data updates for as long as the subscription is active.
TopOfBookMarketDataUnsubscribe	This request is used to unsubscribe from a Top Of Book subscription for a given symbol.

3.7 Response Types

The above subscriptions will be responded to with different response types.

The snapshot messages received after initial subscription requests will not have a response type. This message provides a complete set of order book data after a successful subscription is made.

***Note**: Users are advised to clear out any previous known orderbook information for the given symbol prior to processing a snapshot message.*

Type	Description
MarketDataIncrementalRefresh	A message containing a list of bids and or offer changes. Each bid and offer will contain an updateAction to indicate the type of change it represents
MarketDataIncrementalRefreshTrade	This message will contain one or many trade reports for matched orders.
TopOfBookMarketData	Updates in this message are aggregated by price and indicate the number of orders and total volume available at that price. ***Note**: Users are advised to clear out any previous known orderbook information for the given symbol.*

ErisX

3.8 Response Fields

Within each response message there are a set of fields providing details of the update.

***Note**: Not all fields are received for each message.*

Field	Req	Value
correlation	Y	Value provided by the clearing member request for the subscription
symbol	C	Product code
sendingTime	C	The time the message was sent from the match engine
Bids[ ]	C	A list of buy orders in the current orderbook
Offers[ ]	C	A list of sell orders in the current orderbook
Trades[ ]	C	A list of trade reports
id	C	The id is unique per symbol within a single session. See section below ' Handling 'id' full order book updates'.
updateAction	C	The Market Data update action type. New or Delete
price	C	The price of a corresponding bid, offer or trade.
amount	C	The order quantity for a resting bid or offer
currency	C	The currency of the order value
tickerType	C	PAID: A buy order that aggresses or 'lifts' the offer price. GIVEN: A sell order that aggresses or 'hits' a bid price.
transactTime	C	The time the execution happened on the exchange
size	C	The quantity executed on the trade
count	C	The number of orders represented in the TopOfBook update at a given price level
totalVolume	C	The total order volume for a given price in a TopOfBook update
endFlag	C	**EndOfTrade**. Indicates when no more trades for an event will be published. **EndofEvent**. Will be sent on the final message of a sequence to indicate that all prior messages were part of an atomic matching event.
numberOfOrders	C	In MarketDataIncrementalRefreshTrade indicates number of orders involved in the matching event.

3.9 Handling 'id' for full order book updates

When using the full order book subscription "MarketDataSubscribe", the snapshot and continuous market data messages contain an id that identifies the price to remove or replace in a full book scenario.

The id is unique per instrument within a single session represented as a hexadecimal encoding of a long data type as a string.

Within the same symbol, only one (1) price can be outstanding for any id, and subsequent updates having the same id as an outstanding price replace it or delete it from the book. The action is specified in updateAction New or Delete.

The client session is responsible for monitoring the MDEntryID (278) tag to keep track of these updates.

ErisX

3.10 Example Messages

3.10.1 MarketDataSubscribe

A JSON message should be submitted over the websocket client with **"type": "MarketDataSubscribe"** in order to establish a full order book market data subscription.

Request:

```
{
    "correlation": "15753832469890",
    "type": "MarketDataSubscribe",
    "symbol": "BTC/USD"
}
```

Response - Snapshot:


```
{
  "correlation": "15753832469890",
  "type": "MarketDataIncrementalRefresh",
  "symbol": "BTC/USD",
  "sendingTime": "20191203-14:27:23.563",
  "bids": [
    {
      "id": "1000000003765",
      "updateAction": "NEW",
      "price": 7295,
      "amount": 0.01,
      "symbol": "BTC/USD"
    },
    {
      "id": "100000000375d",
      "updateAction": "NEW",
      "price": 7294,
      "amount": 0.01,
      "symbol": "BTC/USD"
    }
  ],
  "offers": [
    {
      "id": "1000000003660",
      "updateAction": "NEW",
      "price": 7317,
      "amount": 0.01,
      "symbol": "BTC/USD"
    },
    {
      "id": "1000000003665",
      "updateAction": "NEW",
      "price": 7318,
      "amount": 0.01,
      "symbol": "BTC/USD"
    },
```

```
    {
      "id": "1000000003666",
      "updateAction": "NEW",
      "price": 7319,
      "amount": 0.01,
      "symbol": "BTC/USD"
    }
  ],
  "transactTime": "20191203-14:27:23.553029224"
}
```


Response - Incremental updates


```
{
  "correlation": "15753904509040",
  "type": "MarketDataIncrementalRefresh",
  "symbol": "BTC/USD",
  "sendingTime": "20191203-16:28:14.585",
  "bids": [],
  "offers": [
    {
      "id": "10000000037a1",
      "updateAction": "DELETE",
      "price": 7291,
      "amount": 0.01,
      "symbol": "BTC/USD"
    }
  ],
  "transactTime": "20191203-16:28:14.546414345"
}
```


Response - Trade updates:


```
{
  "correlation": "15753904509040",
  "type": "MarketDataIncrementalRefreshTrade",
  "symbol": "BTC/USD",
  "sendingTime": "20191203-16:28:14.583",
  "trades": [
    {
      "updateAction": "NEW",
      "price": 7291.0,
      "currency": "BTC",
      "tickerType": "PAID",
      "transactTime": "20191203-16:28:14.546414345",
      "size": 0.01,
      "symbol": "BTC/USD"
    }
  ]
}
```

ErisX

3.10.2 MarketDataSubscribe - Trades Only

A JSON message should be submitted with **“type”: “MarketDataSubscribe”** and **“tradeOnly”: “True:** in order to subscribe to just trade updates. The first response will include the information from the last trade that took place prior to establishing the subscription.

Request:

```
{
    "correlation": "15753904509040",
    "type": "MarketDataSubscribe",
    "tradeOnly": True,
    "symbol": "BTC/USD"
}
```

Response - Trade updates:

```
{
  "correlation": "15753904509040",
  "type": "MarketDataIncrementalRefreshTrade",
  "symbol": "BTC/USD",
  "sendingTime": "20191203-16:28:14.583",
  "trades": [
    {
      "updateAction": "NEW",
      "price": 7291.0,
      "currency": "BTC",
      "tickerType": "PAID",
      "transactTime": "20191203-16:28:14.546414345",
      "size": 0.01,
      "symbol": "BTC/USD"
    }
  ]
}
```

3.10.3 Market Data Unsubscribe

A JSON message should be submitted over the websocket client with **“type”: “MarketDataUnsubscribe”** in order to cancel an existing subscription.

Request:

```
{
  "correlation": "abc456",
  "type": "MarketDataUnsubscribe",
  "symbol": "BTC/USD"
}
```

Response:

```
{
  "correlation": "abc456",
  "type": "INFO_MESSAGE",
  "message": "Unsubscribed from market data for BTC/USD."
}
```

ErisX

3.10.4 TopOfBookMarketDataSubscribe

A JSON message should be submitted over the websocket client with **"type": "TopOfBookMarketDataSubscribe"** in order to establish a simple market data subscription.

"topOfBookDepth" is a mandatory field, the user should specify the desired depth on the request, if it's not specified it will default to 0 and although the request will be successful no data will be streamed.

Request:

```
{
    "correlation": "abc123",
    "type": "TopOfBookMarketDataSubscribe",
    "symbol": "BTC/USD",
    "topOfBookDepth": 3
}
```

Response:

```
{
    "correlation": "abc123",
    "type": "TopOfBookMarketData",
    "symbol": "BTC/USD",
    "bids": [{
        "count": 1,
        "action": "NEW",
        "price": 95.0,
        "totalVolume": 200.0
    }, ...
    {
        "count": 4,
        "action": "UPDATE",
        "price": 94.0,
        "totalVolume": 5.0
    }],
    "offers": [{
        "count": 3,
        "action": "NEW",
        "price": 96.0,
        "totalVolume": 5.0
    }}
```

3.10.5 TopOfBookMarketDataUnsubscribe

A JSON message should be submitted over the websocket client with **"type": "TopOfBookMarketDataUnsubscribe"** in order to cancel an existing subscription.

Request:

```
{
  "correlation": "abc456",
  "type": "TopOfBookMarketDataUnsubscribe",
  "symbol": "BTC/USD"}
```

Response:

```
{
  "correlation": "abc456",
  "type": "INFO_MESSAGE",
  "message": "Unsubscribed from top of book market data for BTC/USD."
}
```

ErisX

4 Order Entry Service

This section describes a set of messages that allow a clearing member to enter and manage orders though the websocket connection.

4.1 Correlation

Each subscription request must contain a correlation value.

Field	Req	Value
correlation	Y	The provided correlation string will be returned on the response. Use this to map requests to responses. Only alphanumeric (a..z,A..Z,0..9) values are allowed with a max of 50.

4.2 PartyID

A partyID is required for all order related messages. An ErisX member may have multiple partyIDs depending on their account setup.

Users should submit a PartyListRequest with the following fields to receive a list of partIDs that they are enabled for.

Field	Value
correlation	Alphanumeric string
type	PartyListRequest

4.3 ClOrderID

When submitting orders via ErisX Websocket API clearing members must include a PartyID in front of their own specified id value separated by a hyphen. The format of the Client Order ID (**ClOrderID)** must follow the convention below and be unique for the trading session.

ClOrdID = PartyID-[user specified value]

The length of the user specified value should not be more than 40 characters.

4.4 Time in Force

The supported time in force values are described in the following table.

Expiry Condition	Description
Day	Orders submitted with this expiry condition that have not been executed will be expired by the system at the end of the ErisX trading day in which they were entered.
Good Till Cancel (GTC)	Orders with this expiry condition remain open and active until either executed or explicitly canceled by the client.
Good Till Date (GTD)	With this time in force, the submitting client specifies the date at which an order is to be expired if not already executed.
Fill or Kill (FOK)	Unless the full quantity of the order can be executed immediately at the specified price or better, an order with this expiry condition will be canceled.

ErisX

Immediate or Cancel (IOC)	Orders with the expiry condition will be canceled unless a specified minimum quantity can be executed immediately at the specified price or better. Any remaining unfilled quantity is canceled.

4.5 Minimum Permitted Order Entry Size

There is a minimum permitted order entry size maintained on ErisX platform. Orders sent for amounts less than the permitted minimum order entry size will be rejected.

4.6 Timestamping / TransactTime

Messages sent by client applications will need to include TransactTime. The system will validate the value sent down to one second precision and accuracy.

Responses from the match engine will include TransactTime and will be sent with nanosecond precision. YYYYMMDD-HH:MM:ss.SSSSSSSSS.

The timestamp on outgoing messages will represent the time the corresponding message was received by the FIX gateway that resulted in the update.

4.7 Post-Only Order

Post-Only is a new order type that provides a Trader with a way to enter a passive order and guarantee it won't match an order across the bid-ask spread. This order enables Traders to ensure that their orders will always act as a maker and not a taker of liquidity. If the order is accepted it acts just like a standard limit order.

Post-Only flag can be used in messages types **NewLimitOrderSingle**, **NewStopLimitOrderSingle**, **ReplaceLimitOrderSingleRequest** or **ReplaceStopLimitOrderSingleRequest**.

In case a Post-Only type order attempts to cross the market during continuous trading, two execution reports are sent, one for New Order Single and a Cancel due to **invalid ALO** (add liquidity only).

If we consider the following order book (broken down to individual orders to show granularity).

BID Q	BID	ASK	ASK Q
10	9002	9010	50
10	9002		
5	9002		
5	9001		
5	9001		
15	9000		

Example 1: Cancelled

- A Trader places a new post-only order to sell at a price of 9002.
- This order would be immediately canceled as the current best Bid is 9002 and this order would have aggressed the bid and removed liquidity.

ErisX

Example 2: Accept

- A Trader places a new post only order to sell at a price of 9005.
- The order would be accepted as a limit order as it would add liquidity and now becomes the new best offer.

4.8 Futures Specific Functionality

4.8.1 Regulatory Tags

ErisX requires members to populate some specific fields when sending futures orders to the exchange.

Field Name	Value
senderSubId	Value used to identify the user that entered the order
senderLocationId	Used to identify the geographical location of the user that entered the order: [Country],[State if in US] eg; US, IL or UK

The following tags MUST be included when entering or modifying an order:

Field Name	Value
accountType	Used to indicate whether an order is for a Customer(1) or House(2) account.
custOrderCapacity	Used to indicate whether the user entering the order is placing it for themselves or for another member. 1 = Member Trading for own account 2 = Clearing firm trading for its Prop Account 3 = Member trading for another member 4 = All other

4.8.2 Trades which are cleared through a Futures Commission Merchant (FCM)

For users whose trades are cleared through an FCM, an identifier is required to be sent so that those trades are correctly processed by the FCM back office.

Field Name	Value
customerAccountRef	This must contain the FCM Back office account number for the customer of the order.

4.9 New Order Fields

ErisX supports the following order types.

- **Limit** - An order to buy or sell at a specific price or better.
- **Stop-Limit** - An order that combines the features of a stop order and a limit order. The stop price acts as a trigger to enter a limit order into the market.

Field	Req	Value
correlation	Y	Alphanumeric string
type	Y	**NewLimitOrderSingle** or **NewStopLimitOrderSingle**

ErisX

clOrdID	Y	Must start with **partyID-**. The partyId should be a real party ID. Maximum length = partyID-[+40 characters]
currency	Y	The currency for the amount specified in the OrderQty.
side	Y	**BUY or SELL**
symbol	Y	Unique instrument identifier
partyID	N	Party ID of the account
transactionTime	Y	See TransactTime description below. Time at which the order was submitted.
orderQty	Y	Order Quantity
ordType	Y	**LIMIT** or **STOP_LIMIT**
price	Y	Order price
stopPrice	C	Stop price of the Stop Limit Order. For a buy order, the stop price must be set at least one tick below the limit price. For a sell order the stop price must be set at least one tick above the limit price.
timeInForce	O	**Day**, **GoodTillCancel**, **GoodTillDate, FillOrKill, ImmediateOrCancel**
expireDate	C	Only available for GoodTillCancel order. UTC format **YYYYMMDD**
accountType	F	**1**=Customer, **2**=House
custOrderCapacity	F	CTICode (customer type indicator) **1** = Member Trading for own account **2** = Clearing firm trading for its Prop Account **3** = Member trading for another member **4** = All other
senderLocationId	F	Used to identify the geographical location of the user that entered the order: [Country],[State if in US] eg; US, IL or UK
customerAccountRef	O	Customer Account Reference (FCM Back office Account)
senderSubId	F	Value used to identify the user that entered the order
postOnly	O	Indicate if the newOrderSingle is Post-Only or not. **N** = No Post-Only type (default). **Y** = Post-Only type.

4.10 Order Modification

Order parameters such as quantity and expiry condition can be amended on an outstanding order without having to cancel and resubmit the order.

By default, orders that have been partially filled cannot be modified unless the user makes use of the overfill protection logic. See section Overfill protection (New). A reject message will be received if attempting to modify a partially filled order without the use of overfill protection.

When modifying an existing order the associated IDs (origClOrdID and OrderID) are required.

Field	Req	Value
correlation	Y	Alphanumeric string
type	Y	**ReplaceLimitOrderSingleRequest** or **ReplaceStopLimitOrderSingleRequest**
clOrdID	Y	Must start with **partyID-**. The partyID should be a real party ID corresponding to the API credentials.

ErisX

		Maximum length = partyID-[+40 characters]
origClOrdID	Y	Must be the client ID of the original submitted order.
orderID	Y	Must be ErisX assigned ID of the original order.
currency	Y	Quote Currency of the product
side	Y	BUY or SELL
symbol	Y	Unique instrument identifier
timeInForce	O	**Day**, **GoodTillCancel**, **GoodTillDate, FillOrKill, ImmediateOrCancel**
expireDate	C	Only available for GoodTillCancel order. UTC format **YYYYMMDD**
partyID	Y	Must match the partyID- on the clOrdID
transactionTime	O	See TransactTime description below. Time at which the order was submitted.
orderQty	O	Order Quantity
Price	O	Order price
stopPrice	C	Stop price of the Stop Limit Order. For a buy order, the stop price must be set at least one tick below the limit price. For a sell order the stop price must be set at least one tick above the limit price.
overfillProtection	O	Required when trying to modify a partially filled order to specifically request "Overfill Protection" otherwise the modification is rejected. Y = LeavesQty is set to requested quantity - CumQty N = LeavesQty is set to the quantity requested in the cancel replace message
accountType	F	1=Customer, 2=House
custOrderCapacity	F	CTICode (customer type indicator) 1 = Member Trading for own account 2 = Clearing firm trading for its Prop Account 3 = Member trading for another member 4 = All other
senderLocationId	F	Used to identify the geographical location of the user that entered the order: [Country],[State if in US] eg; US, IL or UK
customerAccountRef	O	Customer Account Reference (FCM Back office Account)
senderSubId	F	Value used to identify the user that entered the order
postOnly	O	Indicate if the newOrderSingle is Post-Only or not. **N** = No Post-Only type (default). **Y** = Post-Only type.

4.11 Order Cancellation

When cancelling an existing order the associated IDs (origClOrdID and OrderID) are required.

Field	Req	Value
correlation	Y	Alphanumeric string
type	Y	**CancelLimitOrderSingleRequest** or **CancelStopLimitOrderSingleRequest**
clOrdID	Y	Must start with **partyID-**. The partyID should be a real party ID corresponding to the API credentials. Maximum length = partyID-[+40 characters]
origClOrdID	Y	Must be the client ID of the original submitted order.

ErisX

orderID	Y	Must be ErisX assigned ID of the original order.
currency	Y	Quote Currency of the product
side	Y	BUY or SELL
symbol	Y	Unique instrument identifier
partyID	Y	Must match the partyID- on the clOrdID
transactionTime	O	See TransactTime description below. Time at which the order was submitted.

4.12 Cancel All Orders

Client applications are able to cancel all working orders for a partyID via a single **CancelAllOrdersRequest**.

Field	Req	Value
correlation	Y	Alphanumeric string
type	Y	**CancelAllOrdersRequest**
partyID	Y	Must match the partyID- on the clOrdID

4.13 Overfill protection (New)

If an order has been partially filled, then our custom tag OverfillProtection (5000=Y or N) must be included on the 35=G Order Replace message.

- With Overfill Protection = Y, the original quantity is modified which will update the remaining quantity (LeavesQty) to the new requested qty minus the already filled cumulative quantity.
- Whereas with Overfill Protection = N, the remaining quantity (LeavesQty) is set to the new quantity as specified in the modified message.
- If the Overfill Protection tag 5000 is not set and the order which is requesting modification has been partially filled, then the request will be rejected.

Example:

Given: An original order to buy 5 lots which has been partially filled.

```
Order Quantity = 5, Filled = 3, LeavesQty = 2, Cancelled = 0
```

When: A modify request is received containing an OrderQty of 4 with Overfill Protection = **Y**

Then: The order quantity is set to 4, which reduces the remaining quantity (LeavesQty) quantity down to 1.

```
Order Quantity = 4, Filled = 3, LeavesQty = 1, Cancelled = 0
```

Or:

When: A modify request is received containing an OrderQty of 4 with Overfill Protection = N

Then: The remaining quantity (LeavesQty) is set to 4

ErisX

```
Order Quantity = 7, Filled = 3, LeavesQty = 4, Cancelled = 0
```

4.14 Execution Report Fields

ErisX sends Execution Report messages to:

- Confirm the receipt of an order
- Confirm changes to an existing order
- Reply to order status messages
- Relay order fill information on active orders
- Reject an order

In a normal workflow, after sending an Execution Report message to indicate the receipt of the order, ErisX will continue to send one or more Execution Report messages to relay order fill information. If the order is filled in full, it will be indicated in the Execution Report. In cases of partial fills, ErisX will send Execution Report messages indicating partial fills until the order is completely filled, the client actively cancels the remaining portion of the order, or the remaining portion expires.

For a multiple filled order, the ExecType (150) field reports information on the individual fill and the OrdStatus (39) field reports information on the overall order status.

Execution Reports of fills or partial fills of active orders are sent to all active sessions that have authentication for the relevant party ID. That is, if there are two live sessions that have authenticated using two different set of API credentials and both of them have authentication for a party ID, both sessions will receive Execution Reports for that party ID.

Field	Req	Value
correlation	Y	Alphanumeric string submitted by users
type	Y	ExecutionReport
OrderID	Y	Unique order identifier assigned by ErisX.
ClOrdID	Y	Client assigned order id
OrigClOrdID	Y	Original client assigned order id submitted on the order.
ExecID	Y	Completed trade identifying number.
ExecType	Y	The execution report's type. New, Canceled, Replace, Rejected, Expired, Fill Status,Order Status
OrdStatus	Y	The current state of chain of orders. New, Partial filled, Filled, Canceled, Replaced, Rejected, Expired
OrdRejReason	O	Optional when Rejected.
Account	O	The clearing account name as agreed to by ErisX and the client or else defaulted by the system.
Symbol	Y	The order currency pair.
SymbolSfx	O	SP = spot (default if not specified)
Side	Y	Order side: Buy or Sell
OrderQty	Y	Order quantity. Not sent if Canceled or Rejected.

ErisX

OrdType	O	Supported values are: Limit order or Stop-Limit order
Price	Y	Required for Limit orders
StopPx	C	The price at which a stop order becomes effective.
StopSide	C	Stop Side field for StopLimit. Bid or Offer
Currency	O	The currency for the amount specified in OrderQty field.
ExpireTime	Y	Time/Date of order expiration (always expressed in UTC)
LastQty	O	Quantity bought/sold for this fill. Present when ExecType = F.
LastPx	O	Price at which the current or last fill was made.
LastSpotRate	O	Price for the last fill. Not sent for status requests
LeavesQty	Y	Amount of order open for further execution.
CumQty	Y	Total amount of an order currently executed in a chain of partial fills.
AvgPx	O	The average price at which the order was filled or partially filled.
TradeDate	O	Trades completed after 4 pm CT show the next business day.
TransactTime	O	Execution Reports will be sent with nanosecond precision - YYYYMMDD-HH:MM:SS.sssssssss
Commission	O	Actual Commission (Only for Fills and Partial Fills)
CommCalculated	Y	Calculated Commission
CommType	Y	3 = Absolute (Total monetary amount)
CommCurrency	Y	Currency Commission (USD, BTC)
Text	O	Descriptive text message.
MatchingType	O	Indicates whether or not the maker's price was resting in the book at the time of the match.
TimeInForce	Y	How long an order remains in effect: Good Till Cancel or Good Till Date
ExpireDate	Y	Expiry date in YYYYMMDD format.
partyIDs	O	Party ID of the account
accountType	F	1=Customer, 2=House
custOrderCapacity	F	CTICode (customer type indicator) 1 = Member Trading for own account 2 = Clearing firm trading for its Prop Account 3 = Member trading for another member 4 = All other
senderLocationId	F	Used to identify the geographical location of the user that entered the order: [Country],[State if in US] eg; US, IL or UK
customerAccountRef	O	Customer Account Reference (FCM Back office Account)
senderSubId	F	Value used to identify the user that entered the order
postOnly	O	Indicate if the newOrderSingle is Post-Only or not. **N** = No Post-Only type. **Y** = Post-Only type.
availableBalanceData	O	Provide the member with information regarding their collateral available for trading
availableBalance	O	Collateral available for trading
availableBalanceCurrency	O	Asset type of the collateral available for trading

ErisX

4.15 Order History (Mass Order Status Request)

AVAILABLE BALANCES SENT ON THE MASS ORDER STATUS RESPONSE REFLECT BALANCES AS OF THE TIMESTAMP AVAILABLE IN THE TRANSACTTIME FIELD. THESE BALANCES MAY NOT REFLECT THE MOST UP-TO-DATE BALANCES IF BALANCES HAVE CHANGED DUE TO NON CENTRAL LIMIT ORDER BOOK TRADING ACTIVITY AFTER THE TRANSACTION TIME TIMESTAMP.
EXAMPLES OF ACTIVITY THAT MAY CAUSE NOT UP-TO-DATE BALANCES INCLUDE BUT ARE NOT LIMITED TO:

- **DEPOSITS**
- **WITHDRAWALS**
- **DELIVERIES**
- **BLOCK TRADES**

Order history can be obtained by sending a MassOrderStatusRequest to which ErisX will respond with a set of Execution Report messages containing the current active orders.

A set can comprise Execution Reports for multiple orders and will only include active orders.

The last Execution Report in a set is indicated by LastRptRequested set to “Y”. LastRptRequested is set to “N” if it’s not the end of the messages. In addition TotNumReports can be used to track the total number of execution reports in response to the order mass status request.

If no Execution Report found an empty Execution Report will be sent with order related fields being 0 or “NA”.

4.16 Example Messages

4.16.1 PartyListRequest

A JSON message should be submitted over the websocket client with **“type”:”PartyListRequest”** in order to get all available party IDs for the logged in user.

Example:

```
{
   "type": "PartyListRequest",
   "correlation": "12345abc"}
```

```
{
 "correlation": "12345abc",
 "type": "PartyListResponse",
 "partyIds": [
  "PartyId1",
```

ErisX

```
  "PartyId2",
  "PartyId3"]}
```

4.16.2 NewLimitOrderSingle or NewStopLimitOrderSingle

A JSON message should be submitted over the websocket client with **“type”:”NewLimitOrderSingle”** or **“type”:”NewStopLimitOrderSingle”** in order to place a new order.

Request Spot:

```
{
   "correlation": "foo",
     "type": "NewLimitOrderSingle",
     "clOrdID": "partyID-1",
     "currency": "BTC",
     "side": "BUY",
     "symbol": "BTC/USD",
     "partyID": "partyID",
     "transactionTime": "20181206-20:07:29.196",
     "orderQty": 1.0,
     "ordType": "LIMIT",
     "price": 3720.5,
     "timeInForce":"GoodTillDate",
     "expireDate":"20190308",

}
```

Or:

```
{
     "correlation": "foo",
     "type": "NewStopLimitOrderSingle",
     "clOrdID": "partyID-2",
     "currency": "BTC",
     "side": "BUY",
     "symbol": "BTC/USD",
     "partyID": "partyID",
     "transactionTime": "20181206-20:07:29.196",
     "orderQty": 1.0,
     "ordType": "LIMIT",
     "price": 3720.5,
     "stopPrice": 3720.4,
     "stopSide": "BID"
}
```

Response Spot:

```
{
  "correlation": "foo",
  "type": "ExecutionReport",
  "orderID": "281474976725627",
  "clOrdID": "PartyID-15753985508820",
  "origClOrdID": "PartyID-15753985508820",
  "execID": "281474976823692",
  "execType": "NEW",
  "ordStatus": "NEW",
  "ordRejReason": null,
```

ErisX


```
  "account": "Account",
  "symbol": "BTC/USD",
  "symbolSfx": null,
  "side": "SELL",
  "orderQty": 0.01,
  "ordType": "LIMIT",
  "price": 7503.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 0.01,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191203-18:42:30.915042705",
  "commission": 0.0,
  "commCalculated": 0.1501,
  "commType": "ABSOLUTE",
  "commCurrency": "USD",
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": null,
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191203",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191203-18:42:30.942",
  "targetLocationId": null,
  "custOrderCapacity": 0,
  "accountType": 0,
  "targetSubId": null,
  "customerAccountRef": null
}
```


Request Futures:


```
{
  "correlation": "15754750575091",
  "type": "NewLimitOrderSingle",
  "clOrdID": "PartyID-15754750575091",
  "currency": "BTC",
  "side": "BUY",
  "symbol": "BTCF0",
```

```
  "timeInForce": "Day",
  "transactionTime": "20191204-15:57:37.508",
  "orderQty": "1",
  "ordType": "LIMIT",
  "price": "5000",
  "partyID": "PartyID",
  "accountType": 2,
  "custOrderCapacity": 1,
  "senderLocationId": "US,IL",
  "senderSubId": "Trader1"
}
```


Response Futures:


```
{
  "correlation": "15754750575091",
  "type": "ExecutionReport",
  "orderID": "281474976726608",
  "clOrdID": "PartyID-15754750575091",
  "origClOrdID": "PartyID-15754750575091",
  "execID": "281474976826913",
  "execType": "NEW",
  "ordStatus": "NEW",
  "ordRejReason": null,
  "account": "Account",
  "symbol": "BTCF0",
  "symbolSfx": null,
  "side": "BUY",
  "orderQty": 1.0,
  "ordType": "LIMIT",
  "price": 5000.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 1.0,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191204-15:57:37.486185139",
  "commission": 0.0,
  "commCalculated": 1.0,
  "commType": "ABSOLUTE",
  "commCurrency": "USD",
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": null,
```

ErisX


```
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191204",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191204-15:57:37.515",
  "targetLocationId": "US,IL",
  "custOrderCapacity": 1,
  "accountType": 2,
  "targetSubId": "Trader1",
  "customerAccountRef": null
}
```


4.16.3 ReplaceLimitOrderSingleRequest or ReplaceStopLimitOrderSingleRequest

A JSON message should be submitted over the websocket client with **“type”:”ReplaceLimitOrderSingleRequest”** or **“type”:”ReplaceStopLimitOrderSingleRequest”** in order to replace an existing order.

Request Spot:


```
 {
    "correlation": "0x12309df",
    "type": "ReplaceStopLimitOrderSingleRequest",
    "handlInst": "AutomatedExecutionOrderPrivate",
    "clOrdID": "partyID-4",
    "origClOrdID": "partyID-ORIG_CL_ORD_ID",
    "orderID": "ORD_ID",
    "currency": "BTC",
    "partyID": "FOO",
    "side": "BUY",
    "symbol": "BTC/USD",
    "price": 69,
    "stopPrice": 6993
 }
```


Response Spot:


```
{
  "correlation": "0x12309df",
  "type": "ExecutionReport",
  "orderID": "281474976725627",
  "clOrdID": "PartyID-15753988918391",
  "origClOrdID": "PartyID-15753985508820",
  "execID": "281474976823712",
  "execType": "REPLACE",
  "ordStatus": "REPLACED",
  "ordRejReason": null,
  "account": "Account",
  "symbol": "BTC/USD",
```

```
  "symbolSfx": null,
  "side": "SELL",
  "orderQty": 0.01,
  "ordType": "LIMIT",
  "price": 7504.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 0.01,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191203-18:48:11.881078170",
  "commission": 0.0,
  "commCalculated": 0.1501,
  "commType": "ABSOLUTE",
  "commCurrency": "USD",
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": null,
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191203",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191203-18:48:11.917",
  "targetLocationId": null,
  "custOrderCapacity": 0,
  "accountType": 0,
  "targetSubId": null,
  "customerAccountRef": null
}
```


Request Futures:


```
{
  "correlation": "15754758677961",
  "type": "ReplaceLimitOrderSingleRequest",
  "handlInst": "AutomatedExecutionOrderPrivate",
  "partyID": "PartyID",
  "clOrdID": "PartyID-15754758677961",
  "ordType": "2",
  "origClOrdID": "PartyID-15754758415710",
  "orderID": "281474976726609",
```

```
  "origOrdTif": "Day",
  "currency": "BTC",
  "side": "BUY",
  "price": "5001",
  "orderQty": "2",
  "symbol": "BTCF0",
  "accountType": 2,
  "custOrderCapacity": 1,
  "senderLocationId": "US,IL",
  "senderSubId": "Trader1",
  "overfillProtection": "Y"
}
```


Response Futures:


```
{
  "correlation": "15754758677961",
  "type": "ExecutionReport",
  "orderID": "281474976726609",
  "clOrdID": "PartyID-15754758677961",
  "origClOrdID": "PartyID-15754758415710",
  "execID": "281474976826919",
  "execType": "REPLACE",
  "ordStatus": "REPLACED",
  "ordRejReason": null,
  "account": "Account",
  "symbol": "BTCF0",
  "symbolSfx": null,
  "side": "BUY",
  "orderQty": 2.0,
  "ordType": "LIMIT",
  "price": 5001.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 2.0,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191204-16:11:07.763542459",
  "commission": 0.0,
  "commCalculated": 2.0004,
  "commType": "ABSOLUTE",
  "commCurrency": "USD",
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
```

ErisX


```
  "lastRptRequested": null,
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191204",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191204-16:11:07.793",
  "targetLocationId": "US,IL",
  "custOrderCapacity": 1,
  "accountType": 2,
  "targetSubId": "Trader1",
  "customerAccountRef": null
}
```


4.16.4 CancelLimitOrderSingleRequest or CancelStopLimitOrderSingleRequest

A JSON message should be submitted over the websocket client with **“type”:”CancelLimitOrderSingleRequest”** or **“type”:”CancelStopLimitOrderSingleRequest”** in order to cancel an existing order.

Request:


```
 {
    "correlation": "0x12309df",
    "type": "CancelLimitOrderSingleRequest",
    "partyID": "partyID",
    "clOrdID": "partyID-CL_ORD_ID",
    "origClOrdID": "partyID-ORIG_CL_ORD_ID",
    "orderID": "ORD_ID",
    "currency": "BTC",
    "side": "BUY",
    "symbol": "BTC/US"
 }
```


Response:


```
{
  "correlation": "0x123009df",
  "type": "ExecutionReport",
  "orderID": "281474976725639",
  "clOrdID": "PartyID-15753990353572",
  "origClOrdID": "PartyID-15753990247341",
  "execID": "281474976823721",
  "execType": "CANCELED",
  "ordStatus": "CANCELED",
  "ordRejReason": null,
  "account": "Acciybt",
  "symbol": "BTC/USD",
  "symbolSfx": null,
```

```
  "side": "SELL",
  "orderQty": 0.01,
  "ordType": "LIMIT",
  "price": 7543.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 0.0,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191203-18:50:35.409035868",
  "commission": 0.0,
  "commCalculated": 0.0,
  "commType": "ABSOLUTE",
  "commCurrency": "USD",
  "minQty": 0.0,
  "text": "USER INITIATED",
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": null,
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191203",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191203-18:50:35.438",
  "targetLocationId": null,
  "custOrderCapacity": 0,
  "accountType": 0,
  "targetSubId": null,
  "customerAccountRef": null
}
```


4.16.5 OrderMassStatusRequest

A JSON message should be submitted over the websocket client with **“type”:”OrderMassStatusRequest”** in order to get all known working orders.

Request:

```
{
    "correlation": "foo1",
    "type": "OrderMassStatusRequest",
    "massStatusReqType": "partyID"
```

ErisX

```
  }
```

Response: Execution report:


```
{
  "correlation": "foo1",
  "type": "ExecutionReport",
  "orderID": "281474976725643",
  "clOrdID": "PartyID-15753991771625",
  "origClOrdID": "PartyID-15753991771625",
  "execID": "0",
  "execType": "ORDER_STATUS",
  "ordStatus": "NEW",
  "ordRejReason": null,
  "account": "Account",
  "symbol": "BTC/USD",
  "symbolSfx": null,
  "side": "BUY",
  "orderQty": 0.01,
  "ordType": "LIMIT",
  "price": 7523.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 0.01,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191203-18:52:57.221806384",
  "commission": 0.0,
  "commCalculated": 0.0,
  "commType": null,
  "commCurrency": null,
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": "N",
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191203",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
  ],
  "sendingTime": "20191203-18:53:05.311",
  "targetLocationId": null,
```

```
  "custOrderCapacity": 0,
  "accountType": 0,
  "targetSubId": null,
  "customerAccountRef": null
}
...
...
{
  "correlation": "foo1",
  "type": "ExecutionReport",
  "orderID": "281474976725640",
  "clOrdID": "PartyID-15753991633822",
  "origClOrdID": "PartyID-15753991633822",
  "execID": "0",
  "execType": "ORDER_STATUS",
  "ordStatus": "NEW",
  "ordRejReason": null,
  "account": "Account",
  "symbol": "BTC/USD",
  "symbolSfx": null,
  "side": "SELL",
  "orderQty": 0.01,
  "ordType": "LIMIT",
  "price": 7543.0,
  "stopPrice": 0.0,
  "currency": "BTC",
  "lastPrice": 0.0,
  "lastSpotRate": 0.0,
  "leavesQty": 0.01,
  "cumQty": 0.0,
  "avgPrice": 0.0,
  "tradeDate": null,
  "transactTime": "20191203-18:52:43.446318686",
  "commission": 0.0,
  "commCalculated": 0.0,
  "commType": null,
  "commCurrency": null,
  "minQty": 0.0,
  "text": null,
  "belowMin": null,
  "stopSide": null,
  "matchingType": null,
  "lastRptRequested": "Y",
  "maxShow": 0.0,
  "timeInForce": "Day",
  "expireDate": "20191203",
  "lastQty": 0.0,
  "partyIDs": [
    "PartyID"
```

ErisX


```
  ],
  "sendingTime": "20191203-18:53:05.311",
  "targetLocationId": null,
  "custOrderCapacity": 0,
  "accountType": 0,
  "targetSubId": null,
  "customerAccountRef": null
}
```


Response: System Busy


```
{
  "correlation":"foo1",
  "type":"INFO_MESSAGE",
  "information":"The request has been queued, and will be processed shortly."
}
```


Response: No data


```
{
  "correlation":"foo1",
  "type":"INFO_MESSAGE",
  "information":"No orders to report."
}
```