



ErisX Websocket API V1.0
Market Data and Order Entry



Please contact ErisX sales representatives or help desk personnel for more information on this documentation.

Contents

General	4
WebSocket API Endpoint URL	4
API Credentials	4
API Key permissions	4
Authentication	5
Unsolicited Messages	6
Connection Time-out	7
Rate Limiting	7
Real-time Market Data Service	9
Subscription Requests	9
Request Status	9
Market Status Messages	9
Symbol List Messages	10
Market Data Subscriptions Types	10
Response Types	11
Response Fields	11
Handling 'id' for full order book updates	12
Example Messages	12
MarketDataSubscribe	12
MarketDataSubscribe - Trades Only	14
Market Data Unsubscribe	14
TopOfBookMarketDataSubscribe	15
TopOfBookMarketDataUnsubscribe	15
Order Entry Service	17
Correlation	17
PartyID	17
COrderID	17
Time in Force	17
Minimum Permitted Order Entry Size	18
TransactTime Value	18
TransactTime Precision	18
New Order Fields	18
Order Modification and Cancellation	19
Execution Report Fields	20



Order History (Mass Order Status Request)	21
Example Messages	22
PartyListRequest	22
NewLimitOrderSingle or NewStopLimitOrderSingle	22
ReplaceLimitOrderSingleRequest or ReplaceStopLimitOrderSingleRequest	24
CancelLimitOrderSingleRequest or CancelStopLimitOrderSingleRequest	25
OrderMassStatusRequest	25
Change History	27



1 Change History

Date	Message(s) or Section	Description
20190816		Version 1
20190930	Authentication	A number of changes to better describe the authentication method.

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/>

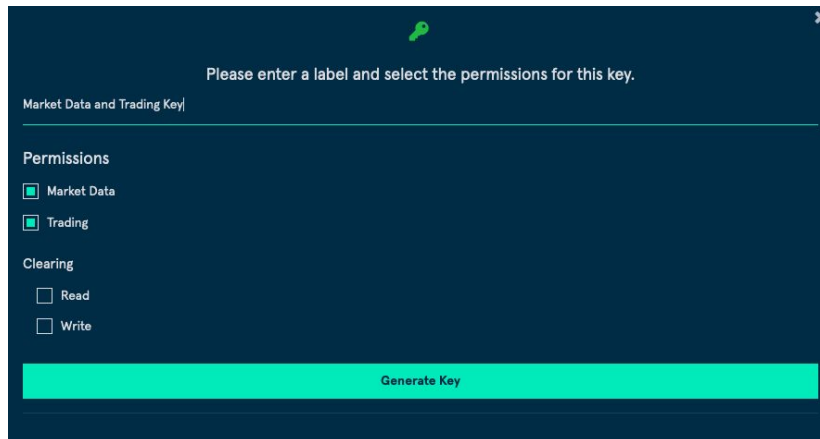
2.2 API Credentials

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.



The screenshot shows a dark-themed form titled "Please enter a label and select the permissions for this key." with a close button in the top right corner. Below the title is a text input field containing "Market Data and Trading Key". Under the heading "Permissions", there are two checked checkboxes: "Market Data" and "Trading". Under the heading "Clearing", there are two unchecked checkboxes: "Read" and "Write". At the bottom of the form is a large red button labeled "Generate Key".

API Key permissions

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

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 2 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')

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

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')

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 existing session will receive a Logout message and after that it will get 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.

2.6 Rate Limiting

Once the connection is established, it will be subject to a messaging rate limit. The limit is based on tokens usage. The maximum number of tokens that can be used per second is 10. Every second the number of available tokens refill 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 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	Token Usage
AuthenticationRequest	1
SecurityList	1
MarketDataSubscribe	1
MarketDataStatus	1
MarketDataUnsubscribe	1
TopOfBookMarketDataSubscribe	1
TopOfBookMarketDataUnsubscribe	1
NewLimitOrderSingle	1
NewStopLimitOrderSingle	1
CancelLimitOrderSingleRequest	1



CancelStopLimitOrderSingleRequest	1
ReplaceLimitOrderSingleRequest	1
ReplaceStopLimitOrderSingleRequest	1
OrderMassStatusRequest	1
PartyListRequest	1

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"}
```



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 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"
}
```



3.4 Symbol 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. New symbols will be sent over the websocket if a Symbol list subscription has been made.

Request :

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

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"
  }, ...]
}
```

3.5 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.
--	----------------------------------

3.6 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.

3.7 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

3.8 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 the same symbol, only one (1) price can be outstanding for any one 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 as either new or delete.

The client session is responsible for monitoring the id value to keep track of these updates.

3.9 Example Messages

3.9.1 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": "abc123",
  "type": "MarketDataSubscribe",
  "symbol": "BTC/USD"
}
```

Response - Snapshot :

```
{
  "correlation": "abc123",
  "symbol": "BTC/USD",
  "bids": [{
    "id": "9",
    "updateAction": "NEW",
    "price": 6994.4,
    "amount": 22.0
  },
  ...
  {
    "id": "215",
    "updateAction": "NEW",
    "price": 6994.47,
    "amount": 5.0
  }],
  "offers": [{
    "id": "132",
    "updateAction": "NEW",
    "price": 6994.5,
    "amount": 5.0
  },
  ...
  {
    "id": "19",
    "updateAction": "NEW",
    "price": 6994.6,
    "amount": 10.0
  }
  ]
}
```

Response - Incremental updates

```
{
  "correlation": "abc123",
  "type": "MarketDataIncrementalRefresh",
  "symbol": "LTC/USD",
  "sendingTime": "20190328-20:31:56.885",
  "bids":
  [
    {
      "id": "114",
      "updateAction": "NEW",
      "price": 0.2,
      "amount": 1.0,
    },
    ...
  ],
  "offers":
  [
    {
      "id": "4",
      "updateAction": "NEW",
      "price": 50.55,
      "amount": 4.0,
    },
    ...
  ]
}
```

Response - Trade updates:

```
{
```



```

"correlation": "abc",
"type": "MarketDataIncrementalRefreshTrade",
"symbol": "LTC/USD",
"sendingTime": "20190328-16:05:48.137",
"trades":
  [
    {
      "updateAction": "NEW",
      "price": 50.5,
      "currency": "LTC",
      "tickerType": "PAID",
      "transactTime": "20190328-16:05:48.130",
      "size": 1.0
    },
    ...
  ]
}

```

3.9.2 MarketDataSubscribe - Trades Only

A JSON message should be submitted with **"type": "MarketDataSubscribe"** and **"tradeOnly": "True"** in order to subscribe to just trade updates.

Request :

```

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

```

Response - Trade updates:

```

{
"correlation": "abc",
"type": "MarketDataIncrementalRefreshTrade",
"symbol": "LTC/USD",
"sendingTime": "20190328-16:05:48.137",
"trades":
  [
    {
      "updateAction": "NEW",
      "price": 50.5,
      "currency": "LTC",
      "tickerType": "PAID",
      "transactTime": "20190328-16:05:48.130",
      "size": 1.0
    },
    ...
  ]
}

```

3.9.3 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": "STATUS",
  "message": "Unsubscribed from market data for BTC/USD."
}
```

3.9.4 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
  }, ...
  {
    "count": 19,
    "action": "UPDATE",
    "price": 97.0,
    "totalVolume": 10.0
  }
  ]
}
```




3.9.5 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": "STATUS",
  "message": "Unsubscribed from top of book market data for BTC/USD."
}
```



4 Order Entry Service

This section describes a set of messages that allow a clearing member to enter and manage orders through 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 partyIDs 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.

ClOrderID = 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 with this expiry type that have not been executed will be expired by the system at the end of the ErisX system day on which they were entered.
Good Till Cancel (GTC)	Orders with this expiry setting remain open and active until either executed or explicitly canceled by the client.
Good Till Date	The submitting client specifies the date at which an order is to be expired if not already executed.



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 TransactTime Value

The following table describes how TransactTime field is used based on message types:

Message Type	Transaction Time
New Order Single	Time at which the order was submitted
Execution Report	Fills: Time at which the trade occurred. Other Events: Time at which the initial order was submitted
Order Cancel Request	Time at which the initial order was submitted
Order Replace Request	Time at which the replace was submitted
Order Cancel Reject	Time at which the reject occurred
List Status	Time at which the report was sent

4.7 TransactTime Precision

The following table describes the precision of (tag 60) TransactTime field:

Message	Seconds	Milliseconds	Microseconds
New	Y		
Partial Fill			Y
Filled			Y
Cancelled	Y		
Replaced	Y		
Rejected		Y	
Expired	Y		

4.8 New Order Fields

ErisX supports the following orders 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
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
transactTime	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.
stopSide	C	Stop Limit Order side
timeInForce	O	Day, GoodTillCancel, GoodTillDate
expireDate	C	Only available for GoodTillCancel order. UTC format yyyyMMdd

4.9 Order Modification and Cancellation

Certain order parameters can be amended on an outstanding order without having to cancel and resubmit the order. The following parameters are eligible for replacement:

- Order Quantity
- Price
- Stop Price

Orders that have been partially filled cannot be modified. A reject message will be received if attempting to modify a partially filled order.

When cancelling or 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. 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
partyID	O	Must match the partyID- on the clOrdID
transactTime	Y	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.

4.10 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.
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 microsec precision - YYYYMMDD-HH:MM:SS.ssssss
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

4.11 Order History (Mass Order Status Request)

Order history can be obtained by sending a MassOrderStatusRequest to which ErisX will respond with a set of Execution Report messages having ExecType set to the appropriate Order Status. There will be one Execution Report for each order status change.



A set can comprise Execution Reports for multiple orders and will include active orders as well as filled, expired and cancelled orders. The set will cover the past 48 hours of order activity for an account.

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".

4.12 Example Messages

4.12.1 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",
  "partyIDs": [
    "partyID1",
    "partyID2"
  ]
}
```

4.12.2 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 :

```
{
  "correlation": "foo",
  "type": "NewLimitOrderSingle",
  "clOrdID": "partyID-1",
  "currency": "BTC",
  "side": "BUY",
  "symbol": "BTC/USD",
  "partyID": "partyID",
}
```



```

    "transactTime": "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",
    "transactTime": "20181206-20:07:29.196",
    "orderQty": 1.0,
    "ordType": "LIMIT",
    "price": 3720.5,
    "stopPrice": 3720.4,
    "stopSide": "BID"
}

```

Response:

```

{
    "correlation": "foo",
    "type": "ExecutionReport",
    "orderID": "4956808967",
    "clOrdID": "partyID-1",
    "origClOrdID": "partyID-1",
    "execID": "280444_70150",
    "execType": "NEW",
    "ordStatus": "NEW",
    "ordRejReason": null,
    "account": "partyID",
    "symbol": "BTC/USD",
    "symbolSfx": null,
    "side": "BUY",
    "orderQty": 1,
    "ordType": "LIMIT",
    "price": 3720.5,
    "stopPrice": 0,
    "stopSide": null,
    "currency": null,
    "lastShares": 0,
    "lastPrice": 0,
    "lastSpotRate": 0,
    "leavesQty": 1,
    "cumQty": 0,
    "avgPrice": 0,

```



```
"tradeDate": null,
"transactTime": null,
"commission": 0,
"commCalculated": 18.6025,
"commType": "ABSOLUTE",
"commCurrency": "USD",
"minQty": 0,
"text": null,
"belowMin": null,
"matchingType": null,
"lastRptRequested": null,
"maxShow": 0.0,
"timeInForce": null,
"expireDate": null,
"partyIDs": ["partyID"]
}
```

4.12.3 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 :

```
{
  "correlation": "0x12309df",
  "type": "ReplaceStopLimitOrderSingleRequest",
  "handlInst": "AutomatedExecutionOrderPublic",
  "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 :

```
{
  "correlation": "foo1",
  "orderID": "erisxAssignedOrderId",
  "origClOrdID": "myClOrdId",
  "ordStatus": "Replaced",
  "execType": "Replaced",
  ...
}
```



4.12.4 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": "foo1",
  "orderID": "erisxAssignedOrderId",
  "origClOrdID": "myClOrdId",
  "ordStatus": "Canceled",
  "execType": "Canceled",
  ...
}
```

4.12.5 OrderMassStatusRequest

A JSON message should be submitted over the websocket client with **"type": "OrderMassStatusRequest"** in order to get all known working, filled, expired and cancelled orders for the past 48 hrs. The transacTime parameter is the starting time from when the request will begin looking for orders, not the current time of the request.

Request :

```
{
  "correlation": "foo1",
  "type": "OrderMassStatusRequest",
  "massStatusReqType": "partyID",
  "transacTime": "20181206-20:07:29"
}
```

Response: Execution report:

```
{
  "correlation": "foo1",
  "MassStatusReqID": "partyID",
  "orderID": "ord1",
  "clOrdID": "cl1",
  "LastRptRequested": "Y",
}
```



```
} ...
```

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."  
}
```