Cboe Digital Websocket API V4.0 Market Data

Please contact Cboe Digital sales representatives or Client Services for more information on this documentation.

Cboe Digital

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
20240316		Version 3.1
	API credentials, Authentication, Post-Only	Added new API permissions layout for Authentication Removed python2 example Added new Post-Only order type
20240519		Version 3.2
	Security Status	Added new SecurityStatus message workflow
	Market Data Response fields	Add new endFlag and numberOfOrders fields
20240617		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
20240721		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
20240817	Execution Report	Corrections: LastPx is lastPrice, AvgPx is avgPrice, senderLocationId is targetLocationId and senderSubId is targetSubId. Added minQty, belowMin, lastRptRequested, maxShow, sendingTime Deleted expireTime, stopSide
20240924		Version 3.5
	Security Status MarketDataIncrementalRefresh MarketDataIncrementalRefreshTrade	Added marketDataID field
20241009	MarketDataIncrementalRefresh MarketDataIncrementalRefreshTrade	Moved marketDataID field to the body of the message
20241023	WebSocket Public API Endpoint URL Websocket Algo Machine API Endpoint	Added testing and production endpoints for public websocket

	URL The Algo Machine Service ExecutionReports	Added testing and production endpoints for algo machine websocket Added new Algo Machine service Added new optional field algoType to ExecutionReports
20241116		Version 3.6
	Security Status	Added haltReason field
20210115	Pegged Order Order on Fill	Add details for Cancel Instructions
20210205	Order on Fill	Add Order on Fill Loop enhancement
20210503	Price Banding	Add Price banding description
20210902		Version 3.7
	Collateral Inquiry Ack Collateral Report	Added new messages to report on account balance information: Collateral Inquiry Ack and Collateral Report
	New Order fields New Order Single and Execution Report examples Supported Order Types	Added support for Market Orders
20211018		Version 3.8
	Market Data, Order Entry and The Algo Machine Service	Added requestId field in request and response message. Correlation field is deprecated and will be removed in future versions (Timeline to follow). To link requests and responses please use the requestId or the clOrdID fields.
20211028	TopOfBookMarketData	Added transactTime field
20211018		Version 3.9
	Unsolicited Messages	Add Connectivity and Trading Status messages
20220606	NewOrderSingle, Order Modification and Order Cancellation	Correlation field is required but will be deprecated in future versions
20220707	Authentication	Remove iat from Javascript payload example
20220830	Request Rejected	Add definition of message with type OrderReject or RequestReject
20231220	Futures Regulatory Tags	Updated descriptions for CTI code (CustOrderCapicity) tag 582
20240210	MarketDataIncrementalRefresh	Add Block Trades in Market Data. TickerType will no longer be available on the public market data service
20240826	Removed all references related to Spot trading	Removed all references related to Spot trading

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 Public Endpoints

Only the Real-Time Market Data Service will be available in the public websocket. These endpoints do not require authentication.

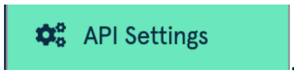
- **Testing - wss://publicmd-api.newrelease.erisx.com**
- **Production - wss://publicmd-api.erisx.com/**

2.2 WebSocket API Private Endpoints

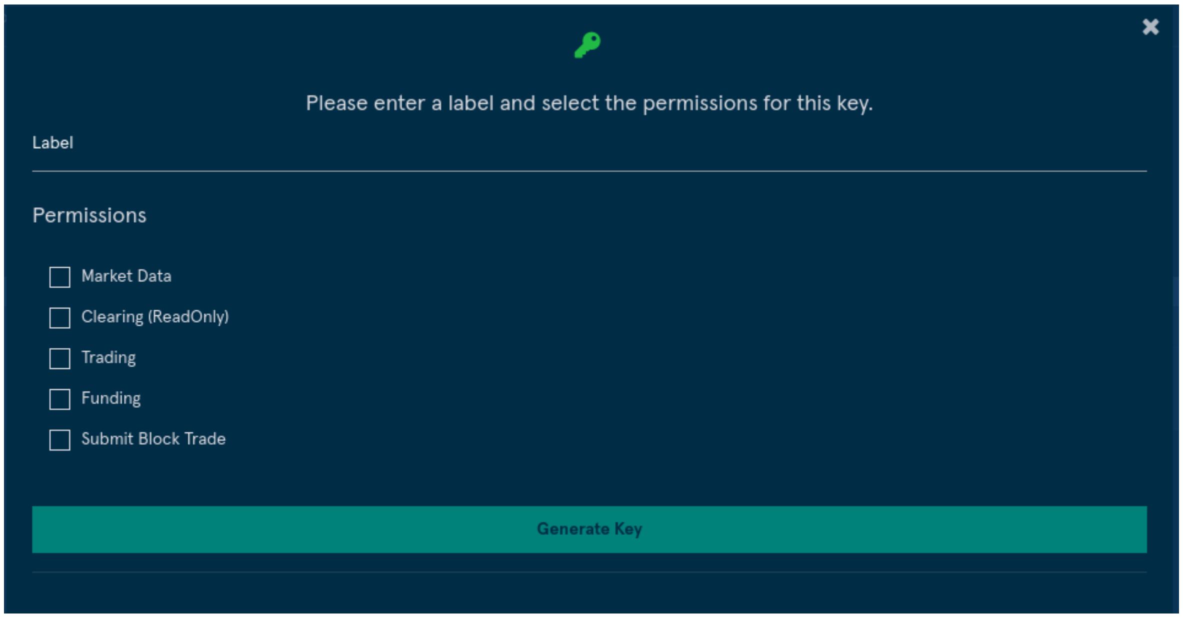
All endpoints listed in this section require authentication at the beginning of the session. See API Credentials and Authentication sections below for further details on how to successfully authenticate.

2.2.1 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.



Please enter a label and select the permissions for this key.

Label

Permissions

- [ ] Market Data
- [ ] Clearing (ReadOnly)
- [ ] Trading
- [ ] Funding
- [ ] Submit Block Trade

API Key permissions

- **Market Data:** An API key can query historical data or subscribe to real time data.
- **Trading:** **NO LONGER SUPPORTED**
- **Clearing (ReadOnly):** Allows an API key to query information about their clearing accounts (Documented Separately).
- **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**

2.2.2 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 in seconds.
- **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 = {
 sub: apiKey
};
var token = jwt.sign(payload, secret, { algorithm: 'HS256'});
```

Python 3 Example:

```
import jwt
```

```python
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 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	N	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. ***THIS FIELD IS DEPRECATED AND WILL STOP BEING SUPPORTED IN FUTURE VERSIONS. PLEASE USE requestId INSTEAD.**
requestId	Y	*USE THIS FIELD INSTEAD OF correlation. The provided request id 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 40.
type	Y	AuthenticationRequest
token	C	Jwt token generated using the method described above

Example request:

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

Example response:

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

2.3 Unsolicited Messages

The WebAPI server will send the following unsolicited messages:

2.3.1 Connectivity

It indicates connectivity status. It will be sent whenever there is an event that changes the connectivity status.

Field	Req	Value
type	Y	Message type: **CONNECTIVITY**
connection	Y	Type of connection to which the message refers: **MARKET_DATA**
connectionStatus	Y	Indicates the connectivity status: **ONLINE**, **OFFLINE**
requestId	Y	**unsolicited**
correlation	N	**unsolicited**. This value is deprecated and will be removed in future versions

```
{
    "type": "CONNECTIVITY",
    "connection": "MARKET_DATA",
    "connectionStatus": "OFFLINE",
    "requestId": "unsolicited",
    "correlation": "unsolicited"
}
```

2.3.2 Trading Status

It indicates state transitions

Field	Req	Value
type	Y	Message type: **TRADING_STATUS**
connection	Y	Type of connection to which the message refers: **MARKET_DATA**
tradingSessionStatusCode	Y	**1**=HALTED, **2**=OPEN, **3**=CLOSED, **4**=PRE_OPEN, **5**=PRE_CLOSE, **6**=REQUEST_REJECTED, **101**=SYSTEM_READY, **102**=OPEN_TRADING, **103**=CLOSE_TRADING, **104**=PRE_OPEN_TRADING, **105**=SYSTEM_DISCONNECT
tradingSessionStatusDescription	Y	HALTED, OPEN, CLOSED, PRE_OPEN,

		PRE_CLOSE, REQUEST_REJECTED, SYSTEM_READY, OPEN_TRADING, CLOSE_TRADING, PRE_OPEN_TRADING, SYSTEM_DISCONNECT
requestId	Y	**unsolicited**

```
{
    "type": "TRADING_STATUS",
    "tradingSessionStatusCode": 101,
    "tradingSessionStatusDescription": "SYSTEM_READY",
    "connection": "MARKET_DATA",
    "requestId": "unsolicited",
    "correlation": "unsolicited"
}
```

2.4 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.

2.5 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
AuthenticationRequest	1
SecurityList	20
MarketDataSubscribe	1
MarketDataStatus	1
MarketDataUnsubscribe	1
TopOfBookMarketDataSubscribe	1
TopOfBookMarketDataUnsubscribe	1

Example response:

```
{"requestId":"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"}
```

2.6 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.

2.7 Request Rejected

If the system in charge processing a request is not available at the time when the request is submitted, the system will respond with a reject message indicating what resource is unavailable. When receiving this response, the user should wait for a small period of time before attempting to send the request again.

Field	Req	Value
type	Y	**RequestRejected**
requestType	Y	Type of the message sent in the request
correlation	N	Value provided by the clearing member request for the request. ***THIS FIELD IS DEPRECATED AND WILL STOP BEING SUPPORTED IN FUTURE VERSIONS. PLEASE USE requestId INSTEAD.**
requestId	N	Value provided by the user in the request
message	Y	Reason for rejection
connectionStatuses	Y	Map containing the unavailable resource and its status

Example:

```
{
    "type": "RequestReject",
    "clOrdID": "12345",
    "message": "MARKET_DATA - OFFLINE",
    "requestType": "MarketDataSubscribe",
    "connectionStatuses":
    {
        "MARKET_DATA": "OFFLINE"
    },
    "requestId": "abcd12",
    "correlation": "abcd12"
}
```

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
requestId	Y	*USE THIS FIELD INSTEAD OF correlation. The provided request id 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 40.
type	Y	The data subscription type
symbol	C	Product code i.e. BTCU24

Example:

```
"correlation": "123456789abcdefg", "type": "MarketDataSubscribe", "symbol": “BTCU24"
```

3.2 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:

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

Response:

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

3.3 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**: Cboe Digital will return all active instruments Other value, Cboe Digital will return all active instruments where securityGroup matches the requested value

		If securityGroup is not specified, Cboe Digital will only return a default subset of contracts

Security List Response

Field	Req	Value
requestId	Y	*USE THIS FIELD INSTEAD OF correlation. The provided request id 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 40.
symbol	Y	Instrument (E.g. BTCU24)
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

Request:

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

Response:

```
{
    "requestId": "12345abc",
    "securities": [
     {
      "currency": "BTC",
      "symbol": "BTCU24",
      "symbolSfx": null,
      "securityDesc": "BTCU24",
      "minTradeVol": 1,
      "maxTradeVol": 100000,
      "roundLot": 1,
      "minPriceIncrement": 1,
      "product": "COMMODITY",
      "cfiCode": "FCXXSX",
      "securityType": "FUT",
      "maturityMonthYear": "202409",
      "contractMultiplier": 0.1,
      "securityExchange": "ERISX",
      "activation": "20240927",
      "lastEligableTradeDate": "20241003",
      "maturityDate": "20240927",
      "lastTradeTime": "15:00:00Z",
      "expiryTime": "15:00:00Z",
      "productCode": null,
      "securityGroup": null,
      "cap": null,
      "floor": null
    },...]
}
```

3.4 Request Status

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

Example:

```
{
  "requestId": "abc123",
  "type": "STATUS",
  "message": "Subscribed to market data for BTCU24."
}
```

3.5 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

Security Status Response

Field	**Req**	**Value**
requestId	O	*USE THIS FIELD INSTEAD OF correlation. The provided request id 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 40.
type	Y	SecurityStatus
security	Y	Contract specification as described in Security List
securityTradingStatus	Y	Current Contract Trading Status: **READY_TO_TRADE_START_OF_SESSION**, **NOT_AVAILABLE_FOR_TRADING_END_OF_SESSION**, **TRADING_HALT**, **PRE_OPEN**
sessionEnd	Y	Status which indicates that a trading session has ended and statistics for the trading session should be reset
sendingTime	Y	Time at which the message was published from Cboe Digital
transactTime	Y	Time at which the trading engine performed an action
marketDataID	N	Sequence number which uniquely identifies all unsolicited market data messages within a trade date, for example MarketDataIncrementalRefresh, MarketDataIncrementalRefreshTrade and SecurityStatus. Messages containing the same Global Market Data ID within a Trade Date should be considered as duplicates.
haltReason	N	Denotes the reason for the Trading Halt. Present when securityTradingStatus = TRADING_HALT

Example:

```
{
  "requestId": "15978405223302",
  "type": "SecurityStatus",
  "security": {
    "currency": "ETH",
    "symbol": "ETHU24",
    "symbolSfx": null,
    "securityDesc": "ETHU24",
    "minTradeVol": 1,
    "maxTradeVol": 10000,
    "roundLot": 1,
    "minPriceIncrement": 0.10,
    "product": "COMMODITY",
    "cfiCode": "FCXXSX",
    "securityType": "FUT",
    "maturityMonthYear": "202409",
    "contractMultiplier": 1,
    "securityExchange": "ERISX",
    "activation": "20240520",
    "lastEligableTradeDate": "20240927",
    "maturityDate": "20240927",
    "lastTradeTime": "15:00:00Z",
    "expiryTime": "15:00:00Z",
    "productCode": null,
    "securityGroup": null,
    "cap": null,
    "floor": null
  },
  "transactTime": "20240818-22:00:00.010000000",
  "securityTradingStatus": "READY_TO_TRADE_START_OF_SESSION",
  "sessionEnd": null,
  "sendingTime": "20240818-22:00:00.042",
  "marketDataID": 1234,
  "haltReason": "EQUIPMENT_CHANGEOVER",
}
```

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.*

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
requestId	Y	*USE THIS FIELD INSTEAD OF correlation. The provided request id 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 40.
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 'lifts' a bid price. "": No tickerType value will be populated for block trades null: This field will not be populated on the public market data service and will always be set to null
marketDataID	C	Sequence number which uniquely identifies all unsolicited market data messages within a trade date, for example MarketDataIncrementalRefresh, MarketDataIncrementalRefreshTrade and SecurityStatus. Messages containing the same Global Market Data ID within a Trade Date should be considered as duplicates.
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.

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:

```
{
    "requestId": "15753832469890",
    "type": "MarketDataSubscribe",
    "symbol": “BTCU24"
}
```

Response - Snapshot:

```
{
  "requestId": "15978405223302",
  "type": "MarketDataIncrementalRefresh",
  "symbol": "ETBTQ0",
  "sendingTime": "20240819-04:26:36.406",
  "marketDataID": 16265510900,
  "bids": [
    {
      "id": "1000000563630",
      "updateAction": "NEW",
      "price": 0.03514,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "100000056369d",
      "updateAction": "NEW",
      "price": 0.03513,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "100000056369e",
      "updateAction": "NEW",
      "price": 0.03512,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "10000005635f7",
      "updateAction": "NEW",
      "price": 0.03511,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "10000005636ee",
      "updateAction": "NEW",
      "price": 0.0351,
      "amount": 5,
      "symbol": "ETBTQ0",
    }
  ],
  "offers": [
    {
      "id": "100000056369f",
      "updateAction": "NEW",
      "price": 0.03519,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "10000005636a0",
      "updateAction": "NEW",
      "price": 0.0352,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "1000000563666",
      "updateAction": "NEW",
      "price": 0.03521,
      "amount": 5,
      "symbol": "ETBTQ0",
    },
    {
      "id": "10000005636ef",
      "updateAction": "NEW",
      "price": 0.03522,
      "amount": 5,
      "symbol": "ETBTQ0",

    }
  ],
  "transactTime": "20240819-04:26:36.382668739",
  "endFlag": null
}
```

Response - Incremental updates

```
{
  "requestId": "269980392094877",
  "type": "MarketDataIncrementalRefresh",
  "symbol": "BTCU24",
  "sendingTime": "20240925-15:55:28.165",
  "marketDataID": 16265510912,
  "bids": [
    {
      "id": "1000000000003",
      "updateAction": "NEW",
      "price": 800,
      "amount": 0.1,
      "symbol": "BTCU24",
    }
  ],
  "offers": [],
  "transactTime": "20240925-15:55:28.093490622",
  "endFlag": "END_OF_EVENT"
}
```

Response - Trade updates:

```
{
  "requestId": "15978410832102",
  "type": "MarketDataIncrementalRefreshTrade",
  "symbol": "BTCU24",
  "sendingTime": "20240819-12:44:50.896",
  "marketDataID": 16265510913,
  "trades": [
    {
      "updateAction": "NEW",
      "price": 64.2,
      "currency": "LTC",
      "tickerType": "PAID",
      "transactTime": "20240819-12:44:50.872994129",
      "size": 2.0,
      "symbol": "BTCU24",
      "numberOfOrders": 1,
    }
  ],
  "endFlag": "END_OF_TRADE"
}
```

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:

```
{
    "requestId": "15753904509040",
    "type": "MarketDataSubscribe",
    "tradeOnly": true,
    "symbol": "BTCU24"
}
```

Response - Trade updates:

```
{
  "requestId": "15978410832102",
  "type": "MarketDataIncrementalRefreshTrade",
  "symbol": "BTCU24",
  "sendingTime": "20240819-12:44:50.896",
  "marketDataID": 16265510914,
  "trades": [
    {
      "updateAction": "NEW",
      "price": 64.2,
      "currency": "BTC",
      "tickerType": "PAID",
      "transactTime": "20240819-12:44:50.872994129",
      "size": 2.0,
      "symbol": "BTCU24",
      "numberOfOrders": 1,
    }
  ],
  "endFlag": "END_OF_TRADE"
}
```

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:

```
{
  "requestId": "abc456",
  "type": "MarketDataUnsubscribe",
  "symbol": “BTCU24"
}
```

Response:

```
{
  "requestId": "abc456",
  "type": "INFO_MESSAGE",
  "message": "Unsubscribed from market data forBTCU24."
}
```

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:

```
{
    "requestId": "abc123",
    "type": "TopOfBookMarketDataSubscribe",
    "symbol": “BTCU24",
    "topOfBookDepth": 3
}
```

Response:

```
{
  "requestId": "15978412650812",
  "type": "TopOfBookMarketData",
  "bids": [
    {
      "action": "NEW",
      "count": 1,
      "totalVolume": 1.0,
      "price": 413.2,
      "lastUpdate": "20240819-12:47:49.975",
      "transactTime": "20240819-12:47:49.975123456"
    },
    {
      "action": "UPDATE",
      "count": 2,
      "totalVolume": 2.00,
      "price": 412.9,
      "lastUpdate": "20240819-12:47:39.984",
      "transactTime": "20240819-12:47:39.984123456"
    }
  ],
  "offers": [
    {
      "action": "NO CHANGE",
      "count": 1,
      "totalVolume": 1.00,
      "price": 413.3,
      "lastUpdate": "20240819-12:47:40.166",
      "transactTime": "20240819-12:47:40.166123456"
    },
    {
      "action": "NO CHANGE",
      "count": 1,
      "totalVolume": 1.56,
      "price": 413.4,
      "lastUpdate": "20240819-12:47:20.196",
      "transactTime": "20240819-12:47:20.196123456"
    }
  ],
  "symbol": “BTCU24"
}
```

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:

```
{
  "requestId": "abc456",
  "type": "TopOfBookMarketDataUnsubscribe",
  "symbol": “BTCU24"}
```

Response:

```
{
  "requestId": "abc456",
  "type": "INFO_MESSAGE",
  "message": "Unsubscribed from top of book market data for BTCU24."
}
```