**WebSocket API Documentation**

This document outlines the WebSocket API for both private (user account) and public (market data) information.

**(Private WebSocket) User Account Information WebSocket Interface**

**Description**

**01.** Private WebSocket connections do not require subscriptions; data is automatically pushed after a successful connection. This includes both trading messages and custom messages.

**02.** Trading messages are identified with the type type-event. Other message types will be defined separately.

**03.** The event field within the body of a trading message can be one of the following: Snapshot, ACCOUNT\_UPDATE, DEPOSIT\_UPDATE, WITHDRAW\_UPDATE, TRANSFER\_IN\_UPDATE, TRANSFER\_OUT\_UPDATE, ORDER\_UPDATE, FORCE\_WITHDRAW\_UPDATE, FORCE\_TRADE\_UPDATE, FUNDING\_SETTLEMENT, ORDER\_FILL\_FEE\_INCOME, START\_LIQUIDATING, FINISH\_LIQUIDATING, or UNRECOGNIZED.

**04.** Ping-Pong Mechanism:

**Server Ping (Heartbeat):**

After a successful WebSocket connection, the server sends a Ping message at a fixed interval. The message body looks like: {"type":"ping","time":"1693208170000"}. The time field is the server's timestamp when the Ping was sent.

The client must respond with a Pong message upon receipt, with a body like: {"type":"pong","time":"1693208170000"}.

If the server doesn't receive a Pong response after 5 consecutive Pings, the server will terminate the connection.

**Client Ping (Latency Measurement):**

After a successful WebSocket connection, the client can also initiate a Ping message with a body like: {"type":"ping","time":"1693208170000"}. The time field is the client's timestamp when the Ping was sent.

The server will immediately respond with a Pong message, with a body like: {"type":"pong","time":"1693208170000"}. The time field in the Pong will match the time field in the client's Ping.

**05.** Authentication:

**Web:**

Browsers don't allow custom headers during WebSocket connections, so special handling is required.

Use the same authentication logic as HTTP. Create a JSON string using the X-edgeX-Api-Signature, X-edgeX-Api-Timestamp key-value pairs, for example: {"X-edgeX-Api-Signature": "00e6b34cf9c3c0ca407cc2fe149fad836206c97201f236137c0e89fd079760470672b5257fa372710b5863d1ec6e0215e5bd6b2c3a319eda88886250a100524706ea3dd81a7fc864893c8c6f674e4a4510c369f939bdc0259a0980dfde882c2d", "X-edgeX-Api-Timestamp": "1705720068228"}.

Base64 encode this JSON string.

During the WebSocket request, pass the base64 encoded value in the SEC\_WEBSOCKET\_PROTOCOL header.

**App/API:**

App/API WebSocket connections can use [custom](https://edgex-1.gitbook.io/edgeX-documentation/api/authentication" \l "private-api) headers. Therefore, Apps/API can continue using the same authentication logic as HTTP, or they can use the Web authentication method described above.

WebSocket is a GET request and there is no need to sign the request body.

**URL: /api/v1/private/ws**

**Payload**

Copy

{ // The type for trading messages is "trade-event". Custom messages have their own defined type. "error" indicates an error message sent by the server. "type": "trade-event", // The body of a trading message has the structure below. The message structure for custom messages will be defined separately by the user. "content": { // The event that triggered the data update "event": "ACCOUNT\_UPDATE", // Data update version "version": "1000", // Data "data": { // Account information "account": [ ], // Collateral information "collateral": [ ], // Collateral transaction details "collateralTransaction": [ ], // Position information "position": [ ], // Position transaction details "positionTransaction": [ ], // Deposit records "deposit": [ ], // Withdrawal records "withdraw": [ ], // Transfer in records "transferIn": [ ], // Transfer out records "transferOut": [ ], // Order information "order": [ ], // Trade details "orderFillTransaction": [ ] } }}

**(Public WebSocket) Market Data WebSocket Interface**

**URL: /api/v1/public/ws**

**Description**

**01.** When subscribing or unsubscribing, the server will validate the channel. For invalid channels, the server will respond with an error message, for example: {"type":"error","content":{"code":"INVALID\_CONTRACT\_ID""msg":"invalid contractId:100000001"}}

**02.** The message structure for subscribing and unsubscribing is: {"type": "subscribe", "channel": "ticker.10000001"}.

**03.** Ping-Pong Mechanism:

**Server Ping (Heartbeat):**

After a successful WebSocket connection, the server sends a Ping message at a fixed interval. The message body looks like: {"type":"ping","time":"1693208170000"}. The time field is the server's timestamp when the Ping was sent.

The client must respond with a Pong message upon receipt, with a body like: {"type":"pong","time":"1693208170000"}.

If the server doesn't receive a Pong response after 5 consecutive Pings, the server will terminate the connection.

**Client Ping (Latency Measurement):**

After a successful WebSocket connection, the client can also initiate a Ping message with a body like: {"type":"ping","time":"1693208170000"}. The time field is the client's timestamp when the Ping was sent.

The server will immediately respond with a Pong message, with a body like: {"type":"pong","time":"1693208170000"}. The time field in the Pong will match the time field in the client's Ping.

**Subscription Metadata**

**Request**

Copy

{ "type": "subscribe", "channel": "metadata"}

**Response**

Copy

{ "type": "subscribed", "channel": "metadata"}

**Payload**

Copy

{ // error "type": "quote-event", "channel": "metadata", "content": { // snapshot quote-event "dataType": "Snapshot", // "channel": "metadata", "data": [ { // Coin information "coin": [ ], // Contract information "contract": [ ] } ] }}

**Subscribe to 24-Hour Market Ticker**

**Channel Explanation**

Channel

Description

ticker.{contractId}

Subscribe to the ticker of contract contractId

ticker.all

Subscribe to the ticker of all contracts

ticker.all.1s

Subscribe to the ticker of all contracts (periodic push)

**Request**

Copy

{ "type": "subscribe", "channel": "ticker.10000001"}

**Response**

Copy

{ "type": "subscribed", "channel": "ticker.10000001"}

**Payload**

Copy

{ "type": "payload", "channel": "ticker.10000001", "content": { "dataType": "Snapshot", "channel": "ticker.10000001", "data": [ { "contractId": "string", "priceChange": "string", "priceChangePercent": "string", "trades": "string", "size": "string", "value": "string", "high": "string", "low": "string", "open": "string", "close": "string", "highTime": "string", "lowTime": "string", "startTime": "string", "endTime": "string", "lastPrice": "string" } ] }}

**Subscribe to K-Line Data**

**Channel Explanation**

Channel

Description

kline.{priceType}.{contractId}.{interval}

Subscribe to the interval K-Line of contract contractId based on priceType

**priceType Parameter**

Value

Description

LAST\_PRICE

Last Price K-Line

MARK\_PRICE

Mark Price K-Line

**interval Parameter**

Value

Description

MINUTE\_1

1-Minute K-Line

MINUTE\_5

5-Minute K-Line

MINUTE\_15

15-Minute K-Line

MINUTE\_30

30-Minute K-Line

HOUR\_1

1-Hour K-Line

HOUR\_2

2-Hour K-Line

HOUR\_4

4-Hour K-Line

HOUR\_6

6-Hour K-Line

HOUR\_8

8-Hour K-Line

HOUR\_12

12-Hour K-Line

DAY\_1

Daily K-Line

WEEK\_1

Weekly K-Line

MONTH\_1

Monthly K-Line

**Request**

Copy

{ "type": "subscribe", "channel": "kline.LAST\_PRICE.10000001.MINUTE\_1"}

**Response**

Copy

{ "type": "subscribed", "channel": "kline.LAST\_PRICE.10000001.MINUTE\_1"}

**Payload**

Copy

{ "type": "payload", "channel": "kline.LAST\_PRICE.10000001.MINUTE\_1", "content": { "dataType": "Changed", "channel": "kline.LAST\_PRICE.10000001.MINUTE\_1", "data": [ { "klineId": "1", "contractId": "10000001", "klineType": "MINUTE\_1", "klineTime": "1688365544504", "trades": "5", "size": "10.1", "value": "100000", "high": "31200", "low": "31000", "open": "3150", "close": "31010", "makerBuySize": "5", "makerBuyValue": "150000" } ] }}

**Subscribe to Order Book**

**Usage Instructions**

After a successful subscription, a full dataset is pushed once initially (depthType=SNAPSHOT), and subsequent pushes will be incremental updates (depthType=CHANGED).

**Channel Explanation**

Channel

Description

depth.{contractId}.{depth}

Subscribe to the order book of contract contractId with a depth of depth

**depth Parameter**

Value

Description

15

15 levels

200

200 levels

**Request**

Copy

{ "type": "subscribe", "channel": "depth.10000001.15"}

**Response**

Copy

{ "type": "subscribed", "channel": "depth.10000001.15"}

**Payload**

Copy

{ "type": "payload", "channel": "depth.10000001.15", "content": { "dataType": "Snapshot", "channel": "depth.10000001.15", "data": [ { "startVersion": "string", "endVersion": "string", "level": 0, "contractId": "10000001", "depthType": "Snapshot", // Data type: SNAPSHOT for full data, CHANGED for incremental data "bids": [ [ "26092", // Price "0.9014" // Size. A size of 0 indicates a deletion. Positive numbers mean increase. Negative numbers mean decrease. ], [ "26091", "0.9667" ] ], "asks": [ [ "26093", "0.964" ], [ "26094", "1.0213" ] ] } ] }}

**Subscribe to Latest Trades**

**Channel Explanation**

Channel

Description

trades.{contractId}

Subscribe to the latest trades of contract contractId

**Request**

Copy

{ "type": "subscribe", "channel": "trades.10000001"}

**Response**

Copy

{ "type": "subscribed", "channel": "trades.10000001"}

**Payload**

Copy

{ "type": "payload", "channel": "trades.10000001", "content": { "dataType": "Changed", "channel": "trades.10000001", "data": [ { "ticketId": "1", "time": "1688365544504", "price": "30065.12", "size": "0.01", "value": "300.6512", "takerOrderId": "10", "makerOrderId": "11", "takerAccountId": "3001", "makerAccountId": "3002", "contractId": "10000001", "isBestMatch": true, "isBuyerMaker": false } ] }}

[PreviousAsset API](https://edgex-1.gitbook.io/edgeX-documentation/api/private-api/asset-api)