Peter Gustafson
writing sample
Coinbase Mar 10, 2022

Exchange WebSocket Overview | Coinbase Cloud
The WebSocket feed is publicly available and provides real-time market data updates for orders and trades. Two endpoints are supported in both production and sandbox:

*   **Coinbase Market Data** is our traditional feed which is available without authentication.
*   **Coinbase Direct Market Data** has direct access to Coinbase Exchange servers and requires [Authentication](https://docs.cloud.coinbase.com/exchange/docs/websocket-auth).

tip

You can subscribe to both endpoints, but if `ws-direct` is your primary connection, we recommend using `ws-feed` as a failover.

info

Coinbase Market Data
production = wss://ws-feed.exchange.coinbase.com
sandbox = wss://ws-feed-public.sandbox.exchange.coinbase.com

Coinbase Direct Market Data
production = wss://ws-direct.exchange.coinbase.com
sandbox = wss://ws-direct.sandbox.exchange.coinbase.com

Protocol[](#protocol "Direct link to Protocol")

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

tip

New message types can be added at any time. Clients are expected to ignore messages they do not support.

Subscribe[](#subscribe "Direct link to Subscribe")

To begin receiving feed messages, you must send a `subscribe` message to the server indicating which channels and products to receive. This message is mandatory—you are disconnected if no `subscribe` has been received within 5 seconds.

caution

To receive feed messages, you must send a `subscribe` message or you are disconnected in 5 seconds.

```

{
    "type": "subscribe",
    "product_ids": [
        "ETH-USD",
        "ETH-EUR"
    ],
    "channels": [
        "level2",
        "heartbeat",
        {
            "name": "ticker",
            "product_ids": [
                "ETH-BTC",
                "ETH-USD"
            ]
        }
    ]
}

```

You receive a `subscriptions` message as a response to an `subscribe` message.

Unsubscribe[](#unsubscribe "Direct link to Unsubscribe")

To unsubscribe from channel/product pairs, send an `unsubscribe` message. The structure is equivalent to `subscribe` messages.

tip

You can also unsubscribe from a channel entirely by providing no product IDs.

```

{
    "type": "unsubscribe",
    "channels": [
        "heartbeat"
```

```
    ]
}

```

You receive a `subscriptions` message as a response to an `unsubscribe` message.

Specifying Product IDs[](#specifying-product-ids "Direct link to Specifying Product IDs")

There are two ways to specify the product IDs to listen for within each channel:

*   You can define product IDs for an individual channel.
*   You can define product IDs at the root of the object—this adds them to all the channels you subscribe to.

```

{
    "type": "unsubscribe",
    "product_ids": [
        "ETH-USD",
        "ETH-EUR"
    ],
    "channels": [
        "ticker"
    ]
}

```

Subscriptions Message[](#subscriptions-message "Direct link to Subscriptions Message")

A `subscriptions` message is sent in response to both [subscribe](#subscribe) and [unsubscribe](#unsubscribe) messages.

In response to a `subscribe` message, the `subscriptions` message lists all channels you are subscribed to. Subsequent subscribe messages add to the list of subscriptions. If you subscribed to a channel without being authenticated, you will remain in the unauthenticated channel.

```

{
```

```
    "type": "subscriptions",
    "channels": [
        {
            "name": "level2",
            "product_ids": [
                "ETH-USD",
                "ETH-EUR"
            ],
        },
        {
            "name": "heartbeat",
            "product_ids": [
                "ETH-USD",
                "ETH-EUR"
            ],
        },
        {
            "name": "ticker",
            "product_ids": [
                "ETH-USD",
                "ETH-EUR",
                "ETH-BTC"
            ]
        }
    ]
}

```

Websocket Compression Extension[](#websocket-compression-extension "Direct link to Websocket Compression Extension")

Websocket compression, defined in RFC7692, compresses the payload of WebSocket messages which can increase total throughput and potentially reduce message delivery latency. The **permessage-deflate extension** can be enabled by adding the extension header. Currently, it is not possible to specify the compression level.

From [RFC7692](https://datatracker.ietf.org/doc/html/rfc7692#section-7.1.3):

The simplest "Sec-WebSocket-Extensions" header in a client (or server's) opening handshake to offer (or accept) use of the "permessage-deflate" extension looks like this:

```
    GET wss://ws-feed.exchange.coinbase.com
```

```
       Sec-WebSocket-Extensions: permessage-deflate
```

Sequence Numbers[](#sequence-numbers "Direct link to Sequence Numbers")

Most feed messages contain a sequence number. Sequence numbers are increasing integer values for each product, with each new message being exactly one sequence number greater than the one before it.

Sequence numbers that are _greater than one integer value_ from the previous number indicate that a message has been dropped. Sequence numbers that are _less_ than the previous number can be ignored or represent a message that has arrived out of order.

In either situation you may need to perform logic to make sure your system is in the correct state.

caution

Even though a WebSocket connection is over TCP, the WebSocket servers receive market data in a manner that can result in dropped messages. Your feed consumer should be designed to handle sequence gaps and out of order messages, or should use channels that guarantee delivery of messages.

tip

To guarantee that messages are delivered and your order book is in sync, consider using the [level2 channel](https://docs.cloud.coinbase.com/exchange/docs/websocket-channels#level2-channel).