API WebSocket User Guide

1 Introduction

1.1 Purpose

An API, or application programming interface, is a set of defined rules that enable different applications to communicate with each other. The purpose of the document is to define the API specifications and provide guidance to readers during the process of API integration. tastyfx offers clients a set of APIs to build their own integration programs specifically designed for tastyfx Forex/RFED Forex trading services. This document explains the parameters and responses for each method with a sample template for each API request.

Here is the list of functionalities which you can access through our APIs:

- **Pre-Trade**: To query for security reference data and to request pricing quotes.
- **Trade**: For placing orders and retrieving order status.
- **Post-Trade**: For position updates and account data.
- **Chart data**: For real time candle data and historical candle data.

1.2 Audience

This help document assumes a level of familiarity with programming. The document is recommended for readers with an understanding of API implementation and writing code using WebSockets. It is primarily targeted at developers who are familiar with the technical aspects of integrating APIs for trading purposes.

1.3 Prerequisites

- A DEMO or LIVE tastyfx Forex account.
- A secured WebSocket over Hypertext Transfer Protocol Secure (HTTPS).

2 Getting started

To begin using our WebSocket APIs, you require the following:

- Trading Account
- Base URL

2.1 Trading account

To use our WebSocket API, you need an tastyfx trading account. The WebSocket API is supported for both DEMO and LIVE accounts. If you don't already have an tastyfx trading account, click here to create a tastyfx trading account.

2.1.1 Demo account

A demo account is a type of trading account provided by tastyfx to allow traders to practice and familiarize themselves with the trading platform and forex markets without risking real money. In the demo account, traders use virtual funds for practice purposes.
Note: You can use demo account only with the demo endpoints.

2.1.2 Live account

A live or production account is an tastyfx trading account that involves real money and allows traders to engage in actual trading in the forex markets.
Note: You can use live account only with the production endpoints.

2.2 Base URL

2.2.1 PreTrade URL

- **Demo URL**: wss://demo-iguspretrade.ig.com/pretrade
- **Production URL**: wss://iguspretrade.ig.com/pretrade

2.2.2 Trade URL

- **Demo URL**: wss://demo-igustrade.ig.com/trade
- **Production URL**: wss://igustrade.ig.com/trade

2.2.3 PostTrade URL

- **Demo URL**: wss://demo-igustrade.ig.com/trade
- **Production URL**: wss://igustrade.ig.com/trade

2.2.4 ChartData URL

- **Demo URL**: wss://demo-iguspretrade.ig.com/pretrade
- **Production URL**: wss://iguspretrade.ig.com/pretrade

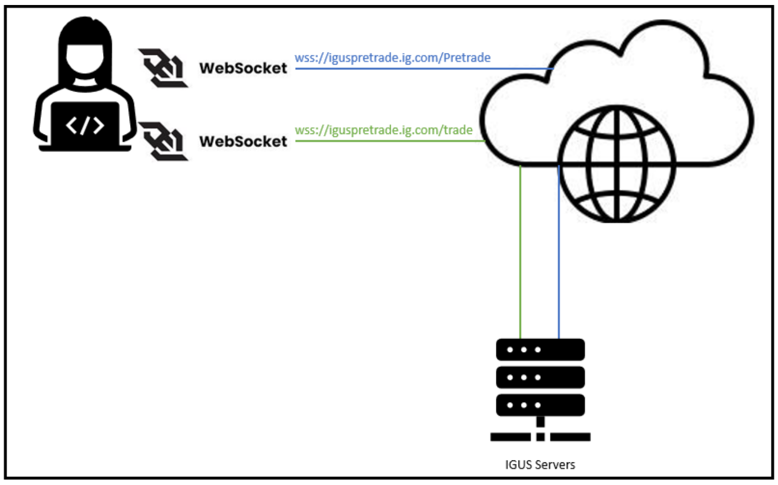

WebSocket
wss://iguspretrade.ig.com/Pretrade
WebSocket
wss://iguspretrade.ig.com/trade
IGUS Servers

Figure 1: tastyfx WebSocket-Server Workflow

2.3 Ports

You must use port 433 for all WebSocket connections. By default, port 433 is designated for all WebSocket connections.

3 WebSocket API

3.1 Introduction

The WebSocket API is an advanced technology that enables an interactive communication session between the client and a server. WebSocket connection is an ideal choice for applications that require real-time updates, such as stock market tickers.

WebSocket APIs provide real-time capabilities and asynchronous communication unlike REST APIs that rely solely on request-response interactions. This makes them particularly well-suited for applications that need quick and responsive updates. For an example, the WebSocket API can continuously provide pricing updates to a client upon their quote request. It operates by sending and receiving JSON payloads based on the FIX Protocol Application Message. This enables seamless and efficient data exchange between client and server.

3.2 FIX protocol

The Financial Information eXchange (FIX) is an information and data protocol used by financial institutions. FIX protocol standard is maintained by FIX trading community. The JSON messages sent and received by the tastyfx WebSocket API are consistent with the FIX protocol.

3.3 Types of WebSocket APIs

IG Group offers clients three Application Programming Interfaces (API), which are based on FIX Protocol Application Messages.

- **PreTrade**: This is for security reference data and quote negotiation. To know more about PreTrade, click here.
- **Trade**: This is for trading and order status. To know more about Trade, click here.
- **PostTrade**: This is for position management and account data. To know more about PostTrade, click here.

IG Group also offers a separate chart data WebSocket API, which is not based on FIX Protocol Application Messages:

- **Streaming chart candle data**: This is for retrieving the continuous and real-time update of candle chart information. To know more about HistoricChartData, click here.
- **Historic chart candle data**: This is for retrieving historic prices. To know more about HistoricChartData, click here.

3.4 Connect to WebSocket API

In order to utilize the functionalities provided by the WebSocket API, you must establish a logical session to the WebSocket endpoint. The initial step is setting up a connection between the WebSocket and IG server.

Before connecting to the WebSocket, it is essential to:

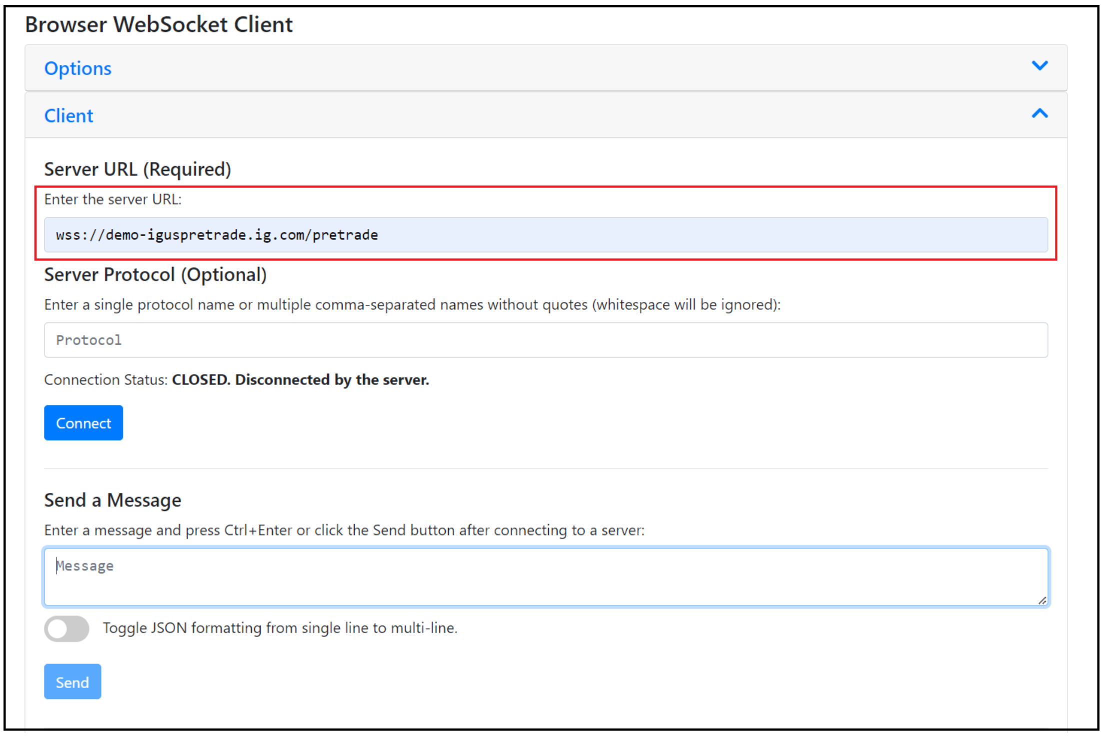
- Determine the correct URL based on the environment and the type of functionality you want to access.
- Open a HTTPS WebSocket connection.
- Create a logical session.

In the below scenario, we are connecting to the DEMO environment and utilizing the PreTrade endpoint. The URL begins with "wss" which indicates that it is a WebSocket secure connection.

Disclaimer: To familiarize you with the process of connecting to the WebSocket API, we will demonstrate connecting WebSocket using a **Browser WebSocket Client** application (a Google Chrome Plugin). It's important to note that this is solely for educational purpose and is not an tastyfx recommendation.

To connect to the WebSocket, perform the following actions:

1. Open **Browser WebSocket Client** application.
2. Click **Client** drop-down.
3. Under **Client**, go to **Server URL (required)**.
4. In the **Enter the server URL** text box, enter the demo URL**: wss://demo-iguspretrade.ig.com/pretrade**.


Browser WebSocket Client

Server URL (Required)

Enter the server URL:

```
wss://demo-iguspretrade.ig.com/pretrade
```

Server Protocol (Optional)

Enter a single protocol name or multiple comma-separated names without quotes (whitespace will be ignored):

Connection Status: **CLOSED. Disconnected by the server.**

Send a Message

Enter a message and press Ctrl+Enter or click the Send button after connecting to a server:

Toggle JSON formatting from single line to multi-line.

Figure 2: Browser WebSocket Client Chrome Extension

5. Click **Connect** to create the connection with WebSocket.

3.4.1 Create a logical session

After a WebSocket connection is established, then you need to create a logical session. Creating a logical session is a 2-phase process, which includes, **Negotiate** and **Establish** phases. The FIX/P (FIX Performance Session Layer) standard provides the messages required for establishing a logical session.

Here are the 2 phases of creating a logical session:

1. **Negotiate**
   - The client must provide a session ID as a reference to the session in future messages.
   - The client's username and password credentials are required for authentication during the login process.
2. **Establish**
   - This phase is to set the client's heartbeat interval to keep the session alive.

Note: When you attempt to establish a WebSocket connection, it is important to note that there is a 30-second time limit for completing the negotiate and establish process. If you exceed the time limit, the session gets auto terminated and you need to re-initiate the process.

Here is a diagram illustrating the **Create a logical session** workflow of WebSocket API:

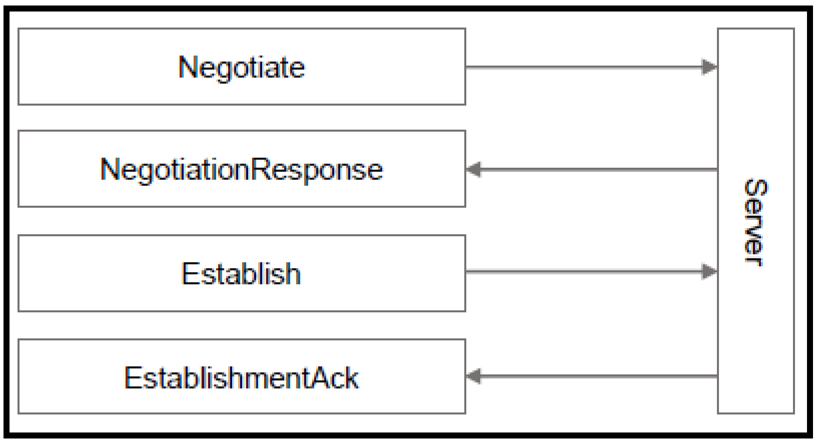


Figure 3: Create a logical session workflow

3.4.1.1 Negotiate a message

After connection is successfully made with the WebSocket, the next step is to send a **Negotiate** message. The purpose of this action is to send your tastyfx credentials to login and this message is a JSON payload.

INSTRUCTIONS	SAMPLE MESSAGES

To send a negotiate message, perform the following actions:
1. Create a **PreTrade/Trade** WebSocket connection or use the existing **PreTrade/Trade** WebSocket session.
2. Change **Timestamp** with the current time in UTC.
Note: **Timestamp** must be the number of milliseconds since the epoch.
3. Change **SessionId** with an appropriate unique identifier value.
Note: **SessionId** acts as a logical identifier and must be unique for each new session. It is recommended to generate UUID (Universal unique identifier) as a **SessionId**.
4. Change **username** with your tastyfx username.
5. Change **password** with your tastyfx password.
Note: The username and password are separated by a colon.
6. You can now send **Negotiate** request to the server.
7. If the negotiate request is successful, then the IG server returns with a **NegotiationResponse** message.
8. If the negotiate request is unsuccessful, then the IG server returns with a **NegotiationReject** message.
	REQUEST MESSAGE
	{
	"MessageType": "Negotiate",
	"Timestamp": 1573727261820000000,
	"SessionId": "621224d9-995a-4a05-bd4e-d40589db71d0",
	"ClientFlow": "Unsequenced",
	"Credentials": {
	"CredentialsType": "login",
	"Token": "username:password"}
	}
	RESPONSE MESSAGE
	{
	"MessageType": "NegotiationResponse"
	"SessionId": "621224d9-995a-4a05-bd4e-d40589db71d0",
	"RequestTimestamp": 1573727261820000000,
	"ServerFlow": "Unsequenced",
	}
	SAMPLE ERROR MESSAGE (BAD CREDENTIALS)
	{
	"MessageType": "NegotiationReject"
	"SessionId": "621224d9-995a-4a05-bd4e-d40589db71d0",
	"Timestamp": 1573727261820000000,
	"Code": "Credentials",
	"Reason": "Bad Credentials",
	}

3.4.1.2 Establish a session

After connecting the WebSocket and successfully receiving the **NegotiationResponse** message from tastyfx, the next important step is to establish a heartbeat frequency. Setting up a heartbeat frequency helps to set the time interval for keeping the session alive. The heartbeat functionality helps to maintain the stability of the session and enables timely detection of any potential issues or network interruptions.

Note:

- The value of **SessionId** must match the value used in the **Negotiate** JSON payload.
- The **KeepaliveInterval** is the duration between heartbeats that you send to the IG server. The purpose of **KeepaliveInterval** is to ensure the session remains active and uninterrupted. It is important to consider internet latency when setting the interval value. To ensure the effectiveness, the interval should be set to a value larger than the periodic timer used in your application.
- You will also receive heartbeats as an tastyfx response. You can see the **Keepaliveinterval** for tastyfx response in **EstablishmentAck** message.

INSTRUCTIONS	SAMPLE MESSAGES
To send an establish message, perform the following actions: 1. Create a **PreTrade/Trade** WebSocket connection or use the existing **PreTrade/Trade** WebSocket connection. 2. Send a **Negotiate** message. 3. Change **Timestamp** with the current time in UTC. **Note**: **Timestamp** must be the number of milliseconds since the epoch. 4. Change **SessionId** with an appropriate unique identifier value. **Note**: **SessionId** acts as a logical identifier and must be unique for each new session. It is recommended to generate UUID (Universal unique identifier) as a **SessionId**. 5. Change **KeepaliveInterval** with the required value for the duration which you want the session to be active. **Note**: The **KeepaliveInterval** is measured in milliseconds. It is recommended to add 65000 (65 seconds) as the **KeepaliveInterval** value. 6. You can now send **Establish** request to the server. 7. If the establish request is successful, then the IG server returns with a **EstablishmentAck** message.	REQUEST MESSAGE **{** **"MessageType": "Establish",** **"Timestamp": 1573727261820000000,** **"SessionId": "621224d9-995a-4a05-bd4e-d40589db71d0",** **"KeepaliveInterval": 600000** **}** RESPONSE MESSAGE **{** **"MessageType": "EstablishmentAck"** **"SessionId": "621224d9-995a-4a05-bd4e-d40589db71d0",** **"RequestTimestamp": 1573727261820000000,** **"KeepaliveInterval": 30001,** **}** SAMPLE ERROR MESSAGE **{** **"MsgType": "BusinessMessageReject",** **"BusinessRejectReason": "Other",** **"Text": "rejecting message (first 200 chars): {\"MessageType\":\"Establish\",\"Timestamp\":1573727261820000000,\"SessionId\":\"621224d9-995a-4a05-bd4e-d40589db71d0\",\"KeepaliveInterval\":6000}",** **"ApplVerID": "FIX50SP2",** **"SendingTime": "2024-02-11T16:44:21.816"** **}**

8. If the establish request is unsuccessful, then the IG server returns with a **BusinessMessageReject** message.	

3.5 Keep session alive

After successfully creating a WebSocket connection, you must regularly send a heartbeat message. Keep session alive functionality ensures that the connection remains active and prevents it from being terminated due to inactivity.

INSTRUCTIONS	SAMPLE MESSAGES
To configure a keep session alive message, perform the following actions: 1. Create a **PreTrade/Trade** WebSocket connection or use the existing **PreTrade/Trade** WebSocket session. 2. You can now send **Keep session alive** request to the server. 3. The IG Server does not respond to the **Keep session alive** message. On receipt of the **Keep session alive** message, the IG server automatically resets the connection timeout timer. **Note**: The IG Server will send the same **Keep session alive** message within the time limit specified in the **Establish** message. This heartbeat allows the client to know that the IG Server is actively connected to the WebSocket.	**REQUEST MESSAGE** **{"MessageType":"UnsequencedHeartbeat"}**

3.6 Pretrade

A pretrade API, short for Pre-Trade Application Programming Interface, provides access to data and functionality related to pre-trade activities in financial markets. It allows you to discover the tastyfx security identifiers for trading in forex pairs and to obtain live quotes.

3.6.1 Request security identifiers from tastyfx

After the WebSocket connection is successfully established, you can now obtain the security identifiers which are required to identify the forex pairs. This allows you to retrieve information such as symbol names, descriptions, pricing details, and other relevant data for the available forex currency pairs.

INSTRUCTIONS	SAMPLE MESSAGES

To request the security identifiers, perform the following actions:
1. Create a **PreTrade** WebSocket connection or use the existing **PreTrade** WebSocket session.
2. Change **SecurityReqID** with an appropriate unique identifier value.
Note: **SecurityReqID** acts as a unique identifier for each new currency pair request.
3. Change **SubscriptionRequestType** to **Snapshot**.
4. Change **SecurityListRequestType** to **AllSecurities**.
5. You can now send **SecurityListRequest** to the server.
6. If the get security list request is successful, then the IG server returns with **SecurityList** message.
Note: In the snippet of all symbol's payload, the **SecListGrp** has been reduced from eighty-two currency pairs to a single currency pair, USD/CAD.
7. If the get security list request is unsuccessful, then the IG server returns with a **BusinessMessageReject** message.
	REQUEST MESSAGE
	{
	"MsgType": "SecurityListRequest",
	"SendingTime": "2023-12-25T15:53:40.996",
	"SecurityReqID": "listReq+1616687620989",
	"SubscriptionRequestType": "Snapshot",
	"ApplVerID": "FIX50SP2",
	"SecAltIDGrp": [],
	"SecurityListRequestType": "AllSecurities"
	}
	RESPONSE MESSAGE
	{
	"SecurityReqID": "listReq+1616687620989",
	"SecurityResponseID": "listReq+1616687620989~1",
	"SecurityRequestResult": "ValidRequest",
	"TotNoRelatedSym": 82,
	"LastFragment": "LastMessage",
	"SecListGrp": [
	{
	"Symbol": "USD/CAD",
	"SecurityID": "CS.D.USDCAD.CZD.IP",
	"SecurityIDSource": "MarketplaceAssignedIdentifier",
	"SecAltIDGrp": [],
	"SecurityGroup": "CURRENCIES",
	"ContractMultiplier": 100000,
	"SecurityDesc": "USD100,000 Contract",
	"ShortSaleRestriction": "NoRestrictions",
	"AttrbGrp": [
	{
	"InstrAttribType": "DealableCurrencies",
	"InstrAttribValue": "CAD"
	}
	],
	"UndInstrmtGrp": [],
	"Currency": "CAD"
	}

	],
	"MsgType": "SecurityList",
	"ApplVerID": "FIX50SP2",
	"SendingTime": "2023-12-25T16:58:48.812"
	}
	
	SAMPLE ERROR MESSAGE
	{
	"MsgType": "BusinessMessageReject",
	"RefMsgType": "SecurityListRequest",
	"BusinessRejectRefID": "listReq+1616687620989",
	"BusinessRejectReason": "Other",
	"Text": "rejecting message (first 200 chars):
	{\"MsgType\":\"SecurityListRequest\",\"SendingTime\":\"2023-
	12-
	25T15:53:40.996\",\"SecurityReqID\":\"listReq+1616687620989\
	",\"SubscriptionRequestType\":\"Snapshot\",\"ApplVerID\":\"F
	IX50SP2\",\"SecAltIDGrp\":[],\"SecurityL",
	"ApplVerID": "FIX50SP2",
	"SendingTime": "2023-12-29T11:37:42.187"
	}

3.6.2 Request quotes

Requesting quotes refers to the action of retrieving current market prices or quotes for the securities.

INSTRUCTIONS	SAMPLE MESSAGES

To request a quote, perform the following actions:
1. Create a **PreTrade** WebSocket connection or use the existing **PreTrade** WebSocket session.
2. Change **QuoteReqID** with an appropriate unique identifier value. **Note**: **QuoteReqID** acts as a unique identifier for each new quote request.
3. Change **SubscriptionRequestType** as **SnapshotAndUpdates**.
4. You can now send **Quote** request to the server. This automatically creates a subscription for quotes updates and the server sends you a snapshot of quotes.
5. If the quote request is successful, then the IG server returns with **Quote** messages.
6. If you want to unsubscribe from **Quote** request, you must change **SubscriptionRequestType** as **DisablePreviousSnapshot**, and then send a request to the server.
7. If you want to cancel or revoke a quote, you need to send a **QuoteCancel** request to the server.
8. If the quote request is unsuccessful, then the IG server returns with a **QuoteRequestReject** message.
	REQUEST MESSAGE
	{
	"MsgType": "QuoteRequest",
	"ApplVerID": "FIX50SP2",
	"CstmApplVerID": "IGUS/PreTrade/V1",
	"SendingTime": "2024-02-20T15:44:52.644",
	"QuoteReqID": "ABz12345L",
	"SubscriptionRequestType": "SnapshotAndUpdates",
	"QuotReqGrp": [
	{
	"Symbol": "GBPUSD",
	"SecurityID": "CS.D.GBPUSD.CZD.IP",
	"SecurityIDSource": "MarketplaceAssignedIdentifier"
	}
	]
	}
	
	RESPONSE MESSAGE
	{
	"MsgType": "Quote",
	"QuoteReqID": "ABz12345L",
	"BidID": "1709018769115431731",
	"OfferID": "1709018769115434732",
	"QuoteType": "Tradeable",
	"SecAltIDGrp": [],
	"BidPx": 1.26942,
	"OfferPx": 1.26952,
	"NetChgPrevDay": 0.00093,
	"ApplVerID": "FIX50SP2",
	"SendingTime": "2024-02-27T07:26:09.413"
	}
	
	DISABLEPREVIOUSSNAPSHOT
	{
	"MsgType": "QuoteRequest",
	"ApplVerID":"FIX50SP2",
	"CstmApplVerID": "IGUS/PreTrade/V1",
	"SendingTime": "2024-02-22T17:44:52.644",
	"QuoteReqID":"ABz12345L",
	"SubscriptionRequestType":"DisablePreviousSnapshot",
	"QuotReqGrp" : [
	{
	"Symbol":"GBPUSD",

```
   "SecurityID":"CS.D.GBPUSD.CZD.IP",
   "SecurityIDSource":"MarketplaceAssignedIdentifier"
   }
  ]
}
```

QUOTE CANCEL

```
{
 "MsgType": "QuoteCancel",
 "ApplVerID":"FIX50SP2",
 "CstmApplVerID": "IGUS/PreTrade/V1",
 "SendingTime": "2024-02-25T16:44:52.937",
 "QuoteReqID":"A0912345",
 "QuoteCancelType":"CancelForOneOrMoreSecurities",
 "NoRelatedSym":[
   {"SecurityID":"CS.D.GBPUSD.CZD.IP",
    "SecurityIDSource":"MarketplaceAssignedIdentifier"}
 ]
}
```

SAMPLE ERROR MESSAGE

```
{
 "MsgType": "QuoteRequestReject",
 "ApplVerID":"FIX50SP2",
 "CstmApplVerID": "IGUS/PreTrade/V1",
 "SendingTime": "2024-02-02T21:14:38.717",
 "QuoteReqID":"Qmw12345",
 "QuoteRequestRejectReason":"UnknownSymbol",
 "QuotReqRjctGrp":[
  {"SecurityID":"CS.D.GBPUSD.CZD.IP",
   "SecurityIDSource":"MarketplaceAssignedIdentifier"}
 ]
}
```

3.7 Trade

3.7.1 Placing an order

Placing an order is the process of submitting a request to execute the trade. It involves sending the necessary information and instructions to the trading platform indicating the desired trade parameters such as the security to be traded, order type, quantity, price, and any additional specifications. To place a single order for a specific security, you can use **NewOrderSingle** message.

Note:

- Prior to placing an order, it is important to pay careful attention to the lot size. The standard size of one lot is typically set as 100,000 units. Kindly ensure that you consider this lot size when determining the quantity for your order. The unit used for quantity on the API is number of lots and not a notional forex amount.
- If you want to see more information about the lot size, you can check the multiplier by looking at the **ContractMultiplier** field from the **SecurityList**.

INSTRUCTIONS	SAMPLE MESSAGES
To place an order, perform the following actions: 1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session. 2. Change **ClOrdID** with an appropriate unique identifier value. **Note**: **ClOrdID** acts as a unique identifier for each new order. You can use **ClOrdID** to correlate execution reports and their corresponding order requests. 3. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 4. You can now send **NewOrderSingle** to the server. 5. If the new order request is successful, then the IG server returns with an **ExecutionReport** message. **Note**: Kindly note that **LastPx**, **AvgPX**, and **Price** are all with respect to order currency, which is USD. 6. If the new order request is unsuccessful, then the IG server returns an **ExecutionReport** message with **OrdRejReason**.	**REQUEST MESSAGE** **{** **"MsgType": "NewOrderSingle",** **"ApplVerID": "FIX50SP2",** **"CstmApplVerID": "IGUS/Trade/V1",** **"SendingTime": "2024-02-02T21:14:38.717",** **"ClOrdID": "12345XYZ=12",** **"Account": "XXXXX",** **"SecurityID": "CS.D.GBPUSD.CZD.IP",** **"SecurityIDSource": "MarketplaceAssignedIdentifier",** **"Side": "Buy",** **"TransactTime": "2024-02-02T21:14:38.717",** **"OrderQty": "6",** **"OrdType": "Limit",** **"Price": "1.15",** **"Currency": "USD",** **"TimeInForce": "GoodTillDate",** **"ExpireTime": "2019-08-02T17:00:00.000"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "ExecutionReport",** **"ApplVerID": "FIX50SP2",** **"CstmApplVerID": "IGUS/Trade/V1",** **"SendingTime": "2024-02-02T21:14:40.001",** **"OrderID": "XXXXXXXXXXXXXXX",** **"ClOrdID": "12345XYZ=12",** **"ExecID": "0997234657176",**

```
  "ExecType": "Trade",
  "OrdStatus": "Filled",
  "Account": "XXXXX",
  "SecurityID": "CS.D.GBPUSD.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "Side": "Buy",
  "OrderQty": "6",
  "OrdType": "Limit",
  "Price": "1.15",
  "TimeInForce": "GoodTillDate",
  "ExpireTime": "2019-08-02T17:00:00.000",
  "Currency": "USD",
  "LastQty": "6",
  "LastPx": "1.15",
  "LeavesQty": "0",
  "CumQty": "6",
  "AvgPx": "1.15",
  "TransactTime": "2024-02-02T21:14:38.717"
}
```

Sample error message

```
{
  "MsgType": "ExecutionReport",
  "OrderID": "ORAAAAPFVZYY5A4",
  "ClOrdID": "12345XYZ=1234",
  "ExecID": "EXAAAAPFVZYY6A4",
  "ExecType": "Rejected",
  "OrdStatus": "Rejected",
  "WorkingIndicator": "NotWorking",
  "OrdRejReason": "Other",
  "Account": "Z33UVI",
  "SecurityID": "CS.D.GBPUSD.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "SecAltIDGrp": [],
  "Side": "Buy",
  "OrderQty": 6,
  "OrdType": "Limit",
  "Price": 0.05,
  "Currency": "USD",
  "TimeInForce": "FillOrKill",
  "OrderAttributeGrp": [],
  "LastQty": 0,
  "LastPx": 0,
  "LeavesQty": 6,
  "CumQty": 0,
  "AvgPx": 0,
  "TransactTime": "2024-02-22T09:04:17.021+0000",
  "Text": "Price has moved beyond your limit order price",
```

	"ApplVerID": "FIX50SP2", "SendingTime": "2024-02-22T09:04:17.042" }

3.7.2 Attaching a stop order

A stop order is an instruction to exit a position if the market moves in opposite direction of your trade and the price hits a certain predetermined level.

Note:

- Please note that you can only place one single stop order for an individual security at any given time.
- If a stop order is added, it will automatically adjust its size in correlation with the aggregate position held for the forex trade. This automatic adjustment ensures that the stop order scales proportionally with the position size change.
- You must ensure that the stop order needs to be placed in the opposite direction of your trade. For example, if you have bought (Buy) a security pair (GBPUSD) and wants to place a stop order, then you need to place the stop order as Sell.
- You need to place the stop loss order as an attached order (**OrderAttributeType=AttachedOrder**).

INSTRUCTIONS	SAMPLE MESSAGES

To attach a stop order, perform the following actions:
1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session.
2. Change **Account** with the account identifier.
Note: Account identifier value is a different value from your demo or production username and password.
3. Change **ClOrdID** with an appropriate unique identifier value.
Note: **ClOrdID** acts as a unique identifier for each new stop loss order. You can use **ClOrdID** to correlate execution reports and their corresponding order requests.
4. You can now send **NewOrderSingle** to the server.
5. If the stop order request is successful, then the IG server returns with an **ExecutionReport** message.
6. If the stop order request is unsuccessful, then the IG server returns an **ExecutionReport** message with **OrdRejReason**.
	REQUEST MESSAGE
	{
	"MsgType": "NewOrderSingle",
	"Account": "XXXXX",
	"OrderQty": "1",
	"OrdType": "Stop",
	"ApplVerID": "FIX50SP2",
	"ClOrdID": "StopLossOrder1+1697547791671",
	"SecurityIDSource": "MarketplaceAssignedIdentifier",
	"TransactTime": "2024-02-17T13:03:11.000+0000",
	"Side": "Sell",
	"StopPx": "0.82869",
	"SendingTime": "2024-02-17T14:03:11.672773",
	"OrderAttributeGrp": [
	{
	"OrderAttributeType": "AttachedOrder",
	"OrderAttributeValue": "Y"
	}
	],
	"Currency": "GBP",
	"TimeInForce": "GoodTillCancel",
	"SecurityID": "CS.D.EURGBP.CZD.IP"
	}
	RESPONSE MESSAGE
	{
	"MsgType": "ExecutionReport",
	"OrderID": "XXXXXXXXXXXXXX",

```
  "ClOrdID": "StopLossOrder1+1697547791671",
  "ExecID": "EXAAAAMHFRXSBA2",
  "ExecType": "New",
  "OrdStatus": "New",
  "WorkingIndicator": "Working",
  "Account": "XXXXX",
  "SecurityID": "CS.D.EURGBP.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "SecAltIDGrp": [],
  "Side": "Sell",
  "OrderQty": 1,
  "OrdType": "Stop",
  "StopPx": 0.82869,
  "Currency": "GBP",
  "TimeInForce": "GoodTillCancel",
  "OrderAttributeGrp": [],
  "LastQty": 0,
  "LastPx": 0,
  "LeavesQty": 1,
  "CumQty": 0,
  "AvgPx": 0,
  "TransactTime": "2024-02-17T13:03:11.737+0000",
  "ApplVerID": "FIX50SP2",
  "SendingTime": "2024-02-17T13:03:11.756"
}
```

SAMPLE ERROR MESSAGE

```
{
  "OrderID": "",
  "ClOrdID": "OHFHLH703987211",
  "ExecID": "generated_c64eeb6a-aa46-4664-a679-
4005ecae58a0",
  "ExecType": "Rejected",
  "OrdStatus": "Rejected",
  "WorkingIndicator": "NotWorking",
  "OrdRejReason": "Other",
  "Account": "Z33UVI",
  "SecurityID": "CS.D.GBPUSD.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "SecAltIDGrp": [],
  "Side": "Buy",
  "OrderQty": 1,
  "Price": 13000,
  "Currency": "GBP",
  "OrderAttributeGrp": [],
  "LastQty": 0,
  "LastPx": 0,
  "LeavesQty": 1,
```

	"CumQty": 0, "AvgPx": 0, "TransactTime": "2024-02-01T06:58:52.348+0000", "Text": "ORDER NOT FOUND", "MsgType": "ExecutionReport", "ApplVerID": "FIX50SP2", "SendingTime": "2024-02-01T06:58:52.361" }

3.7.3 Attaching a take profit order

A take profit order is an instruction to exit a trade if the market moves in your direction of trade and the price hits a certain predetermined level.

Note:

- Please note that you can only place one single **Take profit** order for an individual security at any given time.
- You need to place the take profit order as an attached order (**OrderAttributeType=AttachedOrder**).

INSTRUCTIONS	SAMPLE MESSAGES
To attach a take profit order, perform the following actions: 1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session. 2. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 3. Change **ClOrdID** with an appropriate unique identifier value. **Note**: **ClOrdID** acts as a unique identifier for each new take profit order. You can use **ClOrdID** to correlate execution reports and their corresponding order requests. 4. You can now send **NewOrderSingle** to the server. 5. If the take profit request is successful, then the IG server returns with an **ExecutionReport** message. 6. If the take profit order request is unsuccessful, then the IG server returns an **ExecutionReport** message with **OrdRejReason**.	**REQUEST MESSAGE** **{** **"MsgType": "NewOrderSingle",** **"Account": "XXXXX",** **"OrderQty": "1",** **"OrdType": "Limit",** **"ApplVerID": "FIX50SP2",** **"ClOrdID": "TakeProfitClientOrder+1697547789578",** **"SecurityIDSource": "MarketplaceAssignedIdentifier",** **"TransactTime": "2024-02-17T13:03:09.000+0000",** **"Side": "Sell",** **"SendingTime": "2024-02-17T14:03:09.579846",** **"Price": "0.8787",** **"OrderAttributeGrp": [** **{** **"OrderAttributeType": "AttachedOrder",** **"OrderAttributeValue": "Y"** **}** **],** **"Currency": "GBP",** **"TimeInForce": "GoodTillCancel",** **"SecurityID": "CS.D.EURGBP.CZD.IP"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "ExecutionReport",** **"OrderID": "XXXXXXXXXXXXXXX",** **"ClOrdID": "TakeProfitClientOrder+1697547789578",** **"ExecID": "EXAAAAMHFRXRHA2",** **"ExecType": "New",** **"OrdStatus": "New",** **"WorkingIndicator": "Working",** **"Account": "XXXXX",**

	"SecurityID": "CS.D.EURGBP.CZD.IP",
	"SecurityIDSource": "MarketplaceAssignedIdentifier",
	"SecAltIDGrp": [],
	"Side": "Sell",
	"OrderQty": 1,
	"OrdType": "Limit",
	"Price": 0.8787,
	"Currency": "GBP",
	"TimeInForce": "GoodTillCancel",
	"OrderAttributeGrp": [],
	"LastQty": 0,
	"LastPx": 0,
	"LeavesQty": 1,
	"CumQty": 0,
	"AvgPx": 0,
	"TransactTime": "2024-02-17T13:03:09.660+0000",
	"ApplVerID": "FIX50SP2",
	"SendingTime": "2024-02-17T13:03:09.678"
	}
	SAMPLE ERROR MESSAGE
	{
	"OrderID": "",
	"ClOrdID": "TakeProfitClientOrder+16975477895781",
	"ExecID": "generated_d33b1f16-6334-409a-aa3c-02a5b934afc8",
	"ExecType": "Rejected",
	"OrdStatus": "Rejected",
	"WorkingIndicator": "NotWorking",
	"OrdRejReason": "Other",
	"Account": "Z33UVI",
	"SecurityID": "CS.D.GBPUSD.CZD.IP",
	"SecurityIDSource": "MarketplaceAssignedIdentifier",
	"SecAltIDGrp": [],
	"Side": "Sell",
	"OrderQty": 0.1,
	"Price": 8000,
	"Currency": "USD",
	"OrderAttributeGrp": [],
	"LastQty": 0,
	"LastPx": 0,
	"LeavesQty": 0.1,
	"CumQty": 0,
	"AvgPx": 0,
	"TransactTime": "2024-02-20T02:19:51.142+0000",
	"Text": "ORDER NOT FOUND",
	"MsgType": "ExecutionReport",
	"ApplVerID": "FIX50SP2",

	"SendingTime": "2024-02-20T02:19:51.156" }

3.7.4 Replacing an order

To replace an order, you can use the **OrderCancelReplaceRequest** message. In addition to replacing working orders, you can also replace a stop or take profit orders.

INSTRUCTIONS	SAMPLE MESSAGES
To replace an order, perform the following actions: 1. Create a **Trade** WebSocket connection or use the existing **Trade WebSocket** session. 2. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 3. Change **ClOrdID** with an appropriate unique identifier value. **Note**: **ClOrdID** acts as a unique identifier for each new replace order. The **ClOrdID** used for the replace request must be different and unique from the **ClOrdID** of the order you are attempting to replace. 4. Change **OrderID** with the order ID for which you want to replace the order. **Note**: Alternatively, you can enter **OrigClOrdID** for replacing an order. You must enter the **OrigClOrdID** of the most recent modified order. To successfully replace the same order multiple times, it is essential to provide a different **OrigClOrdID** for each replacement request. 5. You can now send **OrderCancelReplaceRequest** to the server. 6. If the order replace request is successful, then the IG server returns with an **ExecutionReport** message. 7. If the order replace request is unsuccessful, then the IG server returns with an **OrderCancelReject** message.	**REQUEST MESSAGE** **{** **"MsgType":"OrderCancelReplaceRequest",** **"Account":"XXXXX",** **"OrderQty":"1",** **"OrdType":"Limit",** **"ApplVerID":"FIX50SP2",** **"ClOrdID":"restingClientOrderReplaceOrder+1697547766011",** **"SecurityIDSource":"MarketplaceAssignedIdentifier",** **"OrderID":"XXXXXXXXXXXXXXX",** **"TransactTime":"2024-02-17T13:02:46.000+0000",** **"Side":"Buy",** **"SendingTime":"2024-02-17T14:02:46.012489",** **"Price":"0.84875",** **"Currency":"GBP",** **"TimeInForce":"GoodTillCancel",** **"SecurityID":"CS.D.EURGBP.CZD.IP"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "ExecutionReport",** **"OrderID": "XXXXXXXXXXXXXXX",** **"ClOrdID":** **"restingClientOrderReplaceOrder+1697547766011",** **"OrigClOrdID": "restingClientOrder+1697547765386",** **"ExecID": "EXAAAAMHFQ36ZA2",** **"ExecType": "Replaced",** **"OrdStatus": "New",** **"WorkingIndicator": "Working",** **"Account": "XXXXX",** **"SecurityID": "CS.D.EURGBP.CZD.IP",** **"SecurityIDSource": "MarketplaceAssignedIdentifier",** **"SecAltIDGrp": [],** **"Side": "Buy",** **"OrderQty": 1,** **"OrdType": "Limit",** **"Price": 0.84875,** **"Currency": "GBP",** **"TimeInForce": "GoodTillCancel",** **"OrderAttributeGrp": [],** **"LastQty": 0,**

```
  "LastPx": 0,
  "LeavesQty": 1,
  "CumQty": 0,
  "AvgPx": 0,
  "TransactTime": "2024-02-17T13:02:46.049+0000",
  "ApplVerID": "FIX50SP2",
  "SendingTime": "2024-02-17T13:02:46.066"
}
```

SAMPLE ERROR MESSAGE

```
{
  "MsgType": "ExecutionReport",
  "OrderID": "OPAAAAN5DA5CKAV",
  "ClOrdID":
"restingClientOrderReplaceOrder+1697547766011",
  "ExecID": "generated_d6db9568-e4e1-4e6a-82c5-
f206b6b7bc9c",
  "ExecType": "Rejected",
  "OrdStatus": "Rejected",
  "WorkingIndicator": "NotWorking",
  "OrdRejReason": "Other",
  "Account": "Z33UVI",
  "SecurityID": "CS.D.GBPUSD.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "SecAltIDGrp": [],
  "Side": "Buy",
  "OrderQty": 0,
  "Price": 0.84875,
  "Currency": "",
  "TimeInForce": "GoodTillCancel",
  "OrderAttributeGrp": [],
  "LastQty": 0,
  "LastPx": 0,
  "LeavesQty": 0,
  "CumQty": 0,
  "AvgPx": 0,
  "TransactTime": "2024-02-20T02:24:55.423+0000",
  "Text": "ORDER NOT FOUND",
  "ApplVerID": "FIX50SP2",
  "SendingTime": "2024-02-20T02:24:55.432"
}
```

3.7.5 Request working orders

The request working orders allows clients to obtain information about their active orders which are not filled yet. These are orders that have been placed but are not filled yet, waiting for the specified limit price to be triggered.

3.7.5.1 Request a specific working order

To obtain an execution report for a specific working order, you need to send an **OrderStatusRequest** message to the server. If the order is still active, you will receive the parameters and details of the requested working order.

INSTRUCTIONS	SAMPLE MESSAGES
To request a specific working order, perform the following actions: 1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session. 2. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 3. Change **OrderID** with the order ID for which you want to view the working order details. 4. Change **OrdStatusReqID** with an appropriate unique identifier value. **Note**: **OrdStatusReqID** acts as a unique identifier for each new working order request. 5. You can now send **OrderStatusRequest** to the server. 6. If the order status request is successful, then the IG server returns with an **ExecutionReport** message. 7. If the order status request is unsuccessful, then the IG server returns an **ExecutionReport** message with **OrdRejReason**.	**REQUEST MESSAGE** { "MsgType": "OrderStatusRequest", "Account": "XXXXX", "OrderID": "XXXXXXXXXXXXXXX", "SendingTime": "2024-02-17T14:03:35.198078", "OrdStatusReqID": "orderStatusReq+1697547815197", "ApplVerID": "FIX50SP2" } **RESPONSE MESSAGE** { "MsgType": "ExecutionReport", "OrderID": "XXXXXXXXXXXXXXX", "ClOrdID": "LimitGTCOrder1+1697547814992", "OrdStatusReqID": "orderStatusReq+1697547815197", "ExecID": "0", "ExecType": "OrderStatus", "OrdStatus": "New", "WorkingIndicator": "Working", "Account": "XXXXX", "SecurityID": "CS.D.EURGBP.CZD.IP", "SecurityIDSource": "MarketplaceAssignedIdentifier", "SecAltIDGrp": [], "Side": "Buy", "OrderQty": 1, "OrdType": "Limit", "Price": 0.85865, "Currency": "GBP",

```
 "TimeInForce": "GoodTillCancel",
 "OrderAttributeGrp": [],
 "LastQty": 0,
 "LastPx": 0,
 "LeavesQty": 1,
 "CumQty": 0,
 "AvgPx": 0,
 "TransactTime": "2024-02-17T13:03:35.024+0000",
 "MsgType": "ExecutionReport",
 "ApplVerID": "FIX50SP2",
 "SendingTime": "2024-02-17T13:03:35.236"
}
```

Sample error message

```
{
 "OrderID": "NONE",
 "OrdStatusReqID": "orderStatusReq+1697547815197",
 "ExecID": "NONE",
 "ExecType": "OrderStatus",
 "OrdStatus": "Rejected",
 "OrdRejReason": "UnknownOrder",
 "Account": "ORAAAAN6FNE9YAV",
 "SecAltIDGrp": [],
 "OrdType": "Limit",
 "OrderAttributeGrp": [],
 "Text": "NONE FOUND",
 "MsgType": "ExecutionReport",
 "ApplVerID": "FIX50SP2",
 "SendingTime": "2024-02-01T07:02:21.646"
}
```

3.7.5.2 Request all working orders

Request all working orders refers to the functionality that enables you to retrieve and review all your current working orders through a single API request. To request for viewing an account's current open orders, you need to send an **OrderMassStatusRequest** message to the server.

Note:

- For each working order, you will receive multiple execution reports with each report corresponding to a specific order.
- To confirm if the latest report received is the last one among multiple reports, refer to the field named **LastRptRequested**. If the **LastRptRequested** field displays **LastMessage**, it means that the most recent report you received is the final report in the series.

INSTRUCTIONS	SAMPLE MESSAGES
To request the mass working order, perform the following actions: 1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session. 2. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 3. Change **MassStatusReqID** with an appropriate unique identifier value. **Note**: **MassStatusReqID** acts as a unique identifier for each new working orders request. 4. You can now send **OrderMassStatusRequest** to the server. 5. If the order mass status request is successful, then the IG server returns with an **ExecutionReport** message for each active order. 6. If the order mass status request is unsuccessful, then the IG server returns an **ExecutionReport** message with **OrdRejReason**.	**REQUEST MESSAGE** **{** **"MsgType": "OrderMassStatusRequest",** **"Account": "XXXXX",** **"MassStatusReqID": "MassStatusReqID2+1697547750722",** **"MassStatusReqType": "StatusForOrdersForAPartyID",** **"SendingTime": "2024-02-17T14:02:30.723713",** **"ApplVerID": "FIX50SP2"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "ExecutionReport",** **"OrderID": "XXXXXXXXXXXXXXX",** **"ClOrdID": "LimitGTCOrder1+1695908676988",** **"MassStatusReqID": "MassStatusReqID2+1697547750722",** **"LastRptRequested": "NotLastMessage",** **"ExecID": "0",** **"ExecType": "OrderStatus",** **"OrdStatus": "New",** **"WorkingIndicator": "Working",** **"Account": "XXXXX",** **"SecurityID": "CS.D.EURGBP.CZD.IP",** **"SecurityIDSource": "MarketplaceAssignedIdentifier",** **"SecAltIDGrp": [],** **"Side": "Buy",** **"OrderQty": 1,** **"OrdType": "Limit",** **"Price": 0.85491,** **"Currency": "GBP",** **"TimeInForce": "GoodTillCancel",**

```
 "OrderAttributeGrp": [],
 "LastQty": 0,
 "LastPx": 0,
 "LeavesQty": 1,
 "CumQty": 0,
 "AvgPx": 0,
 "TransactTime": "2024-02-28T13:44:37.000+0000",
 "ApplVerID": "FIX50SP2",
 "SendingTime": "2024-02-17T13:02:30.761"
}
```

SAMPLE ERROR MESSAGE

```
{
 "OrderID": "NONE",
 "MassStatusReqID": "MassStatusReqID2+1697547750722",
 "ExecID": "d844a8f8-31f3-4409-94fa-015ced442c7f",
 "ExecType": "OrderStatus",
 "OrdStatus": "Rejected",
 "OrdRejReason": "UnknownOrder",
 "Account": "Z33UVI",
 "SecAltIDGrp": [],
 "OrdType": "Limit",
 "OrderAttributeGrp": [],
 "TransactTime": "2024-02-01T07:41:15.005+0000",
 "Text": "NONE FOUND",
 "MsgType": "ExecutionReport",
 "ApplVerID": "FIX50SP2",
 "SendingTime": "2024-03-01T07:41:15.005"
}
```

3.7.6 Cancel an order

Cancel an order refers to the action of requesting the cancellation of a previously placed order before it reaches the terminal state. To cancel an order, you can use the **OrderCancelRequest** message.

INSTRUCTIONS

To cancel an order, perform the following actions:

1. Create a **Trade** WebSocket connection or use the existing **Trade** WebSocket session.
2. Change **Account** with the account identifier.
   Note: Account identifier value is a different value from your demo or production username and password.
3. Change **ClOrdID** with an appropriate unique identifier value.
   Note: **ClOrdID** acts as a unique identifier for each new cancellation request. The ClOrdID used for the cancellation request must be different and unique from the ClOrdID of the order you are attempting to cancel.
4. Change **OrderID** with the order ID for which you want to cancel the order.
   Note: Alternatively, you can also enter **OrigClOrdID** for cancelling an order. You must enter the **OrigClOrdID** of the most recent modified order.
5. You can now send **OrderCancelRequest** to the server.
6. If the order cancel is successful, then the IG server returns an **ExecutionReport** message with **OrdStatus** field as **Canceled**.
7. If the order cancel is unsuccessful, then the IG server returns with an **OrderCancelReject** message.

SAMPLE MESSAGES

REQUEST MESSAGE

```
{
 "MsgType": "OrderCancelRequest",
 "Side": "Buy",
 "Account": "XXXXX",
 "OrderQty": "1",
 "SendingTime": "2024-02-04T16:59:53.026",
 "OrdType": "Limit",
 "ApplVerID": "FIX50SP2",
 "ClOrdID": "orderCancelReq1+1659628793026",
 "SecurityID": "CS.D.GBPUSD.CZD.IP",
 "SecurityIDSource": "MarketplaceAssignedIdentifier",
 "OrderID": "XXXXXXXXXXXXXXX",
 "TransactTime": "2024-02-04T15:59:53.000"
}
```

RESPONSE MESSAGE

```
{
 "MsgType": "ExecutionReport",
 "OrderID": "XXXXXXXXXXXXXXX",
 "ClOrdID": "orderCancelReq1+1659628793026",
 "OrigClOrdID": "LimitGTCOrder1+1659628792803",
 "ExecID": "EXAAAAKGQFL2YAG",
 "ExecType": "Canceled",
 "OrdStatus": "Canceled",
 "WorkingIndicator": "NotWorking",
 "Account": "XXXXX",
 "SecurityID": "CS.D.GBPUSD.CZD.IP",
 "SecurityIDSource": "MarketplaceAssignedIdentifier",
 "SecAltIDGrp":[],
 "Side": "Buy",
 "OrderQty":1,
 "OrdType": "Limit",
 "Price":1.20409,
 "Currency": "USD",
 "TimeInForce": "GoodTillCancel",
 "OrderAttributeGrp":[],
 "LastQty":0,
 "LastPx":0,
 "LeavesQty":1,
 "CumQty":0,
 "AvgPx":0,
  "TransactTime": "2024-02-04T15:59:53.045",
  "ApplVerID": "FIX50SP2",
  "SendingTime": "2024-02-04T15:59:53.056"
}
```

SAMPLE ERROR MESSAGE

```
{
  "OrderID": "ORAAAAN5MXLMDAV",
  "ClOrdID": "orderCancelReq1+123",
  "ExecID": "generated_5f4698fe-690c-44a6-a812-
63fa32d8577a",
  "ExecType": "Rejected",
  "OrdStatus": "Rejected",
  "WorkingIndicator": "NotWorking",
  "OrdRejReason": "Other",
  "Account": "Z33UVI",
  "SecurityID": "CS.D.GBPUSD.CZD.IP",
  "SecurityIDSource": "MarketplaceAssignedIdentifier",
  "SecAltIDGrp": [],
  "Side": "Buy",
  "OrderQty": 6,
  "Price": 0,
  "Currency": "",
  "OrderAttributeGrp": [],
  "LastQty": 0,
  "LastPx": 0,
  "LeavesQty": 6,
  "CumQty": 0,
  "AvgPx": 0,
  "TransactTime": "2024-02-01T07:08:13.518+0000",
  "Text": "ORDER NOT FOUND",
  "MsgType": "ExecutionReport",
  "ApplVerID": "FIX50SP2",
  "SendingTime": "2024-02-01T07:08:13.527"
}
```

3.8 PostTrade

3.8.1 Request open positions

Request open positions refers to the action of retrieving information about the current open positions in your trading account. You can request to view the list of open positions and subscribe for receiving position updates. To request a list of open positions and subscribe to position updates, send a **RequestForPositions** message.

Note: If you want to obtain information about your profit and loss, you can refer to the **PositionAmountData** field. The **PositionAmountData** field is only visible when there are partial or full closures of positions. If there are no closed positions, the **PositionAmountData** field will not be visible.

INSTRUCTIONS	SAMPLE MESSAGES
To request the open positions, perform the following actions: 1. Create a **PostTrade** WebSocket connection or use the existing **PostTrade** WebSocket session. 2. Change **Account** with the account identifier. **Note**: Account identifier value is a different value from your demo or production username and password. 3. Change **PosReqID** with an appropriate unique identifier value. **Note**: **PosReqID** acts as a unique identifier for each position request. 4. Change **SubscriptionRequestType** as **SnapshotAndUpdates**. 5. Change **PosReqType** as **Positions**. 6. Change **ClearingBusinessDate** to the current date and time of the request. 7. Change **TransactTime** to the current date and time of the request. 8. You can now send **RequestForPositions** to the server. This automatically creates a subscription for position updates and the server sends you a snapshot of current open positions. 9. If the request for positions order request is successful, then the IG server responds with a **RequestForPositionsAck** message. 10. You can see a separate **PositionReport** messages for each position. **PositionReport** messages are specifically designed to provide detailed information about individual positions.	**REQUEST MESSAGE** **{** **"MsgType": "RequestForPositions",** **"Account": "XXXXX",** **"SendingTime": "2024-02-04T16:59:16.750",** **"PosReqID": "PosReqID+123",** **"SubscriptionRequestType": "SnapshotAndUpdates",** **"ApplVerID": "FIX50SP2",** **"ClearingBusinessDate": "2024-02-04T15:59:16.000",** **"PosReqType": "Positions",** **"TransactTime": "2024-02-04T15:59:16.000"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "RequestForPositionsAck",** **"PosMaintRptID": "b262a48c-df39-4208-a997-ff6555fc1c7d",** **"PosReqID": "PosReqID+1659628756744",** **"TotalNumPosReports": 14,** **"PosReqResult": "ValidRequest",** **"PosReqStatus": "Completed",** **"Account": "XXXXX",** **"ApplVerID": "FIX50SP2",** **"SendingTime": "2024-02-04T15:59:16.784"** **}** **POSITIONREPORT** **{** **"MsgType": "PositionReport",** **"PosMaintRptID": "OPAAAAKGCH6TYAG",** **"PositionID": "OPAAAAKGCH6TYAG",** **"PosReqID": "PosReqID+1659628756744",** **"PosReqType": "Positions",** **"TotalNumPosReports": 1,** **"LastRptRequested": "NotLastMessage",**

11. If you want to unsubscribe from request positions, you must change **SubscriptionRequestType** as **DisablePreviousSnapshot**, and then send a **RequestForPositions** request to the server.
12. If the request for positions order is unsuccessful, then the IG server responds with a **RequestForPositionsAck** message.

```
 "PosReqResult": "ValidRequest",
 "UnsolicitedIndicator":
"MessageIsBeingSentAsAResultOfAPriorRequest",
 "ClearingBusinessDate": "2024-02-03",
 "Account": "XXXXX",
 "SecurityID": "CS.D.GBPUSD.CZD.IP",
 "SecurityIDSource": "MarketplaceAssignedIdentifier",
 "SecAltIDGrp": [],
 "Currency": "USD",
 "PositionQty": [
  {
   "PosType": "TotalTransactionQty",
   "LongQty": 1,
   "PosQtyStatus": "Accepted"
  }
 ],
 "PositionAmountData": [],
 "OpenPrice": 1.21415,
 "ApplVerID": "FIX50SP2",
 "TransactTime": "2024-02-04T15:59:33.780"
 "SendingTime": "2024-02-04T15:59:16.785"
}
```

DISABLEPREVIOUSSNAPSHOT

```
{
"MsgType": "RequestForPositions",
"Account": "XXXXX",
"SendingTime": "2024-02-04T16:59:17",
"PosReqID": "PosReqID+1659628756744",
"SubscriptionRequestType": "DisablePreviousSnapshot",
"ApplVerID": "FIX50SP2",
"ClearingBusinessDate": "2024-02-04T15:59:17.000",
"PosReqType": "Positions",
"TransactTime": "2024-02-04T15:59:17.000"
}
```

SAMPLE ERROR MESSAGE

```
{
 "PosMaintRptID": "62ba6f72-f118-4e90-b048-
87d1f5aa4e66",
 "PosReqID": "PosReqID+123",
 "PosReqResult": "InvalidOrUnsupportedRequest",
 "PosReqStatus": "Rejected",
 "Account": "Z33UVI",
 "Text": "InvalidRequestId",
 "MsgType": "RequestForPositionsAck",
 "ApplVerID": "FIX50SP2",
 "SendingTime": "2024-03-01T07:33:44.525"
```

	}

3.9 ChartData WebSocket

3.9.1 Chart candle data

The purpose of chart candle data is to visually represent the price movement of a security over a specific period. The candle chart consists of individual "candles" that represent the price action of an security for a specific time period, such as minutes, hours, days, or months.

Note: The maximum number of chart candle data points that can be requested is 10000.

Here is a table that shows the maximum available history with respect to each interval:

Interval	Maximum available history	Maximum time period since current time
1sec	2 days	166 minutes
1min	42 days	166 hours
5min	365 days	833 hours
15min	365 days	2500 hours
1hour	365 days	10000 hours
1day	Unlimited	240000 hours
1week	Unlimited	1680000 hours
1month	Unlimited	7300000 hours

3.9.1.1 Streaming chart candle data

Streaming chart candle data refers to the continuous and real-time update of candle chart information as new price data becomes available. Streaming candle data allows traders and investors to view dynamically updated candlesticks.

INSTRUCTIONS	SAMPLE MESSAGES

To request the streaming chart data, perform the following actions:
1. Create a **ChartData** WebSocket connection or use the existing **ChartData** WebSocket session. Note: The URL and connection process is same as of **PreTrade** WebSocket connection.
2. Change **ReqID** with an appropriate unique identifier value. **Note**: **ReqID** acts as a unique identifier for each new streaming chart data request.
3. Change **Interval** with the required duration for which you want to view the streaming chart data.
4. You can now send **ChartDataSubscriptionRequest** to the server.
5. If the request for chart data is successful, then the IG server returns with a
	REQUEST MESSAGE
	{
	"MsgType": "ChartDataSubscriptionRequest",
	"ApplVerID": "FIX50SP2",
	"CstmApplVerID": "IGUS/ChartData/V1",
	"SendingTime": "2024-02-02T17:26:57.042",
	"ReqID": "2",
	"SubscriptionRequestType": "SnapshotAndUpdates",
	"SecurityID": "CS.D.GBPUSD.CZD.IP",
	"SecurityIDSource": "MarketplaceAssignedIdentifier",
	"Interval": "FIVE_MIN"
	}
	RESPONSE MESSAGE
	{
	"MsgType": "ChartDataSubscriptionResponse",
	"ReqID": "2",
	"Interval": "FIVE_MIN",
	"CandleData": {
	"StartDate": "2024-02-20T07:00:00.000+00:00",
	"First": {

ChartDataSubscriptionResponse message.
6. If the request for chart data is unsuccessful, then the IG server returns with a **ChartDataRequestReject** message.
	`    "Bid": 1.25861,`
	`    "Offer": 1.25877`
	`   },`
	`   "Last": {`
	`    "Bid": 1.25911,`
	`    "Offer": 1.25927`
	`   },`
	`   "High": {`
	`    "Bid": 1.25911,`
	`    "Offer": 1.25927`
	`   },`
	`   "Low": {`
	`    "Bid": 1.25861,`
	`    "Offer": 1.25877`
	`   }`
	`  },`
	`  "ApplVerID": "FIX50SP2",`
	`  "SendingTime": "2024-02-20T07:01:33.039"`
	`}`
	
	SAMPLE ERROR MESSAGE
	
	`{`
	`  "MsgType": "ChartDataRequestReject",`
	`  "ReqID": "2",`
	`  "SecurityID": "CSGBPUSD.CZD.IP",`
	`  "ChartRequestRejectReason": "UnknownSymbol",`
	`  "ApplVerID": "FIX50SP2",`
	`  "SendingTime": "2024-01-05T06:04:39.894"`
	`}`

3.9.1.2 Historic chart candle data

Historic chart data candles refer to the individual candlesticks that represent price and market data for a specific time period in the past. These candles are part of a historical chart and provide a visual representation of price movements & patterns over a chosen timeframe.

INSTRUCTIONS	SAMPLE MESSAGES
To send a request to view the historic chart data, perform the following actions: 1. Create a **PreTrade** WebSocket connection or use the existing **PreTrade** WebSocket session. 2. Change **ReqID** with an appropriate unique identifier value. 3. Change **Interval** with the required duration for which you want to view the historic chart data. 4. Change **ReqID** with an appropriate unique identifier value. **Note**: **ReqID** acts as a unique identifier for each new historic chart data request. 5. Change **StartDate** and **EndDate** with the start and end date of the timeframe for which you want to see the historic chart data. Note: You must enter **StartDate** and **EndDate** in accordance with UTC timezone. 6. You can now send **HistoricCandleRequest** to the server. 7. If the request for historic chart data is successful, then the IG server returns with a **HistoricCandleResponse** message. 8. If the request for historic chart data is unsuccessful, then the IG server returns with a **BusinessMessageReject** message.	**REQUEST MESSAGE** **{** **"MsgType": "HistoricCandleRequest",** **"ApplVerID": "FIX50SP2",** **"SendingTime": "2024-01-02T17:09:16.602",** **"CstmApplVerID": "IGUS/ChartData/V1",** **"ReqID": "24572562",** **"SecurityID": "CS.D.GBPUSD.CZD.IP",** **"SecurityIDSource": "MarketplaceAssignedIdentifier",** **"Interval": "FIVE_MIN",** **"StartDate": "2024-01-02T09:00:00.000",** **"EndDate": "2024-01-02T12:00:00.000"** **}** **RESPONSE MESSAGE** **{** **"MsgType": "ChartDataSubscriptionResponse",** **"ReqID": "2",** **"Interval": "FIVE_MIN",** **"CandleData": {** **"StartDate": "2024-02-20T07:05:00.000+00:00",** **"First": {** **"Bid": 1.25903,** **"Offer": 1.25913** **},** **"Last": {** **"Bid": 1.25898,** **"Offer": 1.25914** **},** **"High": {** **"Bid": 1.25909,** **"Offer": 1.25922** **},** **"Low": {** **"Bid": 1.25896,** **"Offer": 1.2591** **}** **},** **"ApplVerID": "FIX50SP2",**

	"SendingTime": "2024-02-20T07:07:54.036"
	}
	SAMPLE ERROR MESSAGE
	{
	"MsgType": "BusinessMessageReject",
	"BusinessRejectRefID": "24572562",
	"BusinessRejectReason": "UnsupportedMessageType",
	"Text": "rejecting message (first 200 chars):
	{\"MsgType\":\"HistoricCandleRequest\",
	\"ApplVerID\":\"FIX50SP2\",
	\"SendingTime\":\"2024-02-09T17:09:16.602\",
	\"CstmApplVerID\":
	\"IGUS/PriceHistory/V1\",
	\"ReqID\":\"24572562\",
	\"SecurityID\":\"CS.D.GBPUSD.CZD.IP\",
	\"SecurityIDS",
	"ApplVerID": "FIX50SP2",
	"SendingTime": "2024-02-05T06:17:53.681"
	}

3.10 Request quotas

An order quota refers to the limit or restriction imposed on the frequency at which individuals can send messages related to orders through an API. It specifically pertains to the limitations placed on the API usage and does not affect the content or nature of the messages themselves, such as request or response messages.

Here is the quota limit for **Trade/PostTrade**:

Quota interval	**Max limit**	**Burst interval**	**Burst limit**
1m	240	1s	20

3.11 Key terminologies

3.11.1 Security identifier

A security identifier is a unique code used to identify the instruments (currency pairs) in tastyfx system. It provides a uLastnique designation or code that helps differentiate and identify individual instruments/securities in a consistent manner.

Example: CS.D.USDCAD.CZD.IP

3.12 Different order types

There are several different order types used in trading, each serving specific purposes and catering to different trading strategies and objectives.

3.12.1 Market order

A market order is an instruction from a trader to a broker to execute a trade immediately at the best available price.

3.12.2 Limit order

A limit order is an instruction to your broker to execute a trade only at the specified level. To read more about limit orders, click here.

3.12.2.1 Stop order

A stop order, also known as a stop-loss order, is a conditional limit order used in trading to limit potential losses. It is an order that becomes a market order when the price of a security reaches a specified level, known as the stop price. To read more about stop order, click here.

3.12.2.2 Take profit order

It is a conditional limit order that automatically closes a position once the price of the security reaches the predetermined take-profit level.

3.12.3 Previously quoted order

A previously quoted order refers to an order that is placed with reference to a previously received price from tastyfx. It is important to note that previously quoted orders may be accepted or rejected by I, based on the validation of price tolerance against the current trading level for the respective security.

3.12.4 GoodTillCancel (GTC)

A good till cancel (GTC) is a **TimeInForce** attribute that remains active until it is either filled or manually canceled.

Note: It is mandatory to include a **Stop** order when placing a GTC order.

3.12.5 GoodTillDate (GTD)

A good till date (GTD) is a **TimeInForce** attribute that remains active until a specified date and is automatically canceled if not executed.

Note: It is mandatory to include a **Stop** order when placing a GTD order.

3.12.6 FillOrKill (FOK)

A fill or kill (FOK) is a **TimeInForce** attribute that requires immediate and complete execution of the entire order quantity or it is canceled ('killed') entirely.

3.12.7 ImmediateOrCancel (IOC)

An immediate or cancel (IOC) is a **TimeInForce** attribute that requires immediate execution of any portion of the order that can be filled, with the remaining unfilled quantity gets canceled ('immediately').