REALTIME ADMIN TRADING API INTEGRATION

The realtime trading API is based on a **websocket communication** using the **Protobuf's protocol for message** exchanges.

Protobuf is supported on several languages, so there is attached both the source *.proto* file and an already "compiled" file(.cs) for C# usage.

In order to see the utilization please look at the C# example provided where you can find a sample usage with:

1. Token Request
2. Open connection
3. Login
4. Request account list
5. Request snapshot of orders and positions
6. Contracts request
7. Insert/Modify/Cancel of orders

AUTH REQUEST

As you can see in the picture above and also in the C# example provided, the first thing to do is a POST request to the Auth Service:

- Endpoint:
  - Staging: https://staging-api.volumetricafx.com/api/v2/user/generateTradingToken
  - Production: https://api.volumetricafx.com/api/v2/user/generateTradingToken
- Headers:
  - x-api-key = token provided by email
- Json body's parameters:
  - "login": string = username
  - "password": string = password
  - "withDetails": bool = same as *get.dxfeed API.* If it is setted to true, it returns the exchanges entitled for the user
  - "version": int = version of the trading API used. This documentation is for v3, so use 3
  - "environment": int:
    - 0 – Production = used by propfirms' users
    - 1 – Staging = used for testing purposes by propfirms' managers/developers
  - platform: int
    - use this constant value 10000 (which means system)
  - "connectOnlyTrading": bool
    - true: **since it is just a session for trading API, this field must be sent to true**
- Returns:
  - "success": Boolean = if the auth has been correctly done
    - If true: there is the data field which contains:
      - tradingWssEndpoint: websocket endpoint for realtime trading api
      - tradingWssToken: token to be used when requesting the Login on wss trading api – it can be used only one time
      - tradingRestReportHost: Report API endpoint

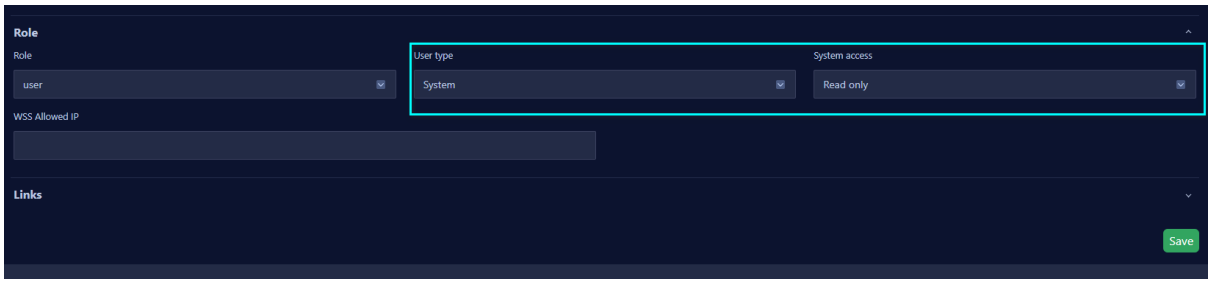
- tradingRestReportToken: Report API token – it can be used multiple times
- tradingRestTokenExpiration: token expiration on unix milliseconds

- If false, more details are available:
  - statusCode: int = http status code, most probably there will be always 401 (Unauthorized)
  - message: string = reason of the reject. Please show the message to the user since it could be easier to understand the reason of the failure: E.g.: dxFeed's agreements not signed, etc.

SYSTEM CREDENTIAL

In order to access the websocket and receive all the users and accounts available, it is necessary to create a system credential from the admin dashboard, in particular:

- userType = system
- systemAccess
  - readonly = it can only receive the information and updates, but it cannot place orders or close positions
  - liquidation = it can also cancel orders or close positions
  - fullTrading = full trading, potentially it could also open new positions



REALTIME TRADING API

As written at the beginning of this document, the realtime trading API is based on a websocket communication using the protobuf message protocol.

The URI of the websocket is provided by the "tradingWssEndpoint" field on the auth response.

While the are two main messages on message exchanges:

- ClientRequestMsg = it is a wrapper of the message from platform to the server. Read on .proto file for more details
- ServerResponseMsg = it is a wrapper of the message sent from the server to platforms.

Inside the .proto file there several comments in order to help the understand of various fields and their usage. Furthermore there are also may comments on the C# example.

This is the order of the operations that must be executed in order to be ready for order submission

- Open websocket connection
- Send a LoginRequestMsg with the token provided on the POST auth request
- Waiting the response of the login and manage the response

- If the login has been successfully, the session is ready to receive the account list and a snapshot for existing positions/orders can be requested. It must be performed using the "InfoReqMsg", which returns the snaphost and automatically also subscribes to the changes. In order to ensure consistency, it is suggested to request multiple modes with the same request through the "Modes" field. Based on the mode, it is returned the following:
    - Mode | 1 = it returns the account list
        - The account number contained on "AccountHeaderMsg" must be stored for the session lifetime because it should be used on account actions. E.g. Order insert, daily pl request and so on
    - Mode | 2 = it returns a snapshot of portfolio and the pending/partial filled orders
        - IMPORTANT: it returns a list of "OrderInfoMsg" and "BracketInfoMsg" with two different types of SnapType. Please open the example to understand how to manage it:
            - 2 | Historical = it is a snapshot of the order
            - 3 | HistPos = it is a snapshot of the portfolio
    - Mode | 3 = it returns a snapshot of the open positions and subscribes to it
        - IMPORTANT: if the account is on NETTING mode, the positionId is always equal to -1, since the reference of the position is the contractId. On futures and stocks, the only supported account type is NETTING.
- Once the initial snapshot has been received, there are several different messages that are received as PUSH from the server:
    - BalanceInfo = it is received at the beginning for a snapshot and then on account's balance changed
    - OrderInfo = it contains the updates for the pending orders
    - BracketInfo = it contains the updates for the pending brackets attached to the orders
    - PositionInfo = it returns the updates for the position which includes the openPnL
    - There are several other updates sent, please check directly on the .proto
- Send a cyclical PING message to the server each 30 seconds. There is a timeout on the server of 1 minute
- Then there are the following messages that may be received and must be handled:
    - Pong msg = returned after a ping message. It could be useful for the client in order to check that session is still open
    - LoggedOff = if it is received, it means that the session must be immediately closed. There is also the reason attached, e.g. concurrent sessions
    - LogMsg = a message that must be shown to the user

REPORT TRADING API

For the terminals which are interested to retrieve the historical information of an account, there are two different possibilities:

- AccountHistoricalSessionReq through the trading WSS, which returns the historical entities of the current session:
    - This call is very light since the information are returned from an in-memory cache
    - Each session starts around 16:05 CT, so after this time, the information are resettet
- Rest API:

- The host is returned on the auth request “tradingRestReportHost” and all the calls must be done adding the following header:
  - “Authorization”: “tradingRestReportToken” generated on auth response
- Please check the APIs available on the swagger attached (swagger_historical.json)