Eze EMS

Release Date: Feb 16, 2023

Eze EMS xAPI Getting Started Guide

This document contains information about getting started with Eze EMS xAPI.

Table of Contents

Legal Information

Copyright

Content

Information in this document is subject to change without notice. In the event that you are using a version of SS&C Eze products other than the most recent version, there may be discrepancies between the content of this documentation and the operation or visual presentation of your older version of the product. SS&C Eze does not warrant that this documentation is error free.

Trademarks

SS&C Eze

http://www.ezesoft.com/

Revision History

The table below provides a snap-shot of the updates in each revision of this document. A bar is displayed on the right side of the page to help you identify updates in the current release.

Version No.	Date	Summary of Update
V2023.1.0.2	Feb 16, 2023	There are no documentation updates in this xAPI release.
v2022.8.0.4	Dec 21, 2022	Added additional information for generating protobuf files in Initial Setup.
v2022.5.0.0	Sep 09, 2022	There are no documentation updates in this xAPI release.
v2022.3.0.0	Jun 10, 2022	There are no documentation updates in these xAPI releases.
v2022.2.3.0	Apr 07, 2022	Added the **GetLevel1MarketData** API.
v2022.2.2.0, v2022.2.1.0, v2022.2.0.0, v2021.9.0.0, v2021.8.1.0, v2021.7.0.0	NA	There are no documentation updates in these xAPI releases.
v2021.5.0.0	Jul 16, 2021	Added more detailed information on getting started using Python code samples. Information on using Python sample application is moved to a stand alone document, refer **Eze EMS xAPI Python Sample Application Guide**. Added new section **Eze EMS xAPI Frequently Asked Questions**.
v2021.2.0.0	Mar 31, 2021	Added the **GetTodaysActivityJson** API.
v2021.1.0.0	Feb 02, 2021	Added the following API and Custom Data Type: • GetStrategyList • StrategyListRow
v2020.9.0.0	Dec 18, 2020	Added the **SubscribeHeartBeat** API.
v2020.8.0.0	Nov 25, 2020	Added the following APIs and Custom Data Type: • ChangePasswordSRP • SubmitAllocationOrder • SubmitTradeReport • AllocationType

Version No.	Date	Summary of Update
		• OrderType • TradeType
v2020.7.3.0	Nov 10, 2020	Updated the stable version details of EMS xAPI.
v2020.7.2.0	Oct 30, 2020	Initial release.

Introduction

The purpose of this document is to help clients get started with the EMS xAPI application. This document provides a step-by-step process of using these APIs.

Eze EMS xAPI is robust and easy-to-use application that allows programmers and trading businesses to complete various trading workflows, and also access key information, including:

- Automating order routing - to smart order routers, algorithms and other trading systems.
- Routing orders to multiple brokers, dark pools, ATS, and MTFs via the Eze EMS Global Routing Network - across asset classes.
- Staging or routing single or pairs orders.
- Accessing balances, positions, executions, and other order details.
- Accessing comprehensive list and basket capabilities.

Although EMS xAPI can operate with all gRPC compatible languages, only Python language information is provided in this document as an example. Refer to this link for more information on gRPC.

Eze EMS xAPI Basics

The Eze EMS xAPI operates in conjunction with your existing Eze EMS account permissioning and entitlements. **The Eze EMS xAPI is not a standalone data feed application that is provided to you independent of the Eze EMS.** Please contact Eze Client Service if you need to request or make changes to appropriate permissions for your account.

Eze EMS xAPI Use Restrictions

As an Eze EMS xAPI user, you are prohibited from retransmitting any Eze Market Data using the Eze EMS xAPI, without the express prior written consent of Eze EMS and the exchanges or other third-party data providers (referred to as ***"Sources"*** in your end user agreement). Any unauthorized retransmission of Eze Market Data is a breach of your end user agreement and will cause immediate termination of your use of the Eze EMS, Eze Market Data, and the Eze EMS xAPI.

Any non-display usage of Eze Market Data, such as use of real- time data in algorithmic trading or program trading, is subject to the rules, regulations, and policies of the applicable exchanges and additional exchange fees may apply. In addition, you may have a non-display usage of Eze Market Data even if a display of real-time data occurs. Please review your Eze EMS end user agreement, and the exchanges' and third-party data providers' rules, regulations, and policies that apply to your use of the Eze EMS API (which apply to Eze EMS xAPI) and/or Eze Market Data. It is the sole responsibility of the Eze EMS xAPI user and each user receiving, directly or indirectly accessing or otherwise using Eze Market Data to determine whether your receipt, access or use is reportable and/or fee liable.

Eze EMS xAPI Version

This document covers all the APIs and updates to the Eze EMS xAPI that are part of 2023.1.0.2 release.

Download EMS xAPI

Contact your SS&C Eze client service representative for downloading Eze EMS xAPI.

Developer Support

- If you are an existing Eze EMS User, log in to access developer support documentation and sample code.
- You can contact us or request a demo if you want to explore more about EMS xAPI.
- You can send us an e-mail apisupport@ezesoft.com or call +1 312-442-8122.

Business Use Cases

Mentioned below are some of the business use cases, where Eze EMS xAPI adds immense value to buy-side businesses:

- **Trade Execution API**: With the help of Eze EMS xAPI, you can add more flexibility and customization to your complex trading workflows. The broker-neutral trading APIs can seamlessly plug into your sophisticated trading strategies and enable you to achieve best execution and value for your strategies from the markets.
- **Market Data API**: Eze EMS xAPI has a range of real-time and historical time-series and Market Data. Subject to the terms of your Eze EMS end user agreement and prior written consent of Eze and applicable exchanges and third-party information providers, a buy-side business can consume the data from the Market Data API into proprietary workflows to enhance predictive models and automated strategies to trade equities, futures, options etc.

Below are some of the use cases supported on Eze EMS xAPI:

- CRUD operations for Single and Pair Orders.
- Order Utility functions such as Checking Positions and Balances.
- Security and Symbol Guides.

About gRPC

Overview

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It can use protocol buffers as both its Interface Definition Language (IDL) and as its underlying message interchange format.

For more information on gRPC refer this link.

For information about the languages supported on gRPC refer to this link.

Getting Started Using Python Code Samples

Prerequisites

The Python 3.9 is installed. If it is not installed already, download and install Python (https://www.python.org/downloads/).

Initial Setup

Follow the steps below for initial setup.

1. Download and install Python (https://www.python.org/downloads/).
2. Run the following commands in command prompt to install **gRPC / gRPC tools** and **SRP**.

```
pip install grpcio == 1.46.0
pip install grpcio-tools == 1.46.0
pip install srp
pip install pycryptodomex
pip install protobuf == 3.20.1
```

Note: For more information on gRPC refer to this link.

3. Download the proto files from Git repository https://github.com/ezesoft/xapi/tree/master/protos.
4. Run the following commands in command prompt to generate the **order_pb2.py**, **order_pb2_grpc.py**, **utilities_pb2.py**, **utilities_pb2_grpc.py**, **market_data_pb2.py** and **market_data_pb2_grpc.py** files.

```
python -m grpc_tools.protoc -I../protos --python_out=. --grpc_python_out=.
../protos/order.proto
python -m grpc_tools.protoc -I../protos --python_out=. --grpc_python_out=.
../protos/utilities.proto
python -m grpc_tools.protoc -I../protos --python_out=. --grpc_python_out=.
../protos/market_data.proto
```

Note: Replace **../protos** with your local machine directory path containing **.proto** files. There are two (2) instances of it in each of the command above.

5. Download the **roots.pem** file from gRPC GitHub repository. Make sure you save this file in the same folder along with _pb2.py files generated in the previous step.

The following sections provide information on Establishing Server Connection, Logging into SRP, Submitting an Order, Subscribing to Order Info, and Requesting and Watching Market Data.

Note: The following examples assume you are comfortable working with Python and Eze EMS data via the command line. If you prefer to work with a GUI interface, we have provided instructions and a sample application as part of the **Eze EMS xAPI Python Sample Application Guide**.

Connecting/Disconnecting

Standard Login (non-SRP)

In order to establish a secured connection with server, you must have the server's host IP address, port number and a local roots.pem (root certificates) file. If you need any information on these details contact your SS&C Eze client service representative.

EMS xAPI supports Secure Remote Password (SRP) protocol, refer the Logging via SRP Method for more details.

Follow the steps below to connect/disconnect EMS xAPI:

1. Import the utility modules generated during step 4 of Initial Setup above, to access the EMS xAPI Utility interface. You can use the below code to run the commands.

```
import grpc
import utilities_pb2 as util
import utilities_pb2_grpc as util_grpc
```

2. Use the below code to create a stub to gain access to the remote Utility interfaces. EMS xAPI requires Transport Layer Security (TLS), so you must establish an encrypted connection. You can use this roots.pem file or it is also included in the EMS xAPI repository provided by the SS&C Eze client service representative.

```
with open(r'.\roots.pem', 'rb') as f: cert = f.read()
channel = grpc.secure_channel('SERVER:PORT', grpc.ssl_channel_credentials(root_
certificates=cert))
util_stub = util_grpc.UtilityServicesStub(channel)
```

Note: The server details and authentication credentials are provided by SS&C Eze client service representative.

3. Use the stub to connect and authenticate with EMS xAPI.

```
connect_response = util_stub.Connect(util.ConnectRequest(UserName='USER',
Domain='DOMAIN', Password='PASSWORD', Locale='LOCALE'))

print('Connect result: ', connect_response.Response)
```

Note: On successful login you receive a unique glx2 UserToken in the response. Do not lose this token as it must be provided in all subsequent calls to the server.

4. You can now verify the login succeeded and disconnect from the server.

```
if connect_response.Response == 'success':

   disconnect_response = util_stub.Disconnect(util.DisconnectRequest
(UserToken=connect_response.UserToken))

   print('Disconnect result: ', disconnect_response.ServerResponse)
```

You have now connected, authenticated, and disconnected from the server using EMS xAPI.

You can also use the code snippet from the Git repository to establish a secured server connection. You need to provide information for the Field names mentioned below.

Field Name	Required?	Data Type	Accepted Values/Examples
UserName	Yes	string	A valid user name
Domain	Yes	string	A valid domain name
Password	Yes	string	A valid password
Locale	Yes	string	The region the user wants to connect. Example: AMERICAS, ASIA etc.

Logging via SRP Method

Secured Remote Password (SRP) is a mechanism that allows user to get authenticated without passing the password to the server. By implementing the SRP support in EMS xAPI server, you are able to send hash coded password during login. For more information on SRP refer to this link.

Note: To login using SRP method your domain has to be SRP enabled. Contact your SS&C Eze client service representative for assistance.

Note: Enter valid USER and DOMAIN details in the code snippet to login. Contact your SS&C Eze client service representative for more information.

SRP login method uses **StartLoginSrp** and **CompleteLoginSrp** APIs to login. Use the code snippet from the Git repository to login to EMS xAPI server via SRP method. You need to provide information for the Field names mentioned below.

Field Name	Required?	Data Type	Accepted Values/Examples
Identity	Yes	string	A valid user identity (combination of user-name@domain)
srpTransactId	Yes	string	Unique transaction ID per user
strEphA	Yes	string	This field is specific to SRP6 Protocol
strMc	Yes	string	This field is specific to SRP6 Protocol
UserName	Yes	string	A valid user name
Domain	Yes	string	A valid domain name
Locale	Yes	string	The region the user wants to connect. Example: AMERICAS, ASIA etc.

Note: On successful login you receive a unique glx2 UserToken in the response. Do not lose this token as it must be provided in all subsequent calls to the server.

Placing an Order

Follow the steps below to place an order:

1. Import the order modules generated during step 4 of Initial Setup above, to access the EMS xAPI Order interfaces.

```
import order_pb2 as ord
import order_pb2_grpc as ord_grpc
```

2. Create a stub to gain access to the remote Order interfaces. Since you are reusing the channel object; there is no need to recreate it.

```
ord_stub = ord_grpc.SubmitOrderServiceStub(channel)
```

3. Use the below code to submit the order request.

```
order_response = ord_stub.SubmitSingleOrder(ord.SubmitSingleOrderRequest
(Symbol='TSLA', Side='BUY', Quantity=5000, Route='ROUTE', Account='ACCOUNT',
OrderTag='MyOrderId', UserToken=connect_response.UserToken))

print('Order result: ', order_response)
```

Note: The route and account information are provided by SS&C Eze client service representative.

Note: When submitting an order, you can optionally populate the **OrderTag** property with a unique identifier so you can match an order event from EMS xAPI with an order in your system.

Alternatively, you can use the code snippet from the Git repository for submitting a single order using EMS xAPI server. Refer the table below and enter valid details in the code snippet to submit an order.

Field Name	Required?	Data Type	Description
Symbol	Yes	string	Valid ticker symbol. Example: AAPL, IBM or VOD.LSE etc.
Side	Yes	string	BUY, SELL, SELLSHORT NOTE: To send an order with side SELLSHORT, the extended field **SHORT_LOCATE_ID** must be assigned. The **SHORT_LOCATE_ID** is an ID assigned to short sell orders. Similarly, to send a **Buy To Cover** order, set the side to **BUY** and assign the extended field **TO_OPEN_POS** to the required volume
Quantity	Yes	int32	Value > 0
Route	Yes	string	Route name as shown in Eze EMS. NOTE: This field is also

Field Name	Required?	Data Type	Description
			referred to as **Exit Vehicle**
Account	Yes	string	Semi colon separated values that represent either Trade or Neutral accounts the user has permission to. Example: TAL;TEST;USER1;TRADE or TAL;TEST;USER2;NEUTRAL
OrderTag	No	string	Order Tag
TicketId	No	string	Ticket ID
UserToken	Yes	string	A server generated GUID that is given as response to the client during the first login
Staged	No	bool	TRUE or FALSE (Note: in order to send a staged order, this field becomes mandatory and has to be set as TRUE only)
ClaimRequire	No	bool	TRUE or FALSE (Note: setting TRUE value envisages a user running Eze EMS who then claims the Order so it can switch from Pending to Live State)
GoodFrom	No	google.protobuf.Timestamp	Time at which order is first valid for execution
TimeInForce	No	ExpirationType	Time or date at which order is no longer valid. In case no value is provided, DAY expiration type is set by default. Refer SubmitSingleOrder API in Eze EMS Technical Reference Document
PriceType	No	PriceTypeEnum	By default the price type is set to **Market**, in case no value is provided. Refer SubmitSingleOrder API in Eze EMS Technical Reference Document

Field Name	Required?	Data Type	Description
Price	No	google.protobuf.DoubleValue	Limit price submitted in order
StopPrice	No	google.protobuf.DoubleValue	Stop Price
UserMessage	No	string	User Message/Notes
ExpirationDate	No	google.protobuf.Timestamp	Date at which order is no longer valid

Getting Order Details

You can request order activity from EMS xAPI to see the status and details of an order. EMS xAPI offers you both unary (request-response) and streaming interface to retrieve order activity. Order activity includes orders you submitted to the system as well as activity on the order, such as executions (fills) or rejections.

Refer the code snippet for GetTodaysActivityJson interface provided in the Git repository for unary (request-response) type order activity. For streaming interface refer SubscribeOrderInfo code snippet provided in the Git repository.

You can add optional filters to refine the data results by adding a single filter IncludeUserSubmitOrder to limit the request to order submission activity.

```
activity_response = util_stub.GetTodaysActivityJson(util.TodaysActivityJsonRequest
(IncludeUserSubmitOrder=True, UserToken=connect_response.UserToken))
```

Note: By default, all include filters are set False.

In the Placing an Order section, an OrderTag was provided on the request. You can use the OrderTag to find a specific order and lookup the EMS xAPI Order ID.

```
xapi_order_id = df[(df['OrderTag']=='MyOrderId')]['OrderId'][0]

print('The xAPI OrderId for my order is ', xapi_order_id)
```

Note: You can use the EMS xAPI Order ID to cancel and modify existing orders.

Getting Execution Details

You can use the same Getting Order Details interface to retrieve order execution details from EMS xAPI by changing the filter.

You can request fill activity from EMS xAPI by adding a single filter IncludeExchangeTradeOrder.

```
activity_response = util_stub.GetTodaysActivityJson(util.TodaysActivityJsonRequest
(IncludeExchangeTradeOrder=True, UserToken=connect_response.UserToken))
```

Refer the code snippet provided in the Git repository to get the execution details.

Cancelling an Order

In the Getting Order Details section, you retrieved the EMS xAPI Order ID of an order placed in the Placing an Order section. In this section, you will use the Order ID to request the order be cancelled.

```
cancel_response = ord_stub.CancelSingleOrder(ord.CancelSingleOrderRequest(OrderId=xapi_
order_id, UserToken=connect_response.UserToken))

print('Cancel result: ', cancel_response)
```

Refer the code snippet provided in the Git repository to cancel an order.

Subscribing to Market Data

Using EMS xAPI you can request a market data snapshot as well as subscribe to future market data updates. This section covers both the snapshot and streaming cases for Level 1 market data.

Follow the steps below to subscribe to market data:

1. To access the EMS xAPI Market Data interfaces, import the market_data modules generated during the step 4 of Initial Setup. Since gRPC streaming is a blocking operation, data coming from the server should be processed on a separate thread so your application is not blocked. To do this, include the threading module as well.

   ```
   from threading import Thread

   import market_data_pb2 as md

   import market_data_pb2_grpc as md_grpc
   ```

2. Use the following code to define a new function to process market data returned by the server:

```
def  handle_data(response):
    try:
    for tick in response:
        if tick.Trdprc1.DecimalValue == 0.0:
            continue
        print('Received market data for {0}, Last: {1}'.format(tick.DispName,
tick.Trdprc1.DecimalValue))
    except  Exception  as e:
    print(e)
```

3. Create a stub to gain access to the remote Market Data interfaces:

```
md_stub = md_grpc.MarketDataServiceStub(channel)
```

4. Request market data from the server. You can request data for one or more securities at a time:

```
response = md_stub.SubscribeLevel1Ticks(md.Level1MarketDataRequest(Symbols=
['TSLA'], Request=True, Advise=True, UserToken=connect_response.UserToken))
```

Note: If you want a snapshot of the market data (i.e., the current values), set the Request parameter to True. If you want to receive all subsequent market ticks (i.e., get future market data updates), set the Advise parameter to True.

5. Create and start the data handler thread and pass the iterable response object to the thread function:

```
t = Thread(target=handle_data, args=(response, ))
t.start()
```

6. When you need to shut down, cancel the request and join the thread to wait for it to terminate gracefully:

```
response.cancel()
t.join()
```

Alternatively, you can use the code snippet from the Git repository to watch market data using EMS xAPI server. You need to provide information for the Field names mentioned below.

Parameter	Required?	Data Type	Description
UserToken	Yes	string	A server generated GUID that is given as response to the client during the first login
Symbols	Yes	repeated string	Valid ticker symbol. Example: AAPL, IBM or VOD.LSE etc.
RegionalExchangeIds	No	repeated string	Regional exchange ID
Request	Yes	bool	If set to **True**, a current snapshot of the data will be retrieved
Advise	Yes	bool	If set to **True**, real-time updates from the server will be registered for

Eze EMS xAPI Frequently Asked Questions

1. How to track an order?

When you create an order using xAPI, you can specify an external identifier in the OrderTag property. xAPI will include the OrderTag in all update messages. This mechanism allows you to match orders between your system and ours.

When you request order activity using either the **Unary** GetActivityJson or the **Streaming** SubscribeOrderInfo API, the OrderTag property will contain the value you specified.

In case you did not specify a value to OrderTag during the order creation, then you will have to match the orders between your system and ours based on properties such as symbol and quantity.

2. What is the difference between Unary and Streaming APIs?

GetActivityJson is an example of a **Unary** RPC. You make a request and get a single response. Unary is sometimes referred to as the request-response communication pattern.

SubscribeOrderInfo is an example of a **Streaming** RPC. You make a request and receive data from the server as it is available. Technically, when calling a Streaming API, you are provided an iterator object; each time you accesses the iterator (typically in a loop), you are blocked until data is available from the server.

Since streaming RPC blocks an application, you should loop through the iterator object from a dedicated thread.

A good example of a Streaming RPC is market data: you request live price updates for a security and receive the updates as they arrive from the exchange.

3. Can Eze OMS/Eze EMS user trigger automated compliance rule from EMS xAPI?

Yes, as an Eze EMS xAPI user you can trigger the automated compliance rule from xAPI and run automated compliance checks for xAPI originated order. This setting is configured in Eze OMS and is similar to the process in Eze EMS.

The FID used for this is **COMPLIANCE_USER_OPTION_FLAGS (31640) w/ Flag = NO_TOUCH_AUTO_COMPLIANCE (1)**. Additionally the extended fields that are to be included with the order are **EZE_OMS_MANAGER;EZE_OMS_TRADER;CUSTODIAN**

4. How to login using SRP?

EMS xAPI supports both SRP and non-SRP login methods. To login using SRP method your domain has to be SRP enabled. If your domain is SRP enabled, both the SRP and standard login methods work. If your domain is not SRP enabled, then only the standard login will work.

5. How can I know if my EMS xAPI session is active?

You can subscribe to **SubscribeHeartBeat** API, to know your EMS xAPI connection status. On subscribing to this API, you will be intimated and requested to reconnect if an active connection with the server is terminated.

Appendix A

The following APIs are provided as part of Eze EMS xAPI. For further information refer to the Eze EMS xAPI Technical Reference Document.

Order APIs		
API Name	**Input**	**Output**
Order (Single Order APIs)		
SubmitSingleOrder	SubmitSingleOrderRequest	SubmitSingleOrderResponse
ChangeSingleOrder	ChangeSingleOrderRequest	ChangeSingleOrderResponse
CancelSingleOrder	CancelSingleOrderRequest	CancelSingleOrderResponse
SubmitAllocationOrder	SubmitAllocationOrderRequest	SubmitAllocationOrderResponse
SubmitTradeReport	SubmitTradeReportRequest	SubmitTradeReportResponse
Order (Pair Order APIs)		
SubmitPairOrder	SubmitPairOrderRequest	SubmitPairOrderResponse
ChangePairOrder	ChangePairOrderRequest	ChangePairOrderResponse
CancelPairOrder	CancelPairOrderRequest	CancelPairOrderResponse
Order (Basket Order APIs)		
SubmitBasketOrder	BasketOrderRequest	BasketOrderResponse
Order (Miscellaneous)		
SubmitSeedData	SubmitSeedDataRequest	SubmitSeedDataResponse
GetUserAccounts	UserAccountsRequest	UserAccountsResponse
SubscribeOrderInfo	SubscribeOrderInfoRequest	stream SubscribeOrderInfoResponse
SubscribeOrderInfoJson	SubscribeOrderInfoJsonRequest	stream Sub-scribeOrderInfoJsonResponse
GetOrderDetailByOrderIdJson	OrderDetailByOrderIdJsonRequest	OrderDetailByOrderIdJsonResponse
GetOrder-DetailByDateRangeJson	Order-DetailByDateRangeJsonRequest	stream Order-DetailByDateRangeJsonResponse

MarketData		
API Name	Input	Output
GetDailyWeeklyMonthlyBars	DailyWeeklyMonthlyBarsRequest	DailyWeeklyMonthlyBarsResponse
GetIntradayBars	IntradayBarsRequest	IntradayBarsResponse
GetOptionChainForUnderlier	OptionChainRequest	OptionChainResponse
GetSymbolReferenceData	SymbolReferenceDataRequest	SymbolReferenceDataResponse
GetTickData	TickDataRequest	TickDataResponse
GetOptionsAndGreekData	OptionsAndGreekDataRequest	OptionsAndGreekDataResponse
GetSecurityData	SecurityDataRequest	SecurityDataResponse
GetOptionSymbolFromDescription	OptionSymbolFromDescriptionRequest	OptionSymbolFromDescriptionResponse
GetDescriptionFromOptionSymbol	DescriptionFromOptionSymbolRequest	DescriptionFromOptionSymbolResponse
SubscribeLevel1Ticks	Level1MarketDataRequest	stream Level1MarketDataResponse
UnSubscribeLevel1Data	UnSubscribeLevel1DataRequest	UnSubscribeLevel1DataResponse
GetLevel1MarketData	Level1MarketDataRequest	Level1MarketDataRecordResponse
SubscribeLevel2Ticks	Level2MarketDataRequest	stream Level2MarketDataResponse
UnSubscribeLevel2Data	UnSubscribeLevel2DataRequest	UnSubscribeLevel2DataResponse
AddSymbols	AddSymbolsRequest	AddSymbolsResponse
RemoveSymbols	RemoveSymbolsRequest	RemoveSymbolsResponse
GetSymbolsFromCompanyName	SymbolsFromCompanyNameRequest	SymbolsFromCompanyNameResponse
GetSymbolFromAlternateSymbology	SymbolFromAlternateSymbologyRequest	SymbolFromAlternateSymbologyResponse

Utilities		
API Name	Input	Output
Connect	ConnectRequest	ConnectResponse
Disconnect	DisconnectRequest	DisconnectResponse
GetTodaysBalances	TodaysBalancesRequest	TodaysBalancesResponse
GetTodaysActivity	TodaysActivityRequest	TodaysActivityResponse
GetTodaysActivityBook	TodaysActivityBookRequest	TodaysActivityBookResponse
GetTodaysBrokenDownPos-itions	TodaysBrokenDownPos-itionsRequest	TodaysBrokenDownPos-itionsResponse
GetTodaysNetPositions	TodaysNetPositionsRequest	TodaysNetPositionsResponse
SubscribeHeartBeat	SubscribeHeartBeatRequest	SubscribeHeartBeatResponse
GetStrategyList	StrategyListRequest	StrategyListResponse
GetTodaysActivityJson	TodaysActivityJsonRequest	TodaysActivityJsonResponse
Secured Remote Password		
StartLoginSrp	StartLoginSrpRequest	StartLoginSrpResponse
CompleteLoginSrp	CompleteLoginSrpRequest	CompleteLoginSrpResponse
ChangePasswordSRP	ChangePasswordSRPRequest	ChangePasswordSRPResponse

Note: Only the mandatory API fields are mentioned here. Contact your SS&C Eze client service representative for a complete list of extended fields.