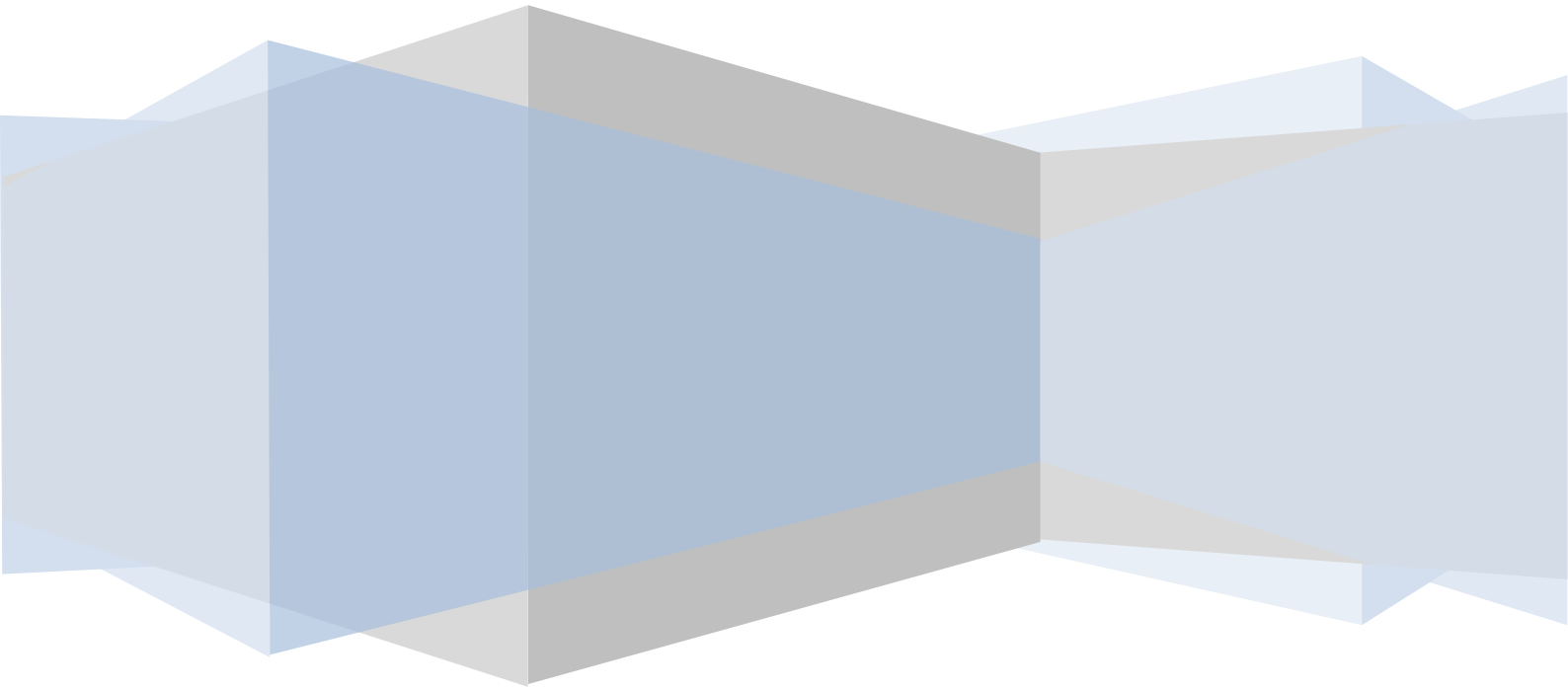
SET CONNECT – ITCH Market Data Protocol Specification

The Stock Exchange of Thailand



Version: 1.6.7

Disclaimer

No part of this document may be reproduced, copied, published, transmitted, or sold in any form or by any means without the expressed written permission of The Stock Exchange of Thailand ("SET").

While reasonable efforts have been made to ensure that the contents of this document are accurate, the document is provided strictly "as is", and no warranties of accuracy are given concerning the contents of the information contained in this document, including any warranty that the document will be kept up to date. SET reserves the right to change details in this document without notice. To the extent permitted by law no liability (including liability to any person by reason of negligence) will be accepted by SET or its employees for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document.

About this document

Before reading this document, please firstly visit the document "SET CONNECT Guideline".

There are 2 protocols using to access the SET CONNECT system for market data dissemination; ITCH and FIX 5.0 SP2. This document describes only for ITCH protocol. For FIX Market Data Protocol, please refer to "SET CONNECT FIXMD Specification" document.

This document describes 2 major parts of ITCH protocol.

1. The semantics of the ITCH protocol and the syntax of the session related ITCH messages that called SoupBinTCP and MoldUDP64.
2. The description of each ITCH message format and the example of message flow.

The purpose of this document together with its related documents is to serve as a specification of the ITCH protocol when implementing a client.

Intended audience

The information in this document is for Independent Software Vendors ("ISV"), Member's in-house developers ("In-house"), Market Data Vendors ("MDV") or Market Data Users which intend to develop market data clients via the ITCH protocol or members which aim to understand the protocol interaction and connectivity methods.

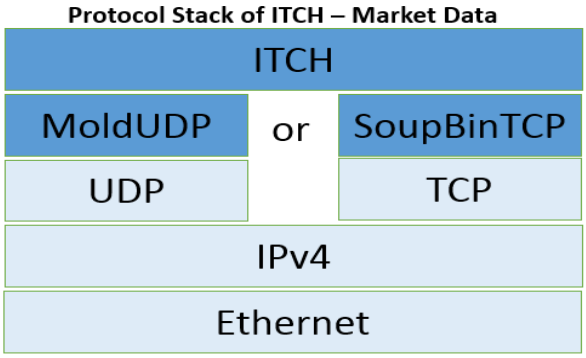
Contents

1. Communication Protocol Overview 5
2. SoupBinTCP Protocol 5
   2.1 Overview 5
      2.1.1 SoupBinTCP Logical Packets 6
      2.1.2 Protocol flow 6
      2.1.3 Heartbeat 7
      2.1.4 Data type 7
   2.2 SoupBinTCP Packet Types 7
      2.2.1 Logical Packets Sent by a SoupBinTCP Server 7
      2.2.2 Logical Packets Sent by a SoupBinTCP Client 9
3. MoldUDP64 Protocol 11
   3.1 Overview 11
   3.2 Assumptions 11
   3.3 Terms 11
   3.4 Downstream packet 12
      3.4.1 Header 12
      3.4.2 Message block 13
      3.4.3 Heartbeats 13
      3.4.4 End of session 13
   3.5 Request packet 14
4. Snapshot 14
   4.1 Glimpse Message format 15
5. ITCH Market Data System Overview 16
   5.1 System Architecture 16
   5.2 ITCH Client Connection 19
      5.2.1 ITCH MDF Client Connection 19
      5.2.2 ITCH MDFA Client Connection 20
      5.2.3 ITCH Multicast Client Connection 21
6. ITCH Client Development 22
   6.1 ITCH MDF and MDFA for Price Feed Client (SoupBinTCP) 22
      6.1.1 Start of Day or Reconnect with Snapshot 23
      6.1.2 Reconnect 24
   6.2 ITCH MDFA for News Client (SoupBinTCP) 24
      6.2.1 Start of Day or Reconnect 24

6.2.2 Download News/Report Attached file 26
6.3 ITCH Multicast Client (MoldUDP64) 26
6.4 ITCH Message Flow 30
6.4.1 ITCH Glimpse Snapshot 30
6.4.2 ITCH Live Feed 30
6.4.3 ITCH MDFA (News) Live Feed 31
6.5 TCH Java Example 31
6.5.1 SoupBinTCP Java library Implementation 32
6.5.2 Multicast Receiver Implementation 34
6.5.3 SFTP Implementation 36
7. ITCH Message format 37
7.1 ITCH Message Overview 37
7.2 ITCH Data Type 39
7.3 Time Messages 39
7.3.1 Seconds Messages (T) 40
7.4 Reference Data Messages 41
Reference Data Overview 41
7.4.1 Order Book Directory (R) 42
7.4.2 Exchange Directory (e) 54
7.4.3 Market Directory (m) 54
7.4.4 Combination Order Book Leg Directory (M) 55
7.4.5 Tick Size Table Entry (L) 56
7.4.6 Price Limit Message (k) 56
7.5 Event and State Change Messages 57
7.5.1 System Event Message (S) 57
7.5.2 Order Book State Message (O) 57
7.5.3 Halt Information Message (I) 60
7.6 Price Messages 62
7.6.1 Market by Price (MBP) Incremental Message (b) 62
7.6.2 Equilibrium Price Message (Z) 64
7.6.3 Trade Ticker Message (i) 66
7.6.4 Trade Statistics Message (I) 68
7.6.5 iNAV Message (f) 70
7.6.6 Index Price Message (J) 70
7.6.7 Market Statistic Message (g) 71
7.6.8 Reference Price Message (Q) 73

7.6.9 Open Interest Message (h) 74
7.7 Market Announcements 75
7.7.1 Market Announcement Message (N) 75
7.7.2 News Message (j) 78
8. The Example Message Flow 84
8.1 SoupBinTCP example message flow 84
8.1.1 Login 84
8.1.2 Heartbeat 86
8.2 ITCH example message flow 87
8.2.1 Reference Data and Corporate Action 87
8.2.2 Trading session and instrument state example flow 88
- In case of normal trading session and instruments are halted or suspended 88
- In case of SET Circuit Breaker 88
- In case of TFEX Price Break 89
- In case of the instruments is on last trading date/time 89
8.2.3 Equilibrium price example flow 90
8.2.4 Market by Price (MBP) example flow 90
8.3 How to maintain the local order book view 91
8.3.1 Example 1 – Initial Market 91
8.3.1.1 An order is added to the book 91
8.3.1.2 Change an order 92
8.3.1.3 Delete an order which deletes the price level 92
8.3.2 Example 2 – A level can temporarily be beyond the maximum level during processing multiple updates in one MBP message 93
Revision History 95

1. Communication Protocol Overview



ITCH is a direct market data feed protocol for SET CONNECT. ITCH is composed of a series of messages that contain basic reference data of the order books, state changes, real-time price and market announcements.

ITCH is an outbound market data feed only. For order entry or drop copy, please refer to SET CONNECT OUCH or FIX specification.

ITCH can use either MoldUDP64 or SoupBinTCP as the transport protocol.

2. SoupBinTCP Protocol

2.1 Overview

SoupBinTCP is a lightweight point-to-point protocol, built on top of TCP/IP sockets that allow delivery of a set of sequenced messages from a server to a client in real-time. SoupBinTCP guarantees that the client receives each message generated by the server in sequence, even across underlying TCP/IP socket connection failures.

SoupBinTCP clients can send messages to the server. These messages are not sequenced and may be lost in the case of a TCP/IP socket failure.

SoupBinTCP is ideal for systems where a server needs to deliver a logical stream of sequenced messages to a client in real-time but does not require the same level of guarantees for client generated messages either because the data stream is unidirectional or because the server application generates higher-level sequenced acknowledgments for any important client-generated messages.

SoupBinTCP is designed to be used in conjunction with higher-lever protocols that specify the contents of the messages that SoupBinTCP messages deliver. The SoupBinTCP protocol layer is opaque to the higher-level messages.

SoupBinTCP also includes a simple scheme that allows the server to authenticate the client on login.

2.1.1 SoupBinTCP Logical Packets

The SoupBinTCP client and server communicate by exchanging a series of logical packets. Each SoupBinTCP logical packet has:

- A two byte big-endian length that indicates the length of rest of the packet (meaning the length of the payload plus the length of the packet type – which is 1)
- A single byte header which indicates the packet type
- A variable length payload

SoupBinTCP Logical Packet Structure:

Two Byte Packet Length	Packet Type	Variable-length payload

ⓘ The SoupBinTCP logical packets do not necessarily map directly to physical packets on the underlying network socket; they may be broken apart or aggregated by the TCP/IP stack.

ⓘ The SoupBinTCP protocol does not define a maximum payload length.

2.1.2 Protocol flow

A SoupBinTCP connection begins with the client opening a TCP/IP socket to the server and sending a Login Request Packet. If the login request is valid, the server responds with a Login Accepted Packet and begins sending Sequenced Data Packets. The connection continues until the TCP/IP socket is broken.

Each Sequenced Data Packet carries a single higher-level protocol message. Sequenced Data Packets do not contain an explicit sequence number; instead both client and server compute the sequence number locally by counting messages as they go. The sequence number of the first sequenced message in each session is always 1.

Typically, when initially logging into a server the client will set the Requested Sequence Number field to 1 and leave the Requested Session field blank in the Login Request Packet. The client will then inspect the Login Accepted Packet to determine the currently active session. Starting at 1, the client begins incrementing its local sequence number each time a Sequenced Data Packet is received. If the TCP/IP connection is ever broken, the client can then re-log into the server indicating the current session and its next expected sequence number. By doing this, the client is guaranteed to always receive every sequenced message in order, despite TCP/IP connection failures.

SoupBinTCP also permits the client to send messages to the server using Unsequenced Data Packets at any time after the Login Accepted Packet is received. These messages may be lost during TCP/IP socket connection failures.

2.1.3 Heartbeat

SoupBinTCP uses logical heartbeat packets to quickly detect link failures. The server will send a Server Heartbeat packet every 1 seconds has passed since the server last sent any data. This ensures that the client will receive data on a regular basis. If the client does not receive anything (neither data nor heartbeats) for an extended period of time, it can assume that the link is down and attempt to reconnect using a new TCP/IP socket.

Similarly, once logged in, the client must send a Client Heartbeat packet every 1 seconds has passed since the client last sent anything. If the server doesn't receive anything from the client for an extended period of time (typically 45 seconds), it can close the existing socket and listen for a new connection.

2.1.4 Data type

Character data fields are standard ASCII bytes. Integer fields are binary big-endian values.

2.2 SoupBinTCP Packet Types

2.2.1 Logical Packets Sent by a SoupBinTCP Server

2.2.1.1 Login Accepted Packet – “A”

The SoupBinTCP server sends a Login Accepted Packet in response to receiving a valid Login Request from the client. This packet will always be the first non-debug packet sent by the server after a successful login request.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"A"	Login Accepted Packet
Session	3	10	Alpha-numeric	The session ID of the session that is now logged into. Left padded with spaces.
Sequence Number	13	20	Numeric	The sequence number in ASCII of the next Sequenced Message to be sent. Left padded with spaces.

2.2.1.2 Login Rejected Packet – “J”

The SoupBinTCP server sends this packet in response to an invalid Login Request Packet from the client. The server closes the socket connection after sending the Login Rejected Packet. The Login

Rejected Packet will be the only non-debug packet sent by the server in the case of an unsuccessful login attempt.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"J"	Login Rejected Packet
Reject Reason Code	3	1	Alpha	See Login Reject Codes below.

Login Reject Codes

Name	Explanation
"A"	Not Authorized. There was an invalid username and password combination in the Login Request Message.
"S"	Session not available. The Requested Session in the Login Request Packet was either invalid or not available.

2.2.1.3 Sequenced Data Packet – "S"

The Sequenced Data Packets act as an envelope to carry the actual sequenced data messages that are transferred from the server to the client. Each Sequenced Data Packet carries one message from the higher-level protocol. The sequence number of each message is implied; the initial sequence number of the first Sequenced Data Packet for a given TCP/IP connection is specified in the Login Accepted Packet and the sequence number increments by 1 for each Sequenced Data Packet transmitted.

Since SoupBinTCP logical packets are carried via TCP/IP sockets, the only way logical packets can be lost is in the event of a TCP/IP socket connection failure. In this case, the client can reconnect to the server and request the next expect sequence number and pick up where it left off.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"S"	Sequenced Data Packet
Message	3	Variable	Any	Defined by a higher-level protocol

2.2.1.4 Server Heartbeat Packet – "H"

The server will send a Server Heartbeat Packet every 1 second has passed where no data has been sent to the client. The client can then assume that the link is lost if it does not receive anything for an extended period of time.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"H"	Server Heartbeat Packet

2.2.2 Logical Packets Sent by a SoupBinTCP Client

2.2.2.1 Login Request Packet – "L"

The SoupBinTCP client must send a Login Request Packet immediately upon establishing a new TCP/IP socket connection to the server. Client and server must have mutually agreed upon the username and password fields. They provide simple authentication to prevent a client from inadvertently connecting to the wrong server. Both Username and Password are case-insensitive and should be padded on the right with spaces.

The server can terminate an incoming TCP/IP socket if it does not receive a Login Request Packet within a reasonable period of time (typically 30 seconds).

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"L"	Login Request Packet
Username	3	6	Alphanumeric	Username
Password	9	10	Alphanumeric	Password
Requested Session	19	10	Alphanumeric	Specifies the session the client would like to log into, or all blanks to log into the currently active session.
Requested Sequence Number	29	20	Numeric	Specifies the next sequence number in ASCII the client wants to receive upon connection, or 0 to start receiving the most recently generated message.

2.2.2.2 Unsequenced Data Packets – "U"

The Unsequenced Data Packets act as an envelope to carry the actual data messages that are transferred from the client to the server. These messages are not sequenced and may be lost in the event of a socket failure. The higher-level protocol must be able to handle these lost messages in the case of a TCP/IP socket connection failure.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"U"	Unsequenced Data Packet
Message*	3	Variable	Any	Message content and length are defined by a higher-level protocol, e.g. ITCH

* Message should not contain any linefeed characters (ASCII 10 decimal, 0x0A hex).

2.2.2.3 Client Heartbeat Packets – "R"

The client should send a Client Heartbeat Packet every 1 second has passed where no data has been sent to the server. The server can then assume that the link is lost if it does not receive anything for 45 seconds.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"R"	Client Heartbeat Packet

2.2.2.4 Logout Request Packet – "O"

The client may send a Logout Request Packet to request the connection be terminated. Upon receiving a Logout Request Packet, the server will immediately terminate the connection and close the associated TCP/IP socket.

Name	Offset	Length	Value	Notes
Packet Length	0	2	Integer	Number of bytes after this field until the next packet
Packet Type	2	1	"O"	Logout Request Packet

3. MoldUDP64 Protocol

3.1 Overview

MoldUDP64 is a networking protocol that allows efficient and scalable transmission of data messages in a “one transmitter to many listeners” scenario. MoldUDP64 is a lightweight protocol layer built on top of UDP that provides a mechanism for listeners to detect and re- request missed packets.

In MoldUDP64 each outbound packet is transmitted only once regardless of the number of listeners. Multiple messages may also be aggregated into a single network packet to reduce network traffic.

This section describes the messages sent between a MoldUDP64 server and its clients. MoldUDP64 transmitters send downstream packets via UDP multicast to transport the normal data stream sent to the listeners. These packets are also sent via UDP unicast in response to a Request message submitted by a listener. MoldUDP64 clients can send these Request messages to request the retransmission of any desired packets from the data stream.

The MoldUDP64 server will transmit on a well-known multicast group for each type of downstream MoldUDP64 data stream on a network. The listeners must subscribe to this multicast group to receive the downstream data. One or more Re-request Servers may also be deployed to service any unicast client requests for retransmission of specific messages. The listeners must be configured with these IP addresses and port combinations to which they can submit the requests.

3.2 Assumptions

All number fields in the MoldUDP64 messages specified in this document (i.e. sequence number, message counts and message lengths etc.) are binary numbers formatted in Big Endian mode (i.e. most significant byte first).

ⓘ This need not apply to the data contained in the Message Data fields of the Message Blocks.

3.3 Terms

Term	Definition
Message	A message is an atomic piece of information carried by the MoldUDP64 protocol. MoldUDP64 can theoretically handle individual messages from zero bytes up to 64KB in length although individual messages should be kept small enough so that the UDP underlying network protocol can efficiently carry the resulting MoldUDP64 packets. ⓘ The contents of a MoldUDP64 message are defined by the higher-level application.
Session	A Session is a sequence of one or more messages. While a single session can last indefinitely, typically the application will define a session to logically group messages together based on time delimitation.

Term	Definition
	Once a session is terminated, no more messages can be sent on that session. Depending on the design of the MoldUDP64 system and the application, receivers may still be able to re- request messages from a terminated session. A session is considered active if it has started but has yet to be terminated.

3.4 Downstream packet

A MoldUDP64 transmitter sends "downstream" packets that are received by MoldUDP64 listeners. A MoldUDP64 packet may contain a payload of 0 or more data stream messages.
Each MoldUDP64 packet consists of a Downstream Packet Header and of a series of Message Blocks. The Message Blocks carry the actual data of the stream.

<table>
<tr><td colspan="3">MoldUDP64 Header (20 bytes)</td><td>Message length (2 bytes)</td><td>ITCH message 1</td><td>Message length (2 bytes)</td><td>ITCH message 2</td><td>…</td></tr>
<tr><td>Session (10 bytes)</td><td>Sequence number (8 bytes)</td><td>Message count (2 bytes)</td><td colspan="5">MoldUDP Packet Payload (ITCH Messages)</td></tr>
<tr><td colspan="8">UDP Payload max 64KB</td></tr>
</table>

3.4.1 Header

Name	Offset	Length	Value	Notes
Session	0	10	Alphanumeric	Indicates the session to which this packet belongs
Sequence number	10	8	Numeric	The sequence number of the first message in the packet
Message count	18	2	Numeric	The number of messages contained in this packet

Session

The id of currently active session.

Sequence Number

The Sequence Number field of the packet Header indicates the sequence number of the first message in the packet. If there is more than one message contained in a packet, any messages following the first message are implicitly numbered sequentially.

Message Count

The number of Message Blocks contained in a MoldUDP64 packet is specified by the Message Count field of the Packet Header. The maximum payload size of a Downstream Packet is determined by the sender. Note that a Message Count of zero denotes a heartbeat.

3.4.2 Message block

The first field of a Message Block is the two-byte Message Length. The remainder of the Message Block is the variable length Message Data field. The first Message Block field will always start immediately following the Header, which is 20 bytes from the beginning of the packet. Subsequent Message Blocks will begin after the last byte of the previous Message Block.

Name	Offset	Length	Value	Notes
Message length	Variable	2	Numeric	Indicates the length in bytes of the message contained in this Message Block
Message Data	Variable	Variable	Numeric	The message data

Message Length

The Message Length is an unsigned binary count representing the number of message data bytes following this Message Length field. The message length field value does not include the two bytes occupied by the message length field. The total size of the message block is the value of the message length field plus two.

Message Data

The Message Data is actual data of the message being transmitted by MoldUDP64.

3.4.3 Heartbeats

Heartbeats are sent periodically by the server so receivers can sense packet loss even during times of low traffic. Typically, these packets are transmitted once per second and contain the next expected Sequence Number. A Heartbeat packet is a MoldUDP64 packet with a Message Count of zero.

3.4.4 End of session

When the current session is complete, Downstream Packets are sent with a Message Count of 0xFFFF(hex, or 65535 in decimal) for a short while in place of Heartbeats. These Downstream Packets contain the next expected Sequence Number, just like Heartbeats. While the End of Session messages persist, re-requests may be made on the current session. This is the last chance to ensure that all messages have been received.

ⓘ In case of process or site failover, clients might or might not receive End of session.

3.5 Request packet

The Request Packet is sent to request the retransmission of a particular message or group of messages. The request packet is sent to a Re-request server. A receiver may need to send this request when it detects a sequence number gap in received messages. The response to a valid Request Packet is a standard Downstream Packet unicast back to the source of the retransmission request. This allows downstream MoldUDP64 users to read the retransmitted Downstream Packet in their multicast processing socket if the request was made from that socket (in other words, the client need only have one socket open to listen to the multicast and to process retransmissions, even though the retransmissions are not multicast).

Name	Offset	Length	Value	Notes
Session	0	10	Alphanumeric	Indicates the session to which this packet belongs
Sequence number	10	8	Numeric	First requested sequence number
Requested message count	18	2	Numeric	The number of messages requested for retransmission

Session
The id of currently active session.

Sequence Number
The Sequence Number field of the packet Header indicates the sequence number of the first message requested.

Requested Message Count
The Message Count indicates the number of messages that should be retransmitted. If the total size of the requested messages exceeds the maximum payload size of one UDP packet, only the number of messages that completely fit will be returned. Additional retransmission requests must be made for the subsequent messages if they are still desired.

4. Snapshot

Glimpse is a complement to some MoldUDP64 or SoupBinTCP business protocols, and can provide a snapshot of the current state of the order books, eliminating the need to read the stream from the first sequence number in order to build a local copy of the server's state.

Typically, Glimpse will use the same ITCH message format as its business application protocol. However, Market Announcements, System Events, Equilibrium Price, Trade Ticker, Market Statistics and News message types are not included in the Glimpse snapshot.

The *End of Snapshot ("G") Message* is the Glimpse specific message. Each Glimpse snapshot ends with a sequence number used to determine from which point the streaming data should be appended.

Glimpse uses the SoupBinTCP transport protocol to receive Glimpse requests and return Glimpse snapshots.

4.1 Glimpse Message format

The end of snapshot message returns the current ITCH sequence number to be used when connecting to the ITCH Live feed.

To maintain a real-time order display, clients should begin to process real-time market data messages beginning with the sequence number stated in this snapshot message. When switch to real-time market data, client has to use the last seconds message from snapshot until receiving the new seconds message from real-time market data.

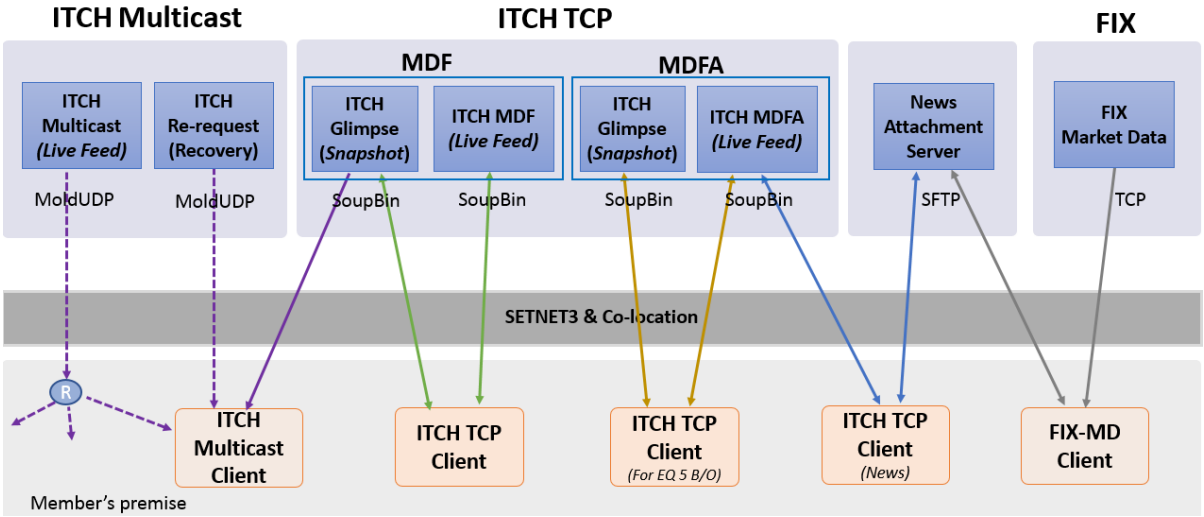
Name	Offset	Length	Value	Notes
Msg Type	0	1	"G"	G – End of Snapshot Message.
Sequence Number	1	20	Alpha	SoupBinTCP sequence number when the snapshot was taken. To be used when logging in to the market data SoupBinTCP feed. ⓘ While Glimpse is a binary feed, the SoupBinTCP uses ASCII characters to represent the sequence number.

5. ITCH Market Data System Overview

5.1 System Architecture

SET Market Data system supports market data and news data from multiple markets e.g. Equity Market (SET/mai), Derivatives Market (TFEX) and etc. However, clients will receive data based on subscription rights. For market data protocols, SET provides 2 protocols including ITCH and FIX 5.0 SP2. This document will describe only ITCH related.

ITCH is a direct market data feed protocol which is made up of a series of sequenced messages. Each message is variable in length based on the message type. The ITCH will be binary encoded over MoldUDP64 or SoupBinTCP. The messages that make up the ITCH protocol are typically delivered using a higher-level protocol that takes care of sequencing and delivery guarantees.



Market Data Gateways

SoupBinTCP is a lightweight point-to-point protocol, built on top of TCP/IP sockets that allow delivery of a set of sequenced messages from a server to a client in real-time. SoupBinTCP guarantees that the client receives each message generated by the server in sequence. For more details, please refer to SoupBinTCP protocol in above section.

SET provides 2 types of SoupBinTCP servers.

- **ITCH MDF** server: provide standard real-time price feed including 10 bid/offer levels for Equity market and 5 levels for Derivatives market. Note that there is no news data.
- **ITCH MDFA** server: provide news for both Equity and Derivatives market. The server also provide additional real-time price feed e.g. 5 bid/offer level for Equity market and LiVE market price feed.

MoldUDP64 is a light-weight networking protocol built on top of UDP that provides a mechanism for listeners to detect and re-request missed packets. Each message is explicitly sequence numbered. If a packet loss is detected by the client, it can re-request that packet from the MoldUDP64. For more details, please refer to MoldUDP64 protocol in above section.

The members who receive MoldUDP64 multicast have to configure connection at network level. The data will be disseminated from **ITCH Multicast** servers. Since MoldUDP64 messages can be lost during transmission, clients can follow recovery strategies as below.

- The clients who has redundant channel of multicast feed should recovery lost messages from redundant channel at client side first.
- If the messages are lost from all channels, clients can recovery from **ITCH Re-request** servers.
- In case that clients have lost connection, you should initial data by getting snapshot from Glimpse instead of sending re-request messages.

A separate connection for obtaining snapshot data, called **ITCH Glimpse** for market data, enabling the user to reconnect intraday and be current with the live stream.

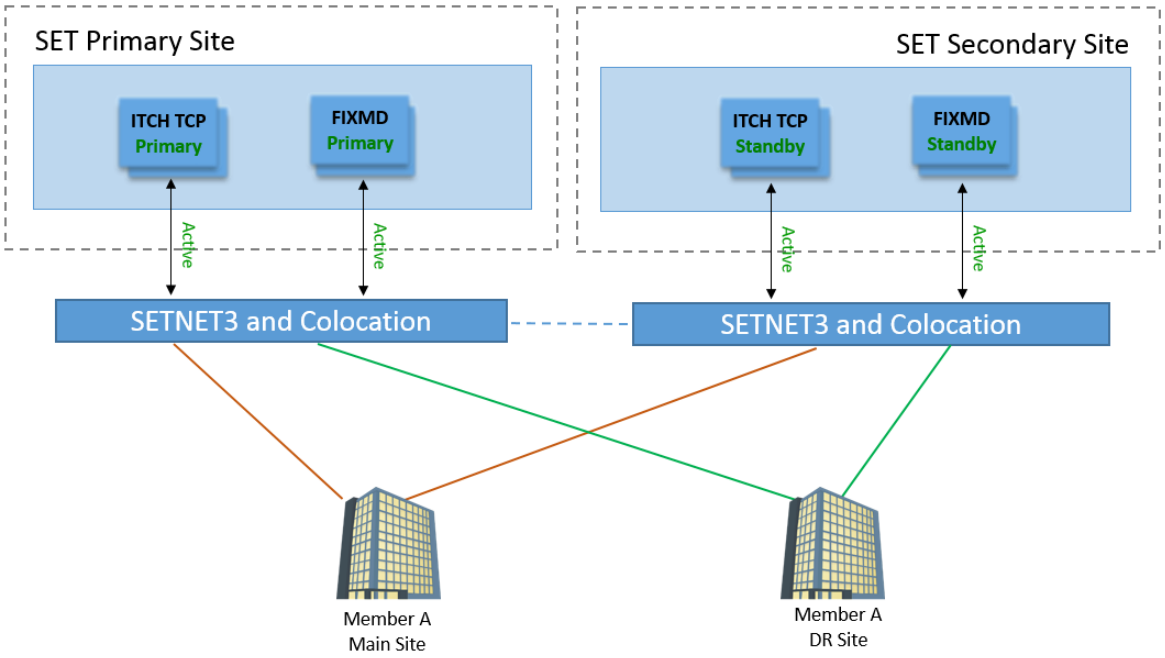
Connecting to Glimpse intraday obtains a snapshot of a subset of messages configured for this stream. For example, the Market By Price (MBP) message the snapshot includes all Price Levels available at the time of the login. The snapshot of the stream is taken at the point in time when the user connects and logs in to Glimpse. However, some message types, e.g. News, Trade Ticker are not provided via Glimpse server. For more details, please refer to Snapshot in above section.

The snapshot is tagged with a sequence number, the point at which one can listen to the live stream. The Glimpse snapshot is available in SoupBinTCP connections only.

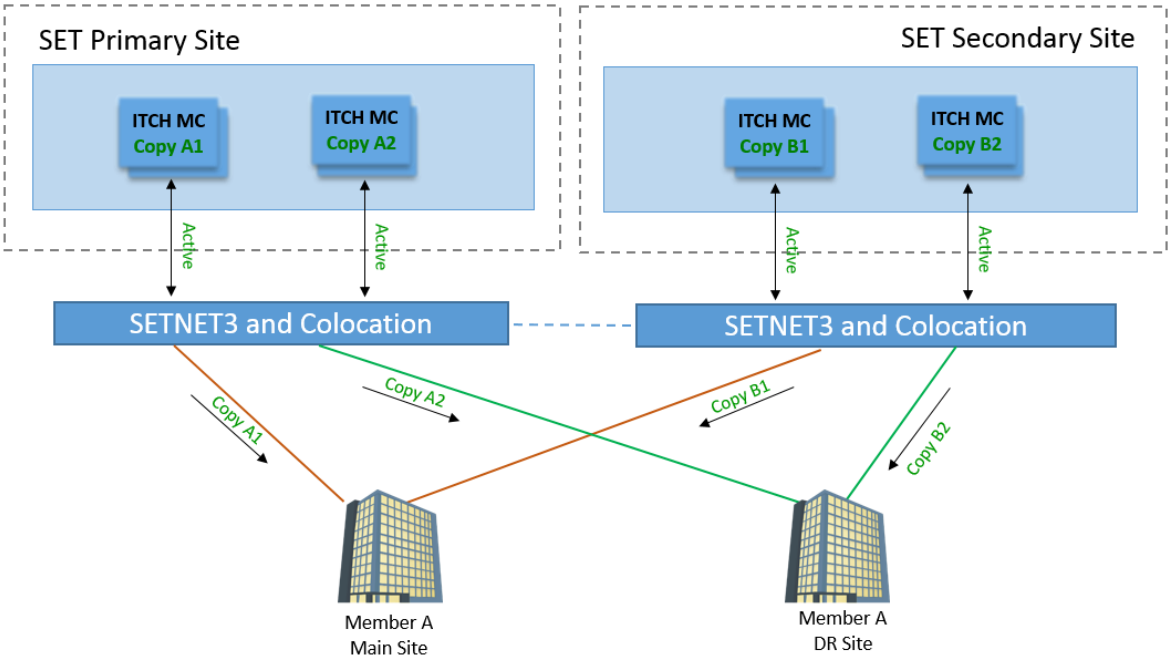
To receive the attached files of News and Financial Statements, the member can use the standard **SFTP** protocol by connecting to **News Attachment Server**.

For the infrastructure, the member can connect to the system via SET integrated network called **SETNET3** network or located in SET **Co-location**.

Moreover, market data is available from both primary site and secondary site. Members can connect to both sites at the same time using different API users. However, the secondary site should be used only for redundancy purpose. The pictures of connection are shown as below.



ITCH TCP Redundant Connection



ITCH Multicast Redundant Connection

Before clients start developing, clients should decide which protocol and gateway type clients want to receive data. Clients can see related information for "Client Connection" and "ITCH Client Development Guideline" for protocol and gateways that clients have chosen in the next section. For ITCH messages, all ITCH gateways use the same format.

Please see summarize of provided data for each gateway in below table.

Provided Data		ITCH Multicast	ITCH MDF	ITCH Glimpse	ITCH MDFA
Standard Price Feed	Equity: 10 bid/offer and other price feed	✓	✓	✓	×
	Derivatives: 5 bid/offer and other price feed				
Additional Price Feed	Equity: 5 bid/offer	×	×	✓	✓
	Other feeds e.g. LiVE market				
News	Equity News and reports	×	×	×	✓
	Derivatives News				

5.2 ITCH Client Connection

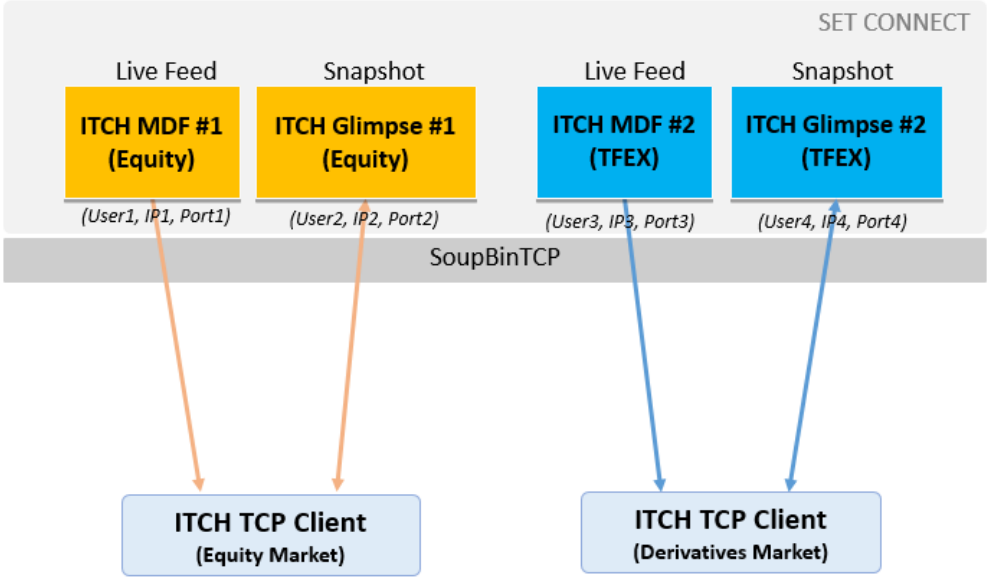
This section describes about client connections to server for each type of ITCH market data gateway. Overall concept of ITCH market data gateways are as follows.

- SET provides different gateways for each data e.g. for Equity market and Derivatives market.
- Different IP and port are defined for MDF, MDFA and Glimpse servers.
- User and password are assigned per connection. This means clients cannot use the same user login for multiple sessions. If new session is created with the same user login, the previous session will be disconnected. For ITCH Multicast and Re-request, user logins are not required.
- If a gateway fails to connect, clients can switch to other gateways. Similarly, if the primary site's gateways fail, clients can switch to the backup site's gateways. SET suggests clients to wait for a few seconds before reconnecting to server.

ⓘ Please note that in production system, SET will assign related gateways to clients for both sites (primary and backup site) in order to load balance gateways.

ⓘ Please see related IP and port for each server environment in Access New SET_CONNECT Environment document.

5.2.1 ITCH MDF Client Connection

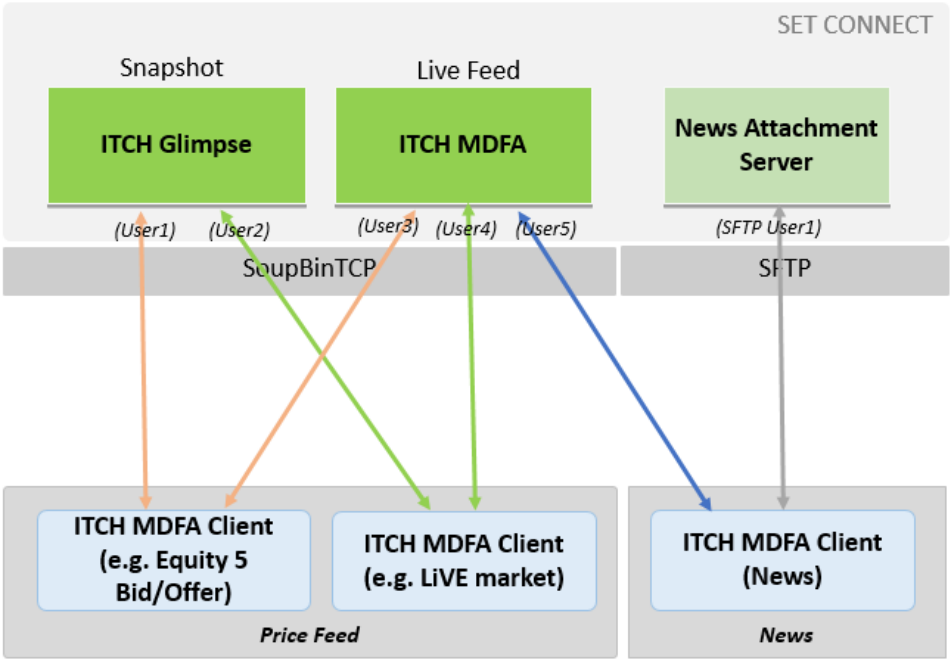


ITCH MDF Client Connection

- SET provides separated set of MDF and Glimpse gateways for Equity and Derivatives markets. Different IP and ports are defined for each gateway.
- Clients connect to related gateways with assigned user and password from SET.

ⓘ ITCH MDF provides only standard price feed. News is not available.

5.2.2 ITCH MDFA Client Connection

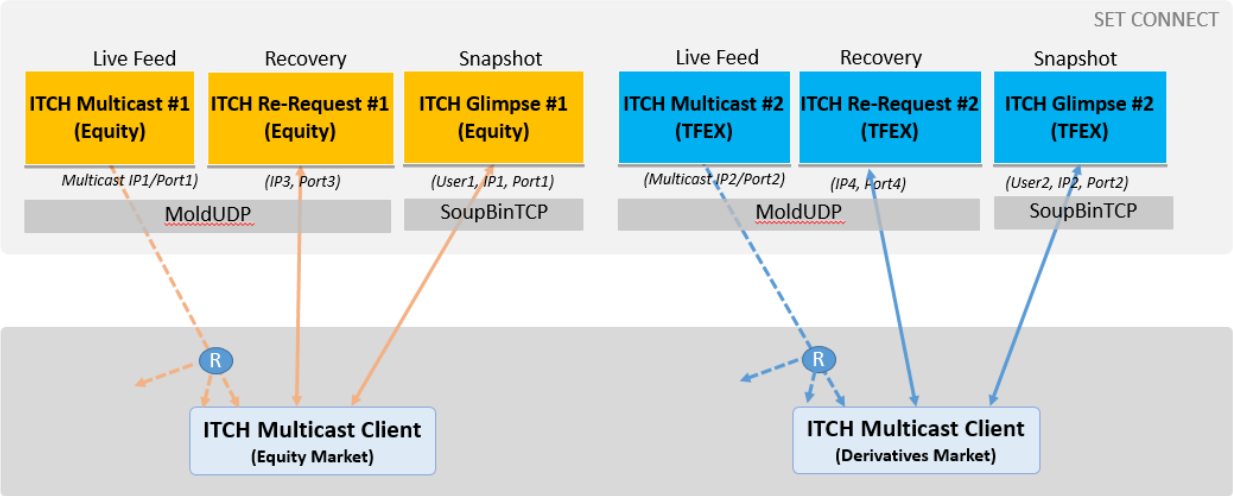


ITCH MDFA Client Connection

- SET provides ITCH MDFA gateways which can support
  - News for Equity and Derivatives market
  - Additional price feed for all markets e.g. Equity bid/offer 5 levels and LiVE market.

  However clients will receive data based on data permission.
- SET also provides ITCH Glimpse for snapshot data of additional price feeds. News client does not need to connect Glimpse server due to no snapshot of News.
- Different MDFA users are assigned for different products.
- Clients connect to related gateways with assigned user and password from SET.
- News attachment server is used for downloading news by SFTP.

5.2.3 ITCH Multicast Client Connection



ITCH Multicast Client Connection

- SET provides separated set of gateways including ITCH Multicast, ITCH Re-Request and ITCH Glimpse for Equity and Derivatives markets. Different IP and ports are defined for each gateway.
- User and password are not required for ITCH Multicast and Re-Request.
  - For ITCH Multicast, only multicast IP and port must be specified.
  - For ITCH Re-Request, only Re-Request server IP and port must be specified
- Clients connect to Glimpse with assigned user and password from SET.

ⓘ ITCH Multicast, Re-request and Glimpse provide only standard price feed. News is not available.

6. ITCH Client Development

The below section is described the development guideline for SoupBinTCP and MoldUDP64 clients. Clients may have different flow from the guideline depends on client application design.

Please ensure that you have completed ITCH Market Data System Overview section before reading this section.

ⓘ For server IP and port mentioned below, please see Access New SET-CONNECT Environment document.

6.1 ITCH MDF and MDFA for Price Feed Client (SoupBinTCP)

ITCH MDF and ITCH MDFA (Price Feed) clients can use the same flow to receive messages from related servers. The details are described below.

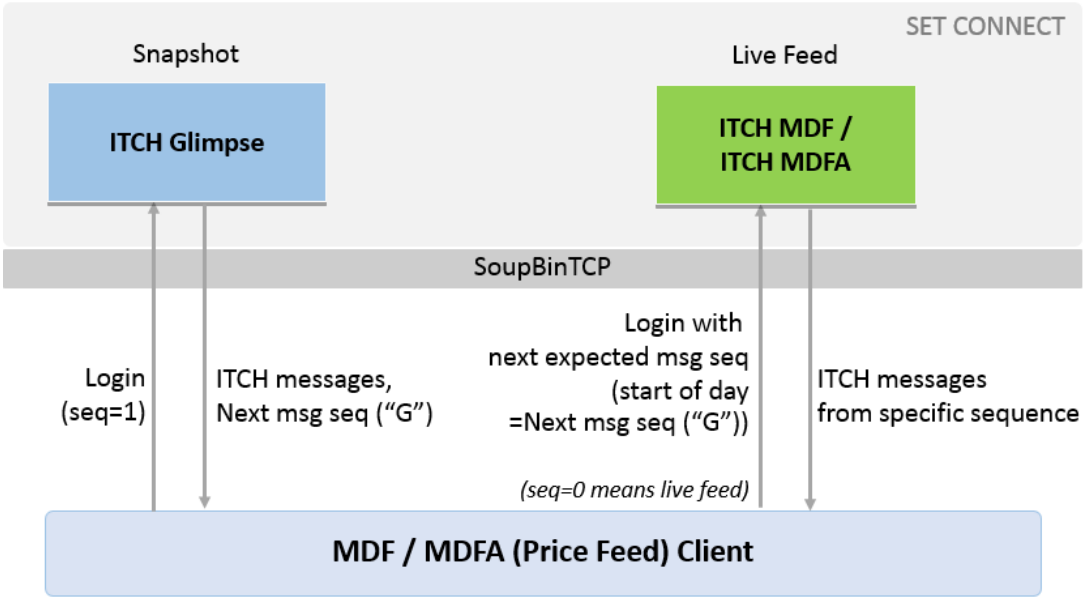
Required configuration

- ITCH MDF and ITCH Glimpse server address and port
- ITCH MDF and ITCH Glimpse username and password (assigned by SET)

or

- ITCH MDFA and ITCH Glimpse server address and port
- ITCH MDFA and ITCH Glimpse username and password (assigned by SET)

ITCH MDF and MDFA (Price Feed) Connection Flow



ITCH MDF and MDFA (Price Feed) Client Connection

6.1.1 Start of Day or Reconnect with Snapshot

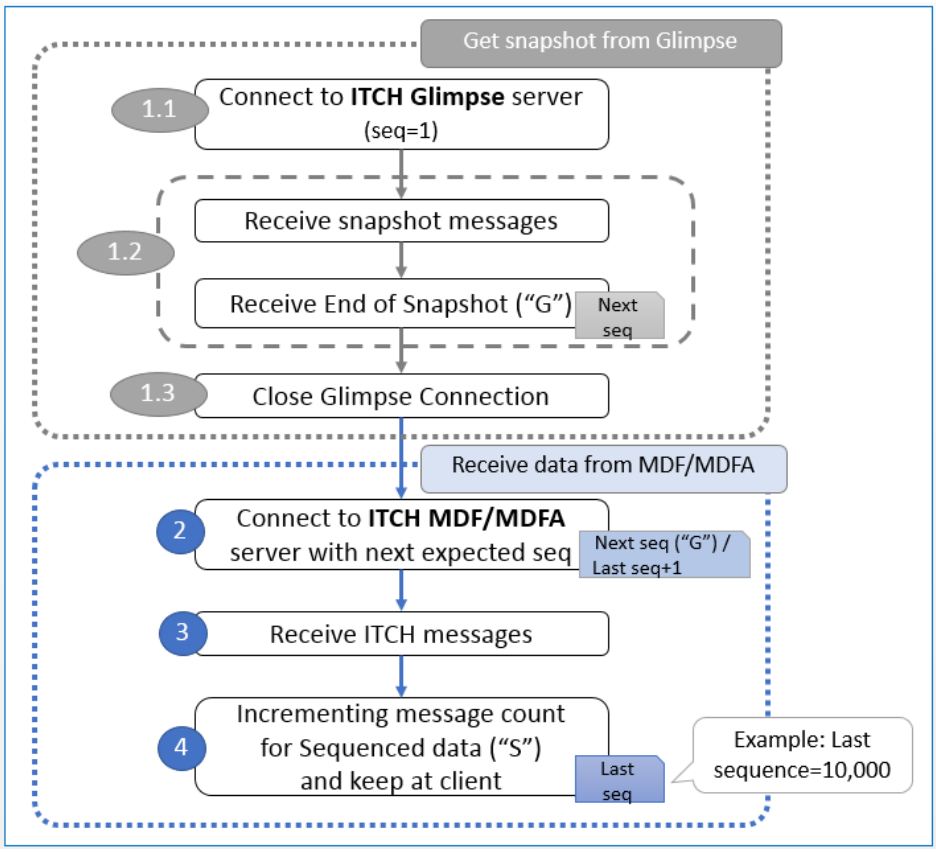
The ITCH MDF and MDFA (Price Feed) clients would typically follow the steps below when connecting in the morning or reconnect during day with snapshot data.

- First step, getting snapshot data from ITCH Glimpse server
- Then receiving live feed from ITCH MDF or ITCH MDFA server.

> ⓘ At start of the day, SET recommends clients to get snapshot data from Glimpse. If the client connects ITCH MDF or ITCH MDFA from sequence=1, the clients will receive a lot of messages for recovered individual trade statistics from the previous night session and individual market by price of long orders. This may have an impact if clients connect to the system late.

Development Guideline Step

1. Get snapshot from Glimpse
   1.1 Connect to ITCH Glimpse server with assigned user and password. In the "Login request message", state message sequence to be 1.
   1.2 Process all received snapshot messages, and store the sequence number received from End of Snapshot message ("G") as the next message sequence.
   1.3 Close Glimpse connection.
2. Connect to ITCH MDF or ITCH MDFA server with assigned user and password. In the "Login request message", specify the stored sequence number received in the Glimpse message stream.
3. Receive ITCH messages.
4. Client has to store last message sequence at client if the messages must be recovered when connection has lost. When a new message is received, increment the message sequence by 1.



ITCH MDF/MDFA (Price Feed) Receiver Flowchart

6.1.2 Reconnect

When clients have lost connection and reconnect to ITCH MDF or MDFA (Price Feed) server, clients have to specify the appropriated message sequence. There are 3 options used for reconnecting as follows. SET suggests clients to wait for a few seconds before reconnecting to server.

Option	Specific Message Sequence	Description
Snapshot and Live feed	End of Snapshot message ("G") from Glimpse ~~+1~~	Reconnecting with snapshot data if lost data is not required. Clients have to connect Glimpse before MDF/MDFA. Please go through the same steps in section 6.1.1 (Start of Day or Reconnect with Snapshot).
Recovery and Live feed	Latest stored message sequence number + 1	Reconnecting with recovery data from expected sequence number if lost messages are important e.g. trade ticker.
Live feed only	0	Reconnecting with live feed only if snapshot and recovered data are not required.

ⓘ IMPORTANT: If clients have lost connection for long time, SET recommends clients to reconnect with snapshot data instead of recovery data.

6.2 ITCH MDFA for News Client (SoupBinTCP)

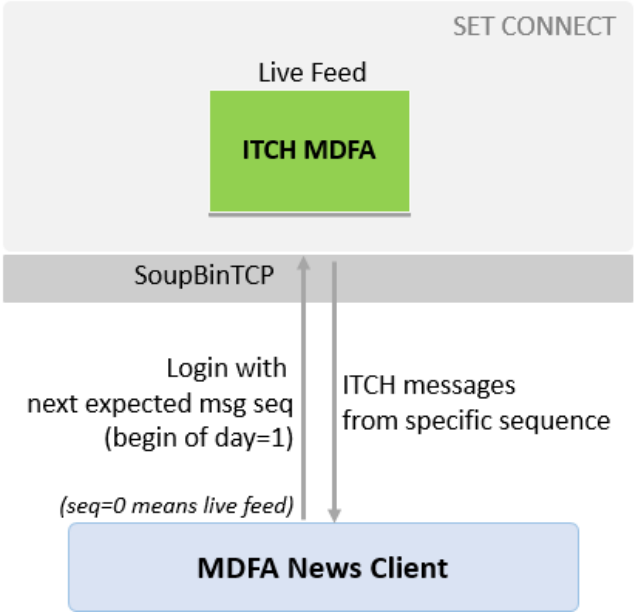
6.2.1 Start of Day or Reconnect

The ITCH MDFA (News) clients would typically follow the steps below when connecting in the morning or reconnecting during day. Please note that ITCH Glimpse is not necessary for News client since no snapshot of News.

Required configuration

- ITCH MDFA server address and port
- ITCH MDFA News username and password (assigned by SET)

ITCH MDFA (News) Connection Flow



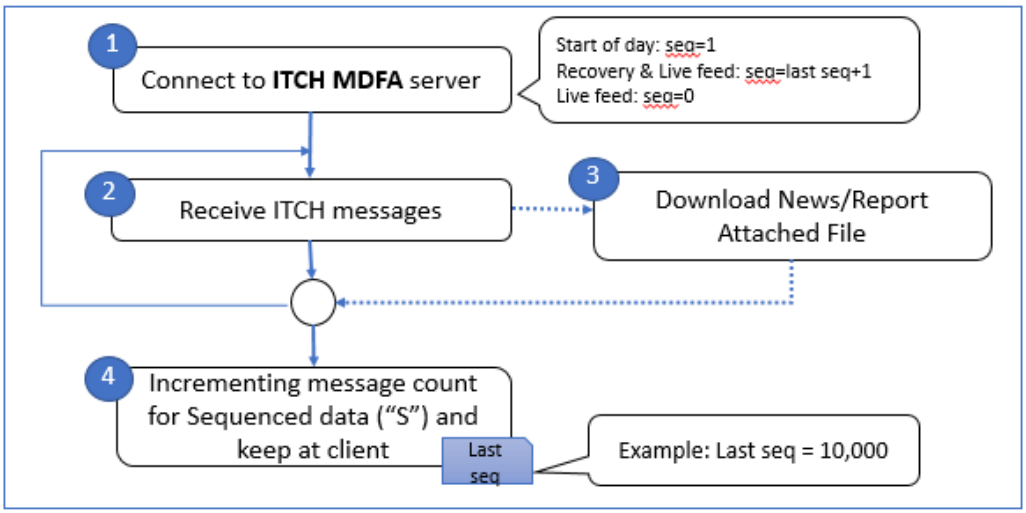
ITCH MDFA (News) Client Connection

Development Guideline Step

1. Connect to the ITCH MDFA server with sequence number. There are 3 options for specifying sequence number as follows. SET suggests clients to wait for a few seconds in case of reconnecting to server.

Option	Specific Message Sequence	Description
Start of day	1	Connect in the morning and receive messages from the beginning of the day.
Recovery and Live feed	Latest stored message sequence number + 1	Reconnecting with recovery data from expected sequence number if missing news is required.
Live feed only	0	Reconnecting with live feed only if missing news is not required.

2. Receive ITCH messages.
3. For news client, download news or report attached file by using SFTP command (as described in section 6.2.2).
4. Client has to store last message sequence at client if the messages must be recovered when connection has lost. When a new message is received, increment the message sequence by 1



ITCH MDFA (News) Receiver Flowchart

6.2.2 Download News/Report Attached file

Required configuration

- News Attachment server address and port
- News SFTP username and password (assigned by SET)

Development Guideline Step

1. When client receives News message from ITCH MDFA server, client has to get the attachment file names from such message.
2. Connect to News Attachment server with assigned user and password using SFTP protocol.
3. Get attachment files. The attachment files may be plain text file e.g. .txt file or binary file e.g. .pdf and .zip file.
   Please see example code from section 6.5.3 (SFTP Implementation).

6.3 ITCH Multicast Client (MoldUDP64)

The ITCH Multicast clients have to join multicast address to get live feed messages. Since the data will be reloaded at system startup, clients will not receive messages from the start of the day. This means that when clients join a multicast address, the gap sequence will always be found. To obtain a snapshot recovery, clients must connect to Glimpse by following the steps below. After recovery with a snapshot is completed, clients detect some lost messages and need to recover them by using Re-request server.

Required configuration

- The UDP port to listen to and the Multicast group to join
- The ITCH Re-Request servers address and port
- ITCH Glimpse server address and port
- ITCH Glimpse Username and password (assigned by SET)

To mitigate UDP packet drop, client is recommended to set parameters as follows:

1) In OS of a multicast receiving node, please configure the system parameter to increase socket receive buffer to at least size of 134,217,728 bytes (128 MB).

For Linux OS, configure the parameter in /etc/sysctl.conf:

```
net.core.rmem_max = 134217728
```

2) In OS of a multicast receiving node, please configure NIC (Network Interface Card) ring receive buffer size to the maximum value supported by the hardware.
For Linux OS, follow the following instructions:
   - To inspect the maximum size supported, run the following command and look for the pre-set maximum value for RX

     ```
     ethtool -g <interface>
     ```

   - To set ring receive buffer size

     ```
     ethtool -G <interface> rx <max_size>
     ```

   Once configured, please verify that the current value for RX is the same as the pre-set maximum value for RX.

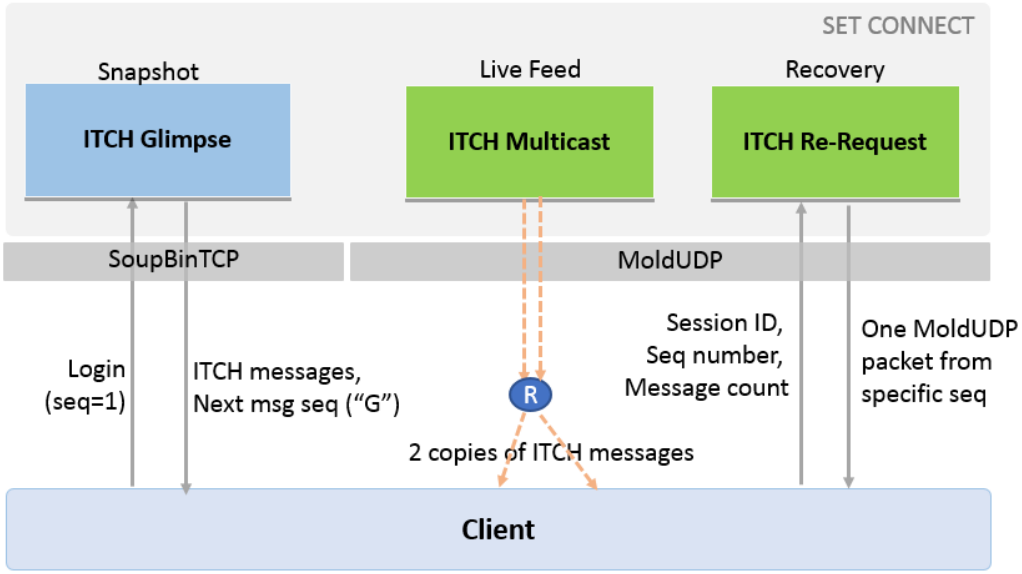
   ⓘ To make the value persistent, please consider add the line of command to /etc/rc.local or elsewhere you see fit.

   Linux says you must run `chmod +x /etc/rc.d/rc.local` to ensure that the script will be executed during boot.

3) In client application, please explicitly allocate the socket receive buffer (SO_RCVBUF) at start-up to be at least size of 134,217,728 bytes (128 MB), which synchronizes with the system parameter (net.core.rmem_max). It is also suggested that this value is parameterized and configurable for future changes.

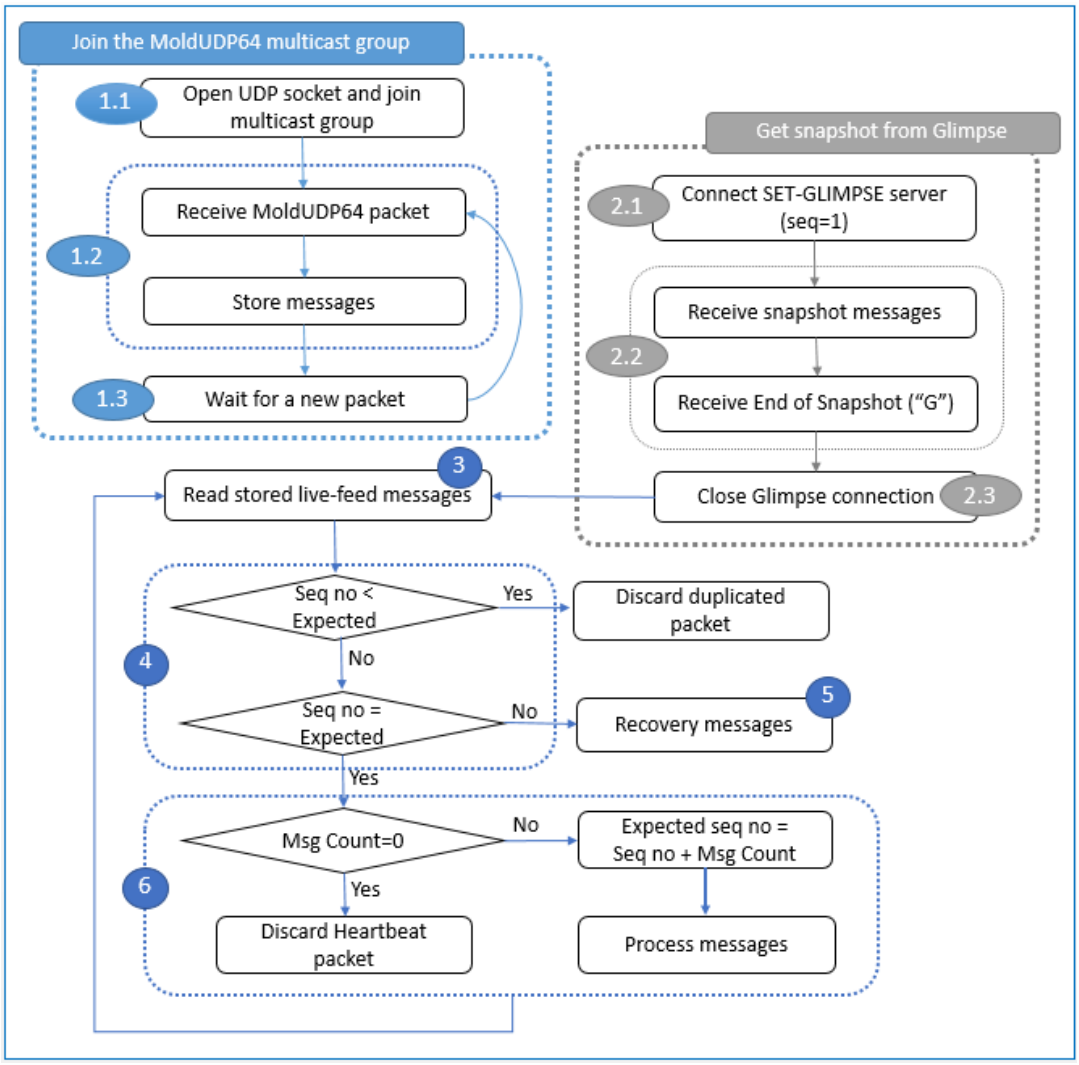
   ⓘ Client can see example of coding from Multicast Receiver Implementation section.

ITCH Multicast Connection Flow



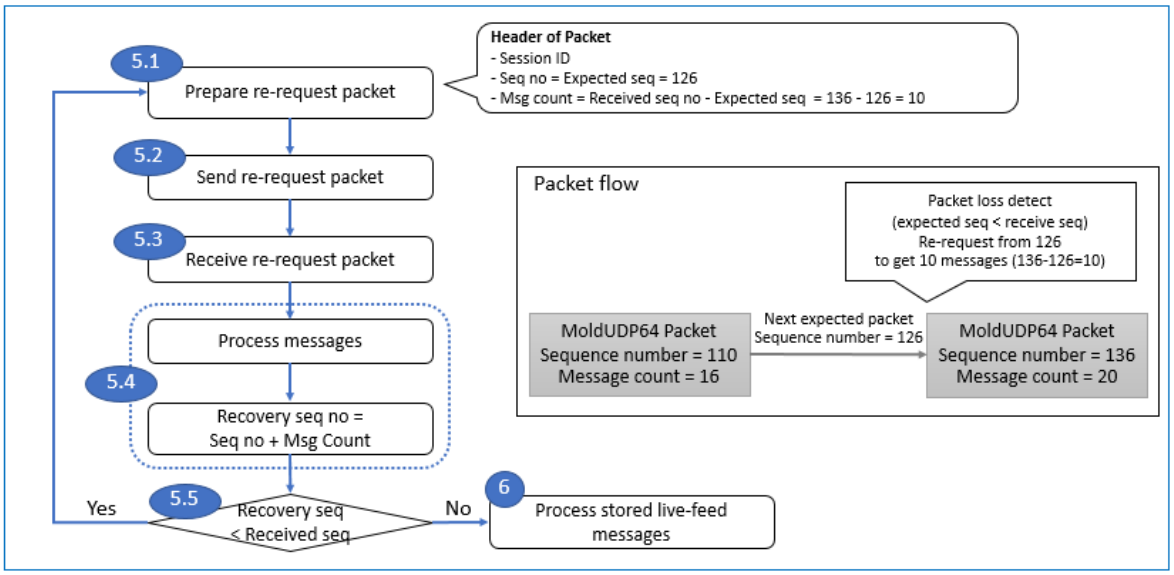
ITCH Multicast Connection Flow

Development Guideline Step



ITCH Multicast Receiver Flowchart

1. Join the MoldUDP64 multicast group and receive messages
   1.1 Open UDP socket and join the multicast group.
   1.2 Receive and store messages from the MoldUDP64 multicast group for later processing.
   1.3 Wait for a new packet and return to step 1.2.
2. Get snapshot from Glimpse
   2.1 Connect to SET-Glimpse server with assigned user and password. In the "Login request message", state message sequence to be 1.
   2.2 Process all received snapshot messages, and store the sequence number received from End of Snapshot message ("G") as the next message sequence.
   2.3 Close Glimpse connection.
3. Read multicast messages from stored messages
4. Check missing messages from received sequence number and expected sequence number.
5. If the sequence number does not match the next expected sequence number, client needs to recovery lost messages as follows:



Recovery by Re-request Flowchart

5.1 Prepare re-request packet including session ID, expected sequence number and message count. Please see detail in MoldUDP64 Protocol – Request packet section.

Example Packet flow:

- Previous packet: MoldUDP64 packet consists of sequence number = 110 and message count = 16.
- Next expected sequence number = received sequence number + message count = 110 + 16 = 126.
- Current packet: Then client receives MoldUDP64 packet with received sequence number = 136. It means that some messages are lost including message sequence number from 126 to 135 (message count = 136 – 126 = 10).

5.2 Send a re-request packet to the re-request server.

5.3 Receive re-request packet.

5.4 Process messages e.g. write last sequence number to file.

5.5 Return to step 5.

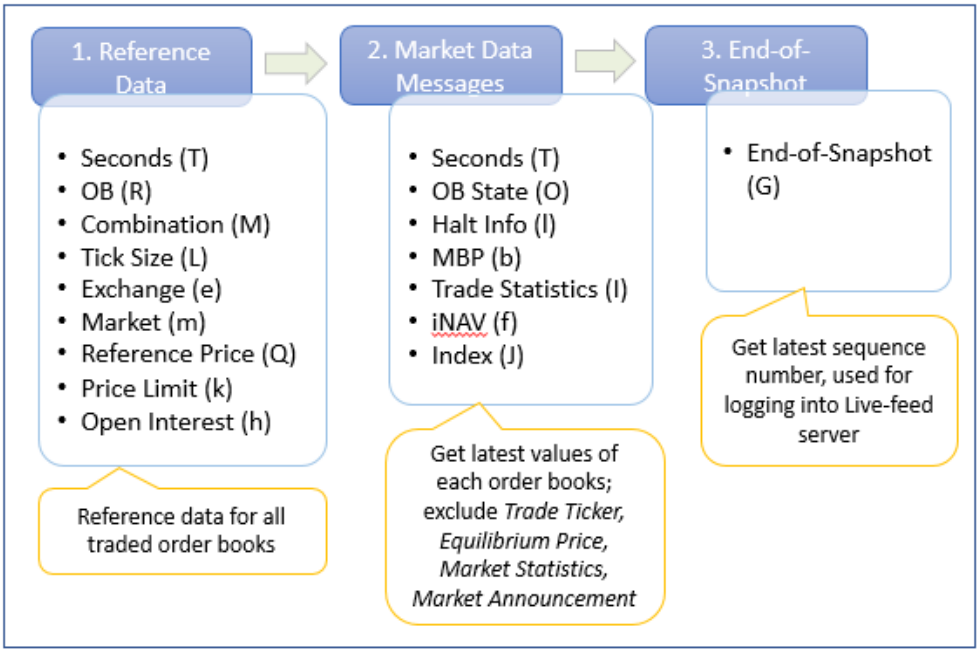
> ⓘ The requested MoldUDP messages need to fit inside one UDP packet. If total packet size is exceed, client will not receive all requested data. Client needs to send another requests until all messages can be recovered completely.
>
> ⓘ While waiting for replies from the request server, additional data may be received on the joined multicast group. Any such message must be stored locally, to be processed after the missing sequence number has been received and processed.

6. Process each of the received messages in the packet. Return to step 3 when process is finished.

6.4 ITCH Message Flow

This section shows ITCH message flow for snapshot or live feed from each gateway type. For message descriptions, please refer to "ITCH Message Format" section.

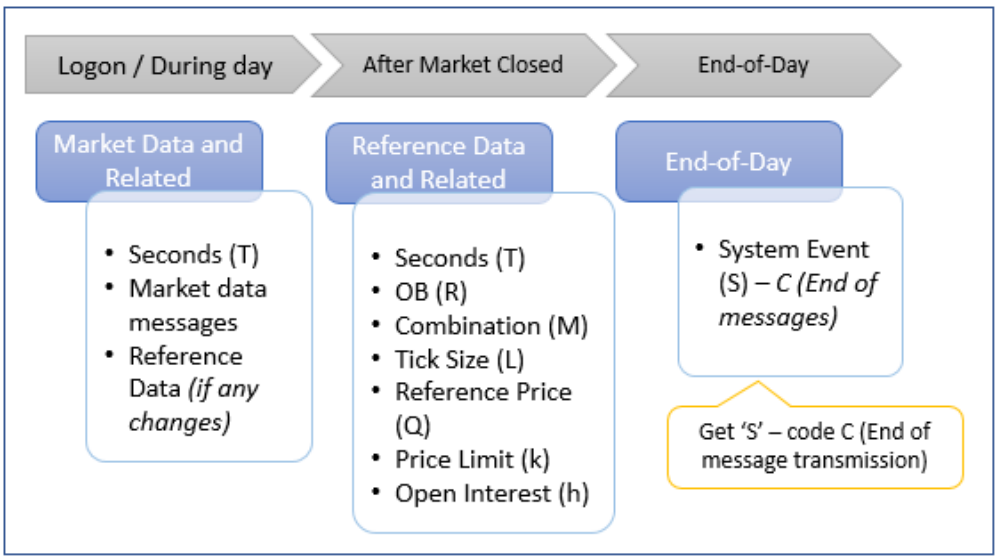
6.4.1 ITCH Glimpse Snapshot



ITCH Glimpse – Snapshot Message Flow

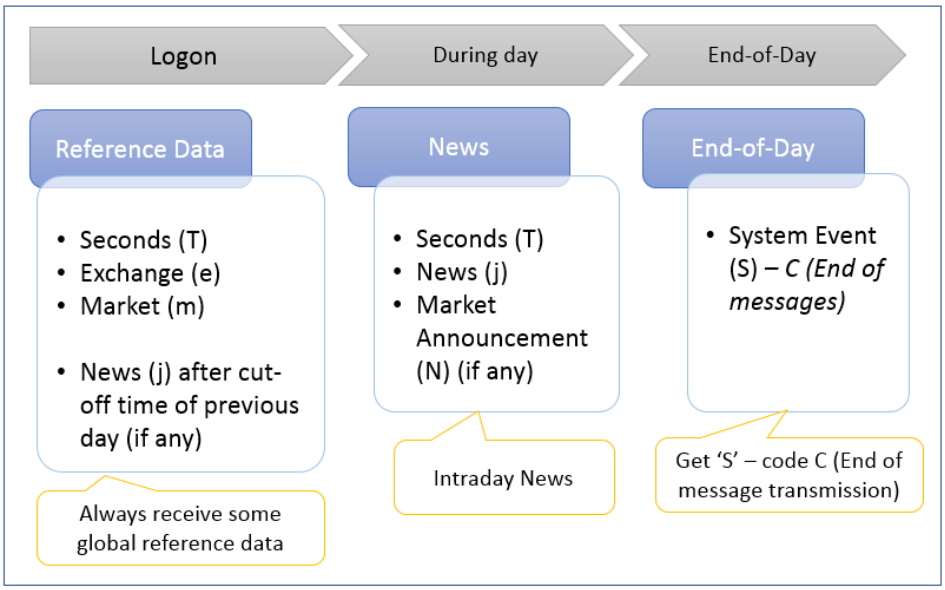
6.4.2 ITCH Live Feed

The below picture is shown live feed flow for ITCH Multicast, ITCH MDF and ITCH MDFA (Price Feed). Normally clients connect Glimpse to get snapshot before connecting ITCH server to receive live feed, therefore, clients will not receive reference data which is transmitted at start of the day.



ITCH Live Feed Message Flow

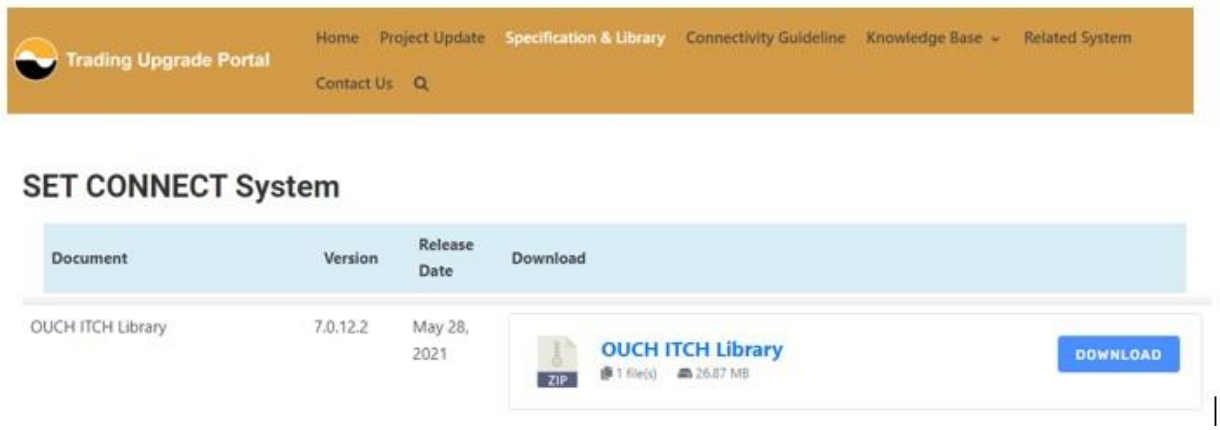
6.4.3 ITCH MDFA (News) Live Feed



ITCH MDFA (News) - Live Feed Message Flow

6.5 TCH Java Example

SET provides ITCH Java library to facilitate the development for ITCH SoubBinTCP. Members can download ITCH Java library from www.setportal.set.or.th under "Trading Upgrade Portal" section.



The library runs on **Open JDK 11.0.6**. The member has to set CLASSPATH pointing to Jar files which are inside "nasdaq" and "thirdparty" folders when development.

For ITCH MoldUDP64 protocol, the member can use ITCH Java library for message conversion. Please note that the library cannot be used for handling multicast connection.

6.5.1 SoupBinTCP Java library Implementation

Creating ITCH client

- Create ITCH client that processes the incoming ITCH messages.
- Register a message listener for receiving messages from the ITCH gateway.
- The message listener must be registered before connecting to the ITCH gateway.

```
// Prepare data user to connect a session
ConnectContextImpl connectContext = new ConnectContextImpl();
connectContext.setPassword("<PWD>");
connectContext.setUserName("<USER>");
connectContext.setHost("<HOST>");
connectContext.setPort(<PORT>);
connectContext.setSequenceNumber(<SEQUENCE NUMBER>);

// Create ITCH client
ItchClient itchClient = new com.nasdaq.ouchitch.itch.impl.ItchClient
                                        (new ConnectionListenerImpl());

// Registers a message listener for receiving messages from the ITCH gateway
itchClient.registerMessageListener(new ItchMessageListenerImpl());

int connectionId = 0;
try {
   connectionId = itchClient.connect(connectContext);
} catch (ClientException e) {
   e.printStackTrace();
}

try {
   // Wait 2 minutes to receive market data
   Thread.sleep(120000L);
} catch (InterruptedException e) {
   e.printStackTrace();
}

// Logout and close connection
try {
   itchClient.logout(connectionId);
} catch (IOException e) {
   e.printStackTrace();
}
itchClient.close() ;

// Terminate process
System.exit(0);
```

Implementation of ConnectionListener

- The implementing callback handler should as fast as possible get the data and return.
- Any long computation should be done in a separate thread.

```
// Login Success: Connection is established with the Server and the login is
accepted
@Override
public void onConnectionEstablished(Connection connection) {
   System.out.println("onConnectionEstablished: connection id = "
                                            + connection.getId());
}

// Login Rejected: In case that client sends invalid user credentials
@Override
public void onConnectionRejected(Connection connection, char reason) {
   System.out.println("onConnectionRejected: connection id = " +
```

```
                            connection.getId() + ", reason = " + reason);
    if (reason == 'A') {
        System.out.println("Not authorized.");
    }
}

// Receive messages in byte format
@Override
public void onDataReceived(Connection connection, ByteBuffer byteBuffer,
                                                    long sequenceNumber) {
  System.out.println("onDataReceived: connection id = " +
                connection.getId() + ", seq number = " + sequenceNumber
                                    + ", byte buffer = " + byteBuffer);
}

// Java library automatically sends/receives heartbeat to/from the server
@Override
public void onHeartbeat(Connection connection) {
   System.out.println("onHeartbeat: connection id = "
                                                + connection.getId());
}

@Override
public void onConnectionClosed(Connection connection) {
  System.out.println("onConnectionClosed: connection id = "
                                                + connection.getId());
}

@Override
public void onConnectionError(Connection connection, String errorMessage) {
System.out.println("onConnectionError: connection id = "
                + connection.getId() + ", error msg = "+ errorMessage);

}
```

Implementation of ItchMessageListener

- Receive all messages coming from the SoupBinTCP server
- Put all messages to buffer then use another thread to process messages.

```
BlockingQueue<Message> buffer = new LinkedBlockingQueue<>();
@Override
public void onMessage(Message message, long sequenceNumber) {
   try {
      buffer.put(message);
   } catch(Exception e) {
      //handle error
   }
}
```

Printing each field of message from Message object

```
if (message instanceof OrderBookDirectoryMessage) {
    OrderBookDirectoryMessage orderBookDirectoryMessage =
                              (OrderBookDirectoryMessage) message;
    System.out.println("Symbol: " +
                  new String(orderBookDirectoryMessage.getSymbol()));
} else if (message instanceof MarketByPriceMessage) {
    MarketByPriceMessage marketByPriceMessage =
                                  (MarketByPriceMessage) message;
    System.out.println("Orderbook ID: " +
                               marketByPriceMessage.getOrderBookId());
}
```

6.5.2 Multicast Receiver Implementation

- The SoupBinTCP Java library does not support connection handling for multicast protocol.
- The SoupBinTCP Java library can be used for message conversion by using "ItchMessageFactorySet" class.

Implementation of Multicast Receiver

- Join the MoldUDP64 multicast group and receive messages

```
String ip = "239.205.148.6";  // Example IP and port
int port = 21000;

InetAddress group = InetAddress.getByName(ip);
MulticastSocket socket = new MulticastSocket(port);

System.out.println("Join multicast group : [" + ip + ":" + port + "]");

// Set receive buffer size before joining to reduce probability of lost message
socket.setReceiveBufferSize(134217728);
socket.joinGroup(group);
byte[] recv = new byte[65536];
while (true) {
    DatagramPacket receive = new DatagramPacket(recv, recv.length);
    socket.receive(receive);
    printMulticastMessage(receive.getData(), receive.getLength());
}
```

```
public string printMulticastMessage(byte[] bytes, int length){
    byte[] hexChars = new byte[length * 3 - 1];    for(int j = 0; j < length;
++j) {
        int v = bytes[j] & 255;
        hexChars[j * 3] = HEX_ARRAY[v >>> 4];
        hexChars[j * 3 + 1] = HEX_ARRAY[v & 15];
        if (j != length - 1) {
            hexChars[j * 3 + 2] = 32;
        }
    }
    return new String(hexChars, StandardCharsets.UTF_8);
}
```

Implementation of Lost Message Recovery

```
// Session ID = 4D313236373130333531
packet.setSessionId(new byte[]{(byte) 0x4D, (byte) 0x31, (byte) 0x32, (byte)
0x36, (byte) 0x37, (byte) 0x31, (byte) 0x30, (byte) 0x33, (byte) 0x35, (byte)
0x31});

System.out.println("Re-request from " + startSeq + " to " + (endSeq - 1));
long curSeq = startSeq;
DatagramSocket rerequestSocket = new DatagramSocket();
InetAddress address = InetAddress.getByName(rerequestIp);
byte[] recv = new byte[65536];
while (curSeq < endSeq) {
    long count = endSeq - curSeq;
    System.out.println("Send rerequest package startSeq[" + curSeq +
                                    "] count[" + count + "]");
    packet.setSeq(curSeq);
    packet.setCount((short) count);
    byte[] bytes = packet.toBytes();

    DatagramPacket receive = new DatagramP  acket(bytes, bytes.length,
                                        address, rerequestPort);
    rerequestSocket.send(receive);

    // Receive re-request packet
    receive = new DatagramPacket(recv, recv.length);
    rerequestSocket.receive(receive);

    packet = printMulticastMessage(receive.getData(), receive.getLength());
    curSeq += packet.getCount();
}
rerequestSocket.close();
```

Prepare re-request packet

Send re-request packet

Receive re-request data

Parsing ITCH message from Byte Buffer to Message object by using Java library

```
// ITCH message = 545F860F87
byte[] itchMessage = new byte[] {(byte)0x54, (byte)0x5f, (byte)0x86, (byte)0x0f,
(byte)0x87};

ItchMessageFactorySet messageFactory = new ItchMessageFactorySet();
Message parse = messageFactory.parse(ByteBuffer.wrap(itchMessage));
System.out.println("msgType[" + parse.getMsgType() + "] : " + parse.getClass());
```

- ***Output from parsing message***
  msgType[84] : class genium.trading.itch42.messages.SecondsMessageImpl
- Printing each field of message from Message object. Please refer to section 6.5.1 (SoupBinTCP Java Library Implementation)

6.5.3 SFTP Implementation

SFTP is used for downloading files from News Attachment server.

Implementation of SFTP by using JSch library

```
JSch jsch = new JSch();
Session session = jsch.getSession(user, host, port);
session.setConfig("StrictHostKeyChecking", "no");
session.setPassword(password);
session.connect();
Channel channel = session.openChannel("sftp");
channel.connect();

ChannelSftp sftpChannel = (ChannelSftp) channel;
sftpChannel.get(remoteFile, localFile);

sftpChannel.exit();
session.disconnect();
```

For more information about SFTP programming in Java, the member can visit below link.

- Library: http://www.jcraft.com/jsch/
- Sample Code: http://www.jcraft.com/jsch/examples/Sftp.java

7. ITCH Message format

7.1 ITCH Message Overview

The SET CONNECT System consolidates all real-time market data to the single channel. The system supports the trading data of Equity Market (SET/mai), Derivatives Market (TFEX) and also SET News. However the client will receive such market data depending on your subscription right.

ITCH Messages on both SoupBinTCP and MoldUDP64 protocol are the same format. ITCH messages contain the following data elements.

1) **Time Messages (Timestamps)**: to identify time of messages
2) **Reference Data messages:** containing necessary information of the instruments and some related trading rules including tick size and price limit
3) **Event and State Change Messages**: containing system status, trading session and halt states of the instruments
4) **Price Messages**: containing mostly price information and other trading related reference including
   - Trading related price feeds: bid/offer, equilibrium price, trade ticker and trade statistics.
   - Trading related reference: indicative NAV, open interest, reference price e.g. closing price, settlement price.
   - Market related price feeds: index values and market statistics
5) **Market Announcements**: containing messages from the exchange and news

The list of message types transmitted via each ITCH market data gateway can be classified in the below table.

Message Type	(Standard Price Feed)		(Additional Price Feed*)		ITCH MDFA (News)
	ITCH MDF / ITCH Multicast	ITCH Glimpse	ITCH MDFA	ITCH Glimpse	
Time					
1. Seconds (T)	✓				
Reference Data					
2. Order Book Directory (R)	✓				×
3. Exchange Directory (e)	✓				✓
4. Market Directory (m)	✓				✓
5. Combination OB Leg Directory (M)	✓ (if any)				×
6. Tick Size Table (L)	✓				×
7. Price Limit (k)	✓		✓ (if any)		×

Message Type	(Standard Price Feed)		(Additional Price Feed*)		ITCH MDFA (News)
	ITCH MDF / ITCH Multicast	ITCH Glimpse	ITCH MDFA	ITCH Glimpse	
Event and State Change					
8. System Event (S)	✓	×	✓	×	✓ (Only C – End of Message)
9. Order Book State (O)	✓		✓ (if any)		×
10. Halt Information (I)	✓		✓ (if any)		×
Price Message					
11. Market by Price (b)	✓ Equity – 10 levels / TFEX – 5 levels		✓ (if any) e.g. Equity 5 levels		×
12. Equilibrium Price (Z)	✓	×	✓ (if any)	×	
13. Trade Ticker (i)		×		×	
14. Trade Statistics (I)		✓		✓ (if any)	
15. iNAV (f)		✓		✓ (if any)	
16. Index Price (J)		✓		✓ (if any)	
17. Market Statistics (g)		×		×	
18. Reference Price (Q)		✓		✓ (if any)	
19. Open Interest (h)		✓		✓ (if any)	
Market Announcement					
20. Market Announcement (N)	✓	×	✓	×	✓
21. News (j)	×				✓
Glimpse					
22. End of Snapshot (G)	×	✓	×	✓	×

ⓘ List of instruments or number of bid/offer levels that clients receive are based on member data permission and gateway type.

ⓘ Data permission for Derivatives will include **last price** of underlying feeds which traded in Equity market. Clients will also receive bid/offer information but don't have rights to distribute/display those information.

ⓘ Examples of MDFA Additional price feed are Equity bid/offer 5 levels, LiVE market price feed. However clients will receive only related messages according to data permission.

7.2 ITCH Data Type

All Integer fields are composed of binary encoded numbers.

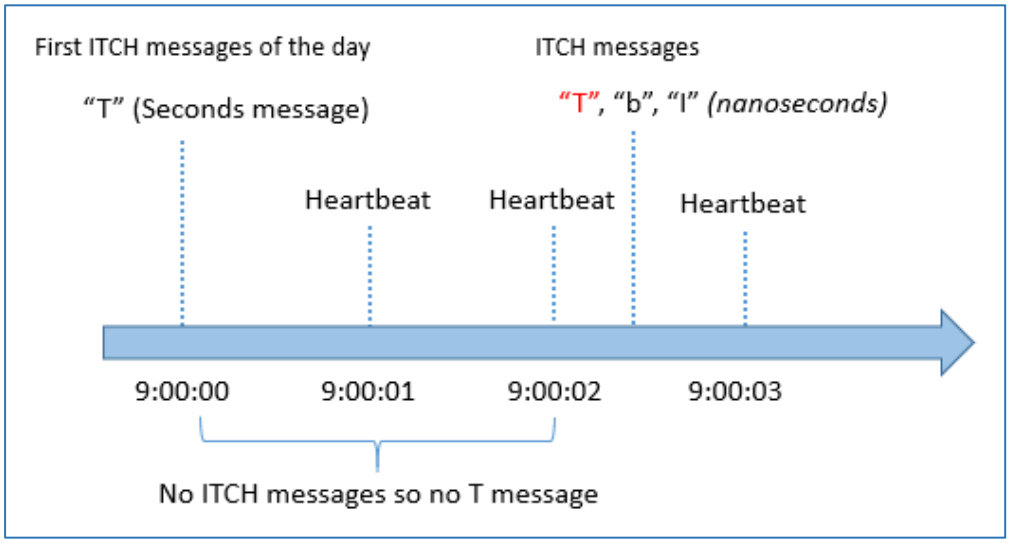
Name	Size	Notes
Numeric	1, 2, 4, 8 or 12 bytes	Unsigned big-endian binary encoded numbers. ⓘ The transport layer, SoupBinTCP or MoldUDP64, uses big-endian for its numeric values.
Alpha	variable	Left justified and padded on the right with spaces. Composed of non-control ISO 8859-1 (Latin-1) encoded bytes.
Price	4 or 8 bytes	Signed integer fields. Converted to the actual value by dividing Price with 10^number of decimals (Decimal in Price) specified in the ITCH Orderbook Directory message. ⓘ If Price is set to decimal -2 147 483 648, this means that no price is available.
Date	4 bytes	Four byte integer value derived from the Numeric data type. The decoded value represents a Date in YYYYMMDD-format.
Time	4 bytes	Four byte integer value derived from the Numeric data type. The decoded value represents a UTC Time in HHMMSS-format.
Timestamp	8 bytes	UNIX Time (number of nanoseconds since 1970-01-01 00:00:00 UTC)

7.3 Time Messages

Timestamps are separated into two pieces:

Timestamp portion	Message Type	Notes
Seconds	Standalone message	Unix time (number of seconds since 1970-01-01 00:00:00 UTC) ⓘ A Timestamp – Second message will be disseminated in second for which there is at least one payload message.
Nanoseconds	Field within individual messages	Reflects the number of nanoseconds since the most recent Timestamp-Seconds message that the payload message was generated.

Example of Time message flow



Time of message is calculated from Second (from Seconds message) and nanoseconds in the other messages which come after.

ⓘ Clients cannot synchronize time from Seconds message. Please synchronize with SETNET3 NTP (Network Time Protocol) server.

7.3.1 Seconds Messages (T)

This message is sent in second for which at least one market data message is being generated. The message contains the number of seconds since the start of 1970-01-01 00:00:00 UTC, also called Unix Time.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"T"	T - Seconds Message
Second	1	4	Numeric	Unix time (number of seconds since 1970-01-01 00:00:00 UTC)

7.4 Reference Data Messages

Reference Data Overview

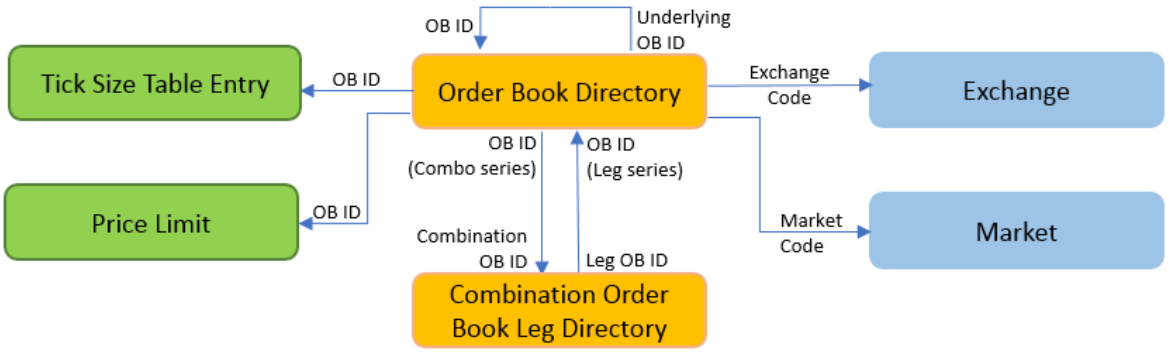
The reference data contains the necessary instrument information for the system to operate.

When the clients connect to ITCH gateway in the morning or get snapshot from GLIMSE server, the server will return several message types as follows.

- **Reference data:** Order Book Directory, Combination Order Book Leg Directory, Exchange, Market, Tick Size Table Entry and Price Limit. The relationship of each message is described below.
- **Related data:** Reference Price including closing price and settlement price, Open Interest and etc. Please see detail in each part of message type.

The clients will receive messages according to subscription rights except for Exchange and Market messages in which all data will be published. The clients have to filter only related Exchange and Market.

Reference Data Relationship



The above figure shows the relationship among necessary reference data message types.

On each day, the order book is defined in the system by a system-unique orderBookID. The clients have to use orderBookID to refer each instrument. However orderBookID can be reused if such instruments have already delisted or expired. Moreover symbol is also unique in the system so if the instrument symbol is changed, the new order book ID will be created.

All instruments will be sent through Order Book Directory messages including Equity stocks, Futures, Options, Combination series, Indices and underlying. The clients can check type of instrument from financial product field.

- For combination series, the clients can find its legs from Combination Order Book Leg Directory by joining orderBookId of combination series to combinationOrderBookId field. And find the detail of legs from joining LegOrderBookId to orderBookId in Order Book Directory.
- For underlying instrument which applicable for some products e.g. Futures, Options, Warrant and DW, the clients can find information by joining underlyingOrderBookId to orderBookId within Order Book Directory message.

  Please note that for underlying which are not traded in SET e.g. foreign index or commodities, there is information only Order Book Directory message and dummy Tick Size. The other information is not provided.

To find information of Market, the clients can join via market code field. Please note that only TH exchange is available in the system.

Moreover, to find more information about Tick Size and Price Limit, the clients can join via order book ID field in each message type

7.4.1 Order Book Directory (R)

At the start of each trading day, Order book directory messages are disseminated for all active securities, including halted securities, in the system.

ⓘ Intra-day transmissions of this message may occur when new order books are added to the system. Updates to existing order books may also be represented by intra-day Order book Directory messages.

There are multiple cases to update instruments information. The example is shown as below. For more detail, please see section 8.2 ITCH example message flow in Reference Data and Corporate Action part.

1) The new instruments of Equity and TFEX are disseminated at the different time. The details are described below.
   - New Equity instruments, which will be traded on the next trading day, will be created at corporate action process after market closed. The client can check instrument flag from OtherSign=N (New list).
   - New TFEX series will be generated in the morning and can be traded on that day. The client will receive new series when logging into the system. In future, TFEX might have instruments that can be created intraday and traded immediately.
2) For the corporate actions which the symbol of instrument will be changed e.g. change symbol of Equity, beneficial CA of TFEX underlying,
   - The status of old instruments will be set to "Issued" and the status of their combination series (if any) will be set to "Delisted". This means the instrument will not be sent out on next trading day.
   - The new instrument will be created with new order book ID with "Active" status.
   - The clients can refer to the old instrument symbol from the Originate From field.
   - The necessary information e.g. closing price, settlement price, will be carried to the new order book ID.
   - For some corporate actions, such as change symbol effective on day T, new TFEX series will be activated at the day T-1 but the underlying order book id information of these new series will be available on the day T.
3) For Equity market, the corporate action process e.g. de-list, change symbol, split par, XD, XR and etc. is activated after market closed. The clients will receive updated information after the process finishes.
   - For the instrument that will be delisted on next trading day (today is the last trading day), OtherSign flag will be set to "X". The instrument will not be sent out on next trading day.
   - Corporate action codes e.g. XD, XR, XA and split flag are set in the evening after CA process which will be effective on next trading day.

Name	Offset	Length	Value	Notes	Equity	TFEX
Msg Type	0	1	"R"	R - Order book Directory Message	✓	✓
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.	✓	✓
Order book ID	5	4	Numeric	Denotes the primary identifier of an order book. ⓘ New order book ID is created when its symbol is changed. ⓘ Expired order book IDs can be reused on next trade date onward for new instruments.	✓	✓
Symbol	9	32	Alpha	Security short name.	✓	✓
Long Name	41	32	Alpha	Long name of security.	✓	N/A
ISIN	73	12	Alpha	ISIN code identifying security. ⓘ Not applicable.	N/A	N/A
Financial Product	85	3	Alpha	Defines the instrument/combo group for order book. Possible values for Equity: CS = Common Stock CSF = Foreign Stock PS =Preferred Stock PSF = Preferred Foreign Stock W = Warrant TSR = Transferable Subscription Rights DWC = Derivative Call Warrants DWP = Derivative Put Warrants DR = Depository Receipts ETF = Exchange Traded Fund UT = Unit Trust UL = Underlying IDX = Index Possible values for TFEX: FC = Cash-Settled Future FP = Physical Delivery Futures OEC = European Call Option OEP = European Put Option WEC = Weekly European Call Option WEP = Weekly European Put	✓	✓

Name	Offset	Length	Value	Notes	Equity	TFEX
				Option CMB = Combination SPT = Spot		
Trading Currency	88	3	Alpha	Trading currency.	✓	✓
Decimals in Price	91	2	Numeric	This value defines the number of decimals used in price for this order book. Each product can have different decimal points.	✓	✓
Decimals in Nominal Value	93	2	Numeric	This value defines the number of decimals in Nominal Value. ⓘ Not applicable	N/A	N/A
Round Lot Size	95	4	Numeric	Indicates the quantity that represent a round lot for the issue	✓	✓
Nominal Value	99	8	Numeric	Nominal value. ⓘ Not applicable	N/A	N/A
Number of Legs	107	1	Numeric	Number of legs.	N/A	✓ (Combo)
Underlying Name	108	6	Alpha	Underlying name. This is used to group the instruments which in the same underlying. ⓘ Underlying name may not be same as underlying instrument e.g. underlying name of LHHOTEL is LHHOT1.	✓	✓
Underlying	114	4	Numeric	ID in numeric of Underlying Name.	✓	✓
Underlying Order book ID	118	4	Numeric	Order book ID of underlying instrument.	N/A	✓
Strike Price	122	4	Price	Strike Price of Options series. ⓘ Refer the number of decimals from “Decimals in Strike Price“ field	N/A	✓
Expiration Date	126	4	Date	Date of expiration.	N/A	✓

Name	Offset	Length	Value	Notes	Equity	TFEX
				ⓘ For combination series, this field contains original expiration date. In case that expiration is changed, please use last trade date instead (if need).		
Decimals in Strike Price	130	2	Numeric	This value defines the number of decimals used in Strike Price for this order book.	N/A	✓
Option type	132	1	Numeric	Option type. 1 = Call 2 = Put 0 = Un-defined	✓ (DW)	✓
Exchange Code	133	1	Numeric	Exchange Code ⓘ Refer to Exchange Directory message	35	35
Market Code	134	1	Numeric	Market ID. ⓘ Refer to Market Directory message	✓	✓
Price Quotation Factor	135	8	Price	The price quotation factor is used to calculate turnover for a trade. ⓘ Refer the number of decimals from "Decimals in Contract Size PQF" field	✓ (Default=1)	✓
Corporate Action Code	143	32	Alpha	Corporate Action Code. Separated by Comma. XA = Ex-All i.e. Ex-Dividend & Ex-Rights XB = Ex-Other Benefits XD = Ex-Dividend XN = Ex-Capital Return XR = Ex-Rights XT = Ex-TSR XW = Ex-Warrant ⓘ CA code is set in the evening after CA process.	✓	N/A
Notification Sign	175	32	Alpha	Notification Sign. Separated by Comma. NP = Notification Pending	✓	N/A

Name	Offset	Length	Value	Notes	Equity	TFEX
				NR = Notification Received C = Caution flag NC = Non-compliance ST = Stabilization ⓘ Notification sign can be updated intraday.		
Other Sign	207	32	Alpha	Other Sign. Separated by Comma. X = Delisted stock S = Split N = New-list stock NG = Non-Performing Group L = For long order cancellation due to CA ⓘ X (Delist), N (New list) and Split are set in the evening after CA process, indicating the instruments status on next trading day. ⓘ For the stock which Other Sign is "NG", the member should not display the stock. ⓘ L is set in morning to indicate that long orders were cancelled due to related CA. Sign will be cleared after market is closed of that day.	✓	N/A
Allow NVDR	239	1	Alpha	Allow NVDR trade A = All S = Sell Only N = None ⓘ N for TFEX series.	✓	N/A
Allow Short Sell	240	1	Alpha	Allow short sell trade Y = True N = False ⓘ N for TFEX series.	✓	N/A
Allow Short Sell On NVDR	241	1	Alpha	Allow short sell trade on NVDR Y = True N = False	✓	N/A

Name	Offset	Length	Value	Notes	Equity	TFEX
				ⓘ N for TFEX series		
Allow TTF	242	1	Alpha	Allow TTF trade (Cancelled) ⓘ Not applicable	N/A	N/A
Par Value	243	8	Price	Par Value	✓	N/A
First Trading Date	251	4	Date	First Trading Date	N/A	✓
First Trading Time	255	4	Time	First Trading Time ⓘ For the first trading date, if time is zero, the instrument will be traded on the next session. If time is specified, the instruments can be traded on that time. ⓘ After the first trading date is passed, time will be zero.	N/A	✓
Last Trading Date	259	4	Date	Last Trading Date	N/A	✓
Last Trading Time	263	4	Time	Last Trading Time	N/A	✓
Market Segment	267	4	Alpha	Market Segment e.g. SET, MAI, TXI, TXS and etc. ⓘ Not available for combination series	✓	✓
Physical Delivery	271	1	Alpha	Y = True N = False ' ' = For combination series	N/A	✓
Contract Size	272	4	Price	Contract size of the product. ⓘ Refer the number of decimals from "Decimals in Contract Size PQF" field	N/A	✓
Sector Code	276	4	Alpha	Sector code or Industry code SET: Sector Code mai: Industry Code	✓	N/A

Name	Offset	Length	Value	Notes	Equity	TFEX
				ⓘ The member can refer Industry and sector name from SET-CONNECT Guideline - Industry Sector documentss ⓘ This field is applicable only for Common stock and Foreign common stocks. It will also be sent for other financial products, but it has no meaning; please disregard it.		
Originates From	280	32	Alpha	Old security symbol of this symbol. This field is provided in case of symbol changed ⓘ This field may be cleared after market closed on next day ⓘ Not available for combination series	✓ (Optional)	✓ (Optional)
Status	312	1	Alpha	A = Active S = Suspended I = Issued (series will be deleted due to the underlying adjustment from CA) D = Delisted ⓘ Suspended status may be used on nearly expired instruments e.g. DW to indicate that it cannot be traded. ⓘ Delisted status is used for delisted instruments e.g. delisted equity stocks, combination series when their outright series are issued and spot underlying if product is terminated. ⓘ Series with Issued or Delisted status will not be sent out on the next trading day.	✓	✓
Modifier	313	2	Numeric	Modifier of the series,	N/A	✓

Name	Offset	Length	Value	Notes	Equity	TFEX
				0 = Normal series name > 0 = series name is changed when there is any corporate action occurred		
Notation Date	315	4	Date	Created date of the instrument	✓	✓
Decimals in Contract Size PQF	319	2	Numeric	This value defines the number of decimals in Contract Size and Price Quotation Factor	N/A	✓

ⓘ SET recommends that members should design your system to easily or automatically add new Market ID, Market segments, Financial Products, Corporate Action Code and Notification Sign to support future product expansion.

Example message (converted to json format)

Equity – Round lot stock


```
{
    "msgType": "R",
    "status": "A",
    "nanos": 302995424,
    "orderBookId": 65606,
    "symbol": "BBL",
    "longName": "BBL_BANGKOK BANK",
    "isin": "",
    "financialProduct": "CS",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 100,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "BBL",
    "underlying": 70,
    "underlyingOrderBookId": 0,
    "strikePrice": 0,
    "expirationDate": 0,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 11,
    "priceQuotationFactor": 1,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "A",
    "allowShortSell": "Y",
    "allowShortSellOnNvdr": "Y",
    "allowTtf": "N",
    "parValue": 1000000,
    "firstTradingDate": 0,
    "firstTradingTime": 0,
    "lastTradingDate": 0,
    "lastTradingTime": 0,
    "marketSegment": "SET",
    "physicalDelivery": "N",
    "contractSize": 1,
```


Equity – Odd Lot stock


```
{
    "msgType": "R",
    "status": "A",
    "nanos": 302995424,
    "orderBookId": 458822,
    "symbol": "BBL-O",
    "longName": "BBL_BANGKOK BANK",
    "isin": "",
    "financialProduct": "CS",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "BBL",
    "underlying": 70,
    "underlyingOrderBookId": 0,
    "strikePrice": 0,
    "expirationDate": 0,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 12,
    "priceQuotationFactor": 1,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "A",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 1000000,
    "firstTradingDate": 0,
    "firstTradingTime": 0,
    "lastTradingDate": 0,
    "lastTradingTime": 0,
    "marketSegment": "SET",
    "physicalDelivery": "N",
    "contractSize": 1,
```

```
    "sectorCode": "02",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20110101,
    "decimalsInContractSizePQF": 0
}
```


```
    "sectorCode": "02",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20110101,
    "decimalsInContractSizePQF": 0
}
```

Futures series	Combination series


```
{
    "msgType": "R",
    "status": "A",
    "nanos": 302995424,
    "orderBookId": 196678,
    "symbol": "BBLH22",
    "longName": "",
    "isin": "",
    "financialProduct": "FC",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "BBL",
    "underlying": 70,
    "underlyingOrderBookId": 65606,
    "strikePrice": 0,
    "expirationDate": 20220330,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 102,
    "priceQuotationFactor": 1000,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20210330,
    "firstTradingTime": 0,
    "lastTradingDate": 20220330,
    "lastTradingTime": 93000,
    "marketSegment": "TXS",
    "physicalDelivery": "N",
    "contractSize": 1000,
    "sectorCode": "02",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20210330,
    "decimalsInContractSizePQF": 0
}
```

```
{
    "msgType": "R",
    "status": "A",
    "nanos": 302995424,
    "orderBookId": 327750,
    "symbol": "BBLM21YU21Y",
    "longName": "",
    "isin": "",
    "financialProduct": "CMB",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 2,
    "underlyingName": "BBL",
    "underlying": 70,
    "underlyingOrderBookId": 65606,
    "strikePrice": 0,
    "expirationDate": 20210629,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 102,
    "priceQuotationFactor": 0,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20200928,
    "firstTradingTime": 0,
    "lastTradingDate": 20210629,
    "lastTradingTime": 0,
    "marketSegment": "",
    "physicalDelivery": "",
    "contractSize": 0,
    "sectorCode": "02",
    "originatesFrom": "",
    "modifier": 2,
    "notationDate": 20200626,
    "decimalsInContractSizePQF": 0
}
```

Options series	Example of special characteristics product


```
{
    "msgType": "R",
    "status": "A",
    "nanos": 855928720,
    "orderBookId": 9129541,
    "symbol": "S50N21P1200",
    "longName": "",
    "isin": "",
    "financialProduct": "OEP",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "SET50",
    "underlying": 20037,
    "underlyingOrderBookId": 85573,
    "strikePrice": 1200,
    "expirationDate": 20210729,
    "decimalsInStrikePrice": 0,
    "optionType": 2,
    "exchangeCode": 35,
    "marketCode": 101,
    "priceQuotationFactor": 200,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20210429,
    "firstTradingTime": 0,
    "lastTradingDate": 20210729,
    "lastTradingTime": 93000,
    "marketSegment": "TXI",
    "physicalDelivery": "N",
    "contractSize": 200,
    "sectorCode": "",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20210525,
    "decimalsInContractSizePQF": 0
}
```

```
{
    "msgType": "R",
    "status": "A",
    "nanos": 751404200,
    "orderBookId": 216667,
    "symbol": "GDU21",
    "longName": "",
    "isin": "",
    "financialProduct": "FP",
    "tradingCurrency": "USD",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "GD",
    "underlying": 20059,
    "underlyingOrderBookId": 85595,
    "strikePrice": 0,
    "expirationDate": 20210929,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 107,
    "priceQuotationFactor": 32148,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20210531,
    "firstTradingTime": 0,
    "lastTradingDate": 20210929,
    "lastTradingTime": 93000,
    "marketSegment": "TXD",
    "physicalDelivery": "Y",
    "contractSize": 1000000,
    "sectorCode": "",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20210601,
    "decimalsInContractSizePQF": 4
}
```

Issued series from CA	New series created from CA


```
{
    "msgType": "R",
    "status": "I",
    "nanos": 855928720,
    "orderBookId": 100011,
    "symbol": "BCPGM21",
    "longName": "",
    "isin": "",
    "financialProduct": "FC",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "BCPG",
    "underlying": 697,
    "underlyingOrderBookId": 66233,
    "strikePrice": 0,
    "expirationDate": 20210629,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 102,
    "priceQuotationFactor": 1014,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20200626,
    "firstTradingTime": 0,
    "lastTradingDate": 20210629,
    "lastTradingTime": 93000,
    "marketSegment": "TXS",
    "physicalDelivery": "N",
    "contractSize": 1000,
    "sectorCode": "9",
    "originatesFrom": "",
    "modifier": 0,
    "notationDate": 20200625,
    "decimalsInContractSizePQF": 0
}
```

```
{
    "msgType": "R",
    "status": "A",
    "nanos": 855928720,
    "orderBookId": 131769,
    "symbol": "BCPGM21X",
    "longName": "",
    "isin": "",
    "financialProduct": "FC",
    "tradingCurrency": "THB",
    "decimalsInPrice": 5,
    "decimalsInNominalValue": 0,
    "roundLotSize": 1,
    "nominalValue": 0,
    "numberOfLegs": 0,
    "underlyingName": "BCPG",
    "underlying": 697,
    "underlyingOrderBookId": 66233,
    "strikePrice": 0,
    "expirationDate": 20210629,
    "decimalsInStrikePrice": 0,
    "optionType": 0,
    "exchangeCode": 35,
    "marketCode": 102,
    "priceQuotationFactor": 1014,
    "corporateActionCode": "",
    "notificationSign": "",
    "otherSign": "",
    "allowNvdr": "N",
    "allowShortSell": "N",
    "allowShortSellOnNvdr": "N",
    "allowTtf": "N",
    "parValue": 0,
    "firstTradingDate": 20200626,
    "firstTradingTime": 0,
    "lastTradingDate": 20210629,
    "lastTradingTime": 93000,
    "marketSegment": "TXS",
    "physicalDelivery": "N",
    "contractSize": 1014,
    "sectorCode": "9",
    "originatesFrom": "BCPGM21",
    "modifier": 1,
    "notationDate": 20210612,
    "decimalsInContractSizePQF": 0
}
```

7.4.2 Exchange Directory (e)

At the start of each trading day, Exchange directory messages are disseminated for all active Exchanges in the system.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"e"	e - Exchange Directory Message
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Exchange Code	5	1	Numeric	Exchange Code
Exchange Name	6	32	Alpha	Exchange Name

ⓘ Only "The Stock Exchange of Thailand" (Exchange =35) is applicable, otherwise please discard.

7.4.3 Market Directory (m)

At the start of each trading day, Market directory messages are disseminated for all active Markets in the system.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"m"	m - Market Directory Message
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Market Code	5	1	Numeric	Market ID
Market Name	6	32	Alpha	Market Full Name
Market Description	38	5	Alpha	Market Code Example for Equity market EQSM = Equity Main Market EQSO = Equity Odd Lot Market EQIM = Equity Inter Main Market EQIO = Equity Inter Odd Lot Market Example for Derivatives market TXA = TFEX Thailand Agriculture TXC = TFEX Thailand Currency TXD = TFEX Thailand Deferred Contract TXE = TFEX Thailand Energy TXI = TFEX Thailand Equity Index TXM = TFEX Thailand Metal TXR = TFEX Thailand Interest Rate TXS = TFEX Thailand Single Stock

ⓘ Market ID is global reference data. Clients will receive all Market IDs including unrelated Market IDs. Please refer to the related Market IDs from market codes in OrderBookDirectory message.

ⓘ Please see updated Market Codes in SET CONNECT Guideline document.

7.4.4 Combination Order Book Leg Directory (M)

This message provides a mapping between a combination order book and one of the combination leg order books.

A Combination instrument is a synthetic instrument consisting of two or more real instruments. Combination instruments are set up as regular order books in which orders can be placed.

The Combination instrument and each of the leg instruments are represented by Order book Directory messages in ITCH. The Combination Order book Leg message represents a mapping between a combination order book and one of its leg order books. For each combination order book, one Combination Order book Leg message will be generated per leg that the combination consists of.

ⓘ Intra-day transmissions of this message may occur when new combination order books are added to the system. Updates to existing combination order books may also be represented by intra-day Order book Directory messages.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"M"	M - Combination Order book Directory Message
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Combination Order book ID	5	4	Numeric	Denotes the primary identifier of an order book. ⓘ Expired Order book IDs may be reused on next trade date onward for new instruments.
Leg Order book ID	9	4	Numeric	Order book ID of Leg instrument
Leg Side	13	1	Alpha	Values: B = As Defined C = Opposite
Leg Ratio	14	4	Numeric	

Example message (converted to json format)

BBLU20H21	BB3Z20H21
`{ "msgType": "M", "nanos": 692788260, "orderBookId": 2621500, "legOrderBookId": 2228284, [BBLU20] "legSide": "C", "legRatio": 1 }`	`{ "msgType": "M", "nanos": 692788260, "orderBookId": 478797, "legOrderBookId": 216653, [BB3Z20] "legSide": "B", "legRatio": 1 }`
`{ "msgType": "M", "nanos": 692788260, "orderBookId": 2621500, "legOrderBookId": 2359356, [BBLH21] "legSide": "B", "legRatio": 1 }`	`{ "msgType": "M", "nanos": 692788260, "orderBookId": 478797, "legOrderBookId": 347725, [BB3H21] "legSide": "C", "legRatio": 1 }`

7.4.5 Tick Size Table Entry (L)

This message contains information on a tick size for a price range. Together, all Tick Size messages with the same Order book ID form a complete Tick Size Table. Each Order book has a set of Tick Size Table Entries to define its tick size table.

ⓘ The number of decimals in prices are given by the Order book Directory message for this Order book.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"L"	L - Tick Size Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Order book ID	5	4	Numeric	The order book this entry belongs to.
Tick Size	9	8	Price	Tick Size for the give price range
Price From	17	4	Price	Start of price range for this entry.
Price To	21	4	Price	End of price range for this entry.

7.4.6 Price Limit Message (k)

The Price Limit Message is used to publish upper and lower price limits per orderbook.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"k"	h – Price Limit Message.

Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
Upper limit	9	4	Price	Upper limit
Lower limit	13	4	Price	Lower limit

ⓘ If upper and lower limit are sent out as -2 147 483 648 (min integer value), it means no price limit. The clients must use 0.01 for lower limit and use the truncated decimal of absolute value of (min integer value / price decimal from OrderBookDirectory) for upper limit.

For example, if price decimal is 5, upper limit will be 21,474. If price decimal is 4, upper limit will be 214,748.

7.5 Event and State Change Messages

7.5.1 System Event Message (S)

The system event message type is used to signal a market or data feed handler event.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"S"	S – System Event Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Event Code	5	1	Alpha	The system supports the following event codes on a daily basis: “O” = Start of Messages. ⓘ Outside of time stamp messages, the start of day message is the first message sent in any trading day. “C” = End of Messages. ⓘ This is always the last message sent in any trading day.

ⓘ This message is not available for ITCH Glimpse.

ⓘ Normally clients connect Glimpse to get snapshot data before receiving ITCH live feed, therefore, clients will not receive Event Code = O which is the first transmitted message of the day.

7.5.2 Order Book State Message (O)

The Order book state message relays information on state changes. The message is sent to indicate active state of the instruments including trading session and halt status depending on which state has higher priority.

Each product can have different trading session time, so trading session can be different in the

same Market ID. To support future functionality, members might design to support order book state per instruments.

Please see message flow from ITCH example message flow section.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"O"	O – Order Book State Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	5	4	Numeric	Order book identifier
State Name	9	20	Alpha	Name of active state of the order book. Possible values are shown as below table.

Possible values of State Name

There are 3 groups of state names; normal trading session, halt/circuit breaker related and internal operation states. The state names are separately defined for Equity (SET/mai) and Derivatives (TFEX) markets. The state name ending with "_E" is for Equity market and ending with "_D" is for Derivatives market. ~~Please note that the member can discard the State ending with "_I" which is used for Index instruments~~. Please note that the states of index orderbooks, such as PRE-OPEN1_I, are for the Exchange internal use and can be changed without prior notice. The members must discard them.

SET recommends members to display state name as per business term. Please refer recommended display name and time of each session from SET CONNECT Guideline document.

State Name	Example for Equity (SET/mai)	Example for Derivatives (TFEX)
Normal Trading Session	- CLOSE_E (First and last session of the day) - PRE-OPEN1_E - PRE-OPEN_E (for non-intermission product) - OPEN1_E - OPEN_E (for non-intermission product) - INTERMISSION_E - PRE-OPEN2_E - OPEN2_E - PRE-CLOSE_E - OFF-HOUR_E - MARKETCLOSE_E **Future Used** - FREEZE1_E - FREEZE2_E - FREEZE3_E ⓘ FREEZE states are inserted between PRE-OPEN and OPEN state.	- CLOSE_D (First and last session of the day) - PRE-MORNING_D - MORNING_D - DAY_D (for non-intermission product) - TRANSITION1_D - INTERMISSION_D - PRE-AFTERNOON_D - AFTERNOON_D - PRE-SETTLEMENT_D - TRANSITION2_D - SETTLEMENT_D - INTERMISSION2_D - INTERMISSION3_D - PRE-NIGHT_D - NIGHT_D - TRANSITION3_D - MARKETCLOSE_D - DAYCLOSE_D
Halt/Circuit Breaker Related State	- HALT_E - SUSPEND_E - PAUSE_E - CIRCUIT_BREAKER_E - FULLHALT_E (Not allow for both new and cancel orders) - PRE-OPEN_E (after lifting Market Full Halt or Circuit Breaker) **Future Used** - PRE-OPEN_CB_E (after Order Book Circuit Breaker is triggered) - EXPIRED_E	- HALT_D - SUSPEND_D - PAUSE_D - CIRCUIT_BREAKER_D - FULLHALT_D (Not allow for both new and cancel orders) - PRE-OPEN_D (after lifting Market Full Halt or Circuit Breaker) - PRE-OPEN_CB_D (after Price Break is triggered) - EXPIRED_D (when last trading time of the series)
Operation State	- STARTUP_E (initial operation tasks) - SAVECLOSING_E (to save closing price of the instruments) - RESET_STAT_E (to reset trade statistics before CLOSE system)	- To generate new series ○ SERIES_GEN_DAY_D ○ COMBO_GEN_DAY_D ○ SERIES_GEN_NIGHT_D - RESET_STAT_D (to reset trade statistics for Day markets before CLOSE system)

ⓘ Please see example of messages transmission in section The Example Message Flow - Trading session and instrument state example.

ⓘ Please see updated State Names and trading schedule from SET CONNECT Guideline document.

ⓘ Please prepare your systems to support future used states as well.

7.5.3 Halt Information Message (I)

The Halt Information message relays information on instrument state changes. If the series is halted, Halt information messages are sent out for both outright and combination series. This message is also sent when the state of instrument differs from its market state.

Please see message flow from ITCH example message flow section.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"I"	I – Halt Information Message. ⓘ L in small letter
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Order Book ID	5	4	Numeric	Order book identifier
Instrument State	9	20	Alpha	Name of the halt and related state, or state of instrument ⓘ When halt is lifted and return to normal trading session, the value is sent out as blank in order to clear the instrument state.

Possible values of Instrument State

Halt states can be performed for the whole market or by instruments based on business cases.

- If there is a change in halt status or trading session in the instrument level, Halt Information message will be sent out.
- If there is a change in halt status or trading session in market level, Halt Information message is not sent out.
  For example, Halt information will not be sent out for Circuit Breaker which is normally halted for the whole market. Only Order Book State is sent out for Circuit Breaker.

Possible values of instrument states can be listed in the below table.

Instrument State	Equity	TFEX
Related Halt State	- HALT_E - SUSPEND_E - PAUSE_E - FULLHALT_E (Not allow for both new and cancel orders) - PRE-OPEN_E (If there is auction period after the halt is lifted) - ‘ ‘ (blank when state is cleared) **Future Used** - PRE-OPEN_CB_E (after Order Book Circuit Breaker is triggered) - EXPIRED_E	- HALT_D - SUSPEND_D - PAUSE_D - FULLHALT_D (Not allow for both new and cancel orders) - PRE-OPEN_D (If there is auction period after the halt is lifted) - PRE-OPEN_CB_D (after Price Break is triggered) - EXPIRED_D (when last trading time of the series) - ‘ ‘ (blank when state is cleared)
Instrument State	- Normal trading sessions and operation states can be set as instrument state (refer to possible values of State Name in Order Book State message)	

ⓘ Please see example of messages transmission in section 8.2.2 (Trading session and instrument state example).

ⓘ Please see updated halt states from SET CONNECT Guideline document.

ⓘ Please prepare your systems to support future used states as well.

7.6 Price Messages

7.6.1 Market by Price (MBP) Incremental Message (b)

The Market by Price Message is used to publish order book prices per price level.

- Each market can have different maximum levels of bid and offer, for example, 5 levels for Derivatives market and 10 levels for Equity market.
- Multiple update actions can be in one MBP message. Clients must process the received level items in the same order as received, and in the very order the update items appear within the message. During processing one MBP message with multiple update actions, the price levels can temporarily be beyond the Maximum Level. Once after all update items of an MBP message have been processed, clients should drop price levels beyond the Maximum Level, if such exists.
- This message has variable length. One message can contain multiple levels of both bid and ask sides.
- The message sends only the changed levels by sending levelUpdateAction field to be New, Change or Delete. The clients have to maintain bid and offer at client side by using side and level fields as keys. Please see examples of how to maintain the local order book view in section 8.3.

Level Update Action	How to handle
New	• Add record at specific side and level • Lower levels are shifted down • Remove the lowest level which is out of price level (if any)
Change	• Update price and quantity at specific side and level
Delete	• Delete records of specific side, level and number of records which have to be deleted is specified in "number of deletes" • Lower levels are shifted upwards

Please see message flow from ITCH example message flow section.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"b"	b – Market by Price Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Order book ID	5	4	Numeric	
Maximum Level	9	1	Numeric	The maximum price level (for example 5, 10). Price levels > Maximum Level are not applicable and shall be deleted by a consumer of the Market by Price Message.
Number of Level items	10	1	Numeric	The number of times the item fields below are repeated. Number of Level items is >= 0. If value is 0 then no Array of Items will be used.

Name	Offset	Length	Value	Notes
Array of items	11+n* 16	16 * Number of Level items	Alpha	Array of Price Update items. The item counter n for the Offset is counted from 0 up to Number of Level items -1 when Number of Level items > 0.
Level Update Action	0	1	Alpha	Type of market update action. N = New C = Change D = Delete From
Side	1	1	Alpha	B = Bid A = Ask
Level	2	1	Numeric	The numeric order of the price level, where "1" is the first price level.
Price	3	4	Price	Price for the Level. Value is set to -2 147 483 648 (min integer) for 2 cases: • For Market Price (MP/ATO/ATC) which price is not specified • If "Level Update Action" is 'D', price is not used for a Price Delete Item
Quantity	7	8	Numeric	Quantity for the Level. Value is 0 and not used for a Price Delete Item (when update action is 'D' = Delete From).
Number Of Deletes	15	1	Numeric	Number of Levels to Delete. Value is > 0 and used for a Price Delete Item (when update action is 'D' = Delete From). Value is 0 and not used for a Price Update Item (when update action is 'N' or 'C').

Example message (converted to json format)

Insert ask price @ level 1	**Insert bid price @ level 2**

```
{
   "msgType": "b",
   "nanos": 413775312,
   "orderBookId": 65596,
   "maximumLevel": 5,
   "items": [
      {
         "levelUpdateAction": N,
         "side": A,
         "level": 1,
         "price": 11500000,
         "quantity": 100,
         "numberOfDeletes": 0
      }
   ]
}
```

```
{
   "msgType": "b",
   "nanos": 169745866,
   "orderBookId": 65596,
   "maximumLevel": 5,
   "items": [
      {
         "levelUpdateAction": N,
         "side": B,
         "level": 2,
         "price": 10500000,
         "quantity": 200,
         "numberOfDeletes": 0
      }
   ]
}
```


Update ask quantity @ level 3


```
{
   "msgType": "b",
   "nanos": 754965202,
   "orderBookId": 65596,
   "maximumLevel": 5,
   "items": [
      {
         "levelUpdateAction": C,
         "side": A,
         "level": 3,
         "price": 12500000,
         "quantity": 300,
         "numberOfDeletes": 0
      }
   ]
}
```


Delete bid price 2 levels (from level 1) and insert bid price @ level 4


```
{
   "msgType": "b",
   "nanos": 625801935,
   "orderBookId": 65596,
   "maximumLevel": 5,
   "items": [
      {
         "levelUpdateAction": D,
         "side": B,
         "level": 1,
         "price": -2147483648,
         "quantity": 0,
         "numberOfDeletes": 2
      },
      {
         "levelUpdateAction": N,
         "side": B,
         "level": 1,
         "price": 10500000,
         "quantity": 100,
         "numberOfDeletes": 0
      }
   ]
}
```


7.6.2 Equilibrium Price Message (Z)

This message is used during auction period. The message provides the changes in equilibrium price. When the auction deal is matched, the equilibrium price is cleared by sending out as -2 147 483 648 (min integer). This means that the equilibrium price will not be carried over to the next session. Please see message flow from ITCH example message flow section.

Clients can obtain the following information from the provided fields:

- Matched quantity = minimum value of bid and ask quantity
- Imbalance = Bid quantity – Ask quantity. If the imbalance is negative, the surplus is on the Ask side.

There are some special cases related to market order (ATO/ATC/MP) that equilibrium price can be less than lower price limit or greater than upper price limit for 1 tick size. This means that the

auction trade price may be outside of price range limit (e.g. lower price limit - 1 tick size or upper price limit + 1 tick size).

Name	Offset	Length	Value	Notes
Msg Type	0	1	"Z"	Z – Equilibrium Price Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Order book ID	5	4	Numeric	
Bid Quantity	9	8	Numeric	Quantity at equilibrium price on the bid side.
Ask Quantity	17	8	Numeric	Quantity at equilibrium price on the ask side.
Equilibrium Price	25	4	Price	Equilibrium Price.
Best Bid Price	29	4	Price	Best Bid Price of Limit order. ⓘ Please ignore, inconsistency with MBP
Best Ask Price	33	4	Price	Best Ask Price of Limit order. ⓘ Please ignore, inconsistency with MBP
Best Bid Quantity	37	8	Numeric	Best Bid Quantity of Limit order. ⓘ Please ignore, inconsistency with MBP
Best Ask Quantity	45	8	Numeric	Best Ask Quantity of Limit order. ⓘ Please ignore, inconsistency with MBP

ⓘ If Price is set to decimal -2 147 483 648 (min integer value), this means that no price is available.

7.6.3 Trade Ticker Message (i)

The purpose of Trade Ticker Message is to publish Trade Ticker of the instruments.

Deal source field specifies type of deal as follows:

- **Auction:** deals which matched with auction price when Pre-open state changes to Open state. The system aggregates those deals to one trade ticker message for each order book.
- **Auto Match:** deals which automatically match by the system
- **Trade Report:** deals from the negotiation outside the system before entering to the system
- **Combo to Combo:** deals which are generated from matching between combination orders. Trade tickers of both combination series and its outright series are set as Combo-to-Combo deal source.

Please note that trading prices may be higher or lower than the price limit under certain circumstances. Such trading prices are also reflected in trade statistics.

- Auction trade price, as mentioned in Equilibrium Price message.
- Matched price of outright and combination series which deal source is Combo to Combo deal source due to combination orders matching behavior. SET recommends members to display Combo to Combo flag to indicate that price can exceed price limit.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"i"	i – Trade Ticker Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
Deal ID	9	8	Numeric	ID of deal ⓘ Please display as 16 char hexadecimal on screen
Deal Source	17	1	Numeric	1 = Auction 2 = Auto match 3 = Trade Report 4 = Combo to Combo ⓘ Combo to Combo is applicable only for Derivatives market
Price	18	4	Price	Trade price
Quantity	22	8	Numeric	Trade quantity

Name	Offset	Length	Value	Notes
Deal Date Time	30	8	Timestamp	Deal execution time ⓘ For cancelled deal, this field retains original deal execution time.
Action	38	1	Numeric	1 = New 3 = Canceled
Aggressor	39	1	Alpha	B = Bid A = Ask N = None ⓘ None is used for no aggressor side deal e.g. trade report, auction deal and combo-match-outright deal (of combination series record)
Trade Report Code	40	2	Numeric	Code of trade report type Valid values for Equity market: - 0 = For auction deal and auto-match - 10-19 = TR_BIGLOT related (Deal which has a criteria of big lot transaction) - 20-29 = TR_BUYIN related (Deal for Buy-in purpose) - 30-39 = TR_OFFHOUR related (Deal during Run-off period) - 40-49 = TR_FOREIGN related (Deal from foreign stocks which is not TR_BIGLOT) - 50-99 = Other trade report types Valid values for Derivatives market: - 0 = For auction deal and auto-match - 100 - 199 : Normal Trade Report - 200 - 229 : EFP (Exchange of Futures for Physicals) - 230 - 249 : Other trade report types ⓘ The members can see list of trade report types from the "SET CONNECT Guideline TradeReport" document.

7.6.4 Trade Statistics Message (I)

The Trade Statistics Message is used to publish Trade Statistics per Orderbook ID.

- For series traded in TFEX Night markets, Trade Statistics of previous day will be carried to next trading day. This means trade statistics on day T is calculated from trading data of Night session on date T-1 and Day session on day T. When the clients connect to the system on day T, you will receive carried trade statistics of Night session of day T-1.
- Resetting Trade statistics flow
  - Trade statistics of TFEX Night market will be reset to zero when entering INTERMISSION3_D session (before Pre-Night session).
  - Trade statistics of Equity market and TFEX Day market will be reset as zero when entering RESET_STAT session (please refer to Order Book State Message).

  The members can discard messages after such session if you don't want trading statistics to be cleared. However you have to ensure that the statistics is reset properly in the next trading day.
- There is no trade statistics message of combination series. Only trade ticker is provided.
- Price of outright series which is generated from system to support combo-to-combo deal is not reflected in trade statistics messages of outright series. However, volume is still included.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"I"	I – Trade Statistics Message. ⓘ i in capital letter
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	ⓘ Combination series is not provided trading statistics.
Open Price	9	4	Price	Open price of current day
High Price	13	4	Price	Highest price of current day
Low Price	17	4	Price	Lowest price of current day
Last Price	21	4	Price	Last auto-match trade price (including auction price)
Last Auction Price	25	4	Price	Last auction price
Turnover Quantity	29	8	Numeric	Calculated as sum of Traded Quantity Includes auction, auto-match and trade report turnover value

Name	Offset	Length	Value	Notes
Reported Turnover Quantity	37	8	Numeric	Calculated as sum of Traded Quantity of trade report only
Turnover Value	45	8	Price	Calculated as sum of Traded Price * Traded Quantity Includes auction, auto-match and trade report turnover value ⓘ TFEX turnover value is not calculated (sending 0)
Reported Turnover Value	53	8	Price	Calculated as sum of Traded Price * Traded Quantity of trade report only ⓘ TFEX turnover value is not calculated (sending 0)
Average Price	61	4	Price	Average price of current day
Total Number Of Trades	65	8	Numeric	The total number of trades today. Including trades from all trading types, e.g. auction, auto-match and trade report.

ⓘ If Price is set to decimal -2 147 483 648, this means that no price is available.

Example message (converted to json format)

Trade Statistics – AOM and Trade Report (include all prices)


```
{
   "msgType": "I",
   "nanos": 106001694,
   "orderBookId": 65537,
   "openPrice": 410000,
   "highPrice": 410000,
   "lowPrice": 400000,
   "lastPrice": 400000,
   "lastAuctionPrice": 410000,
   "turnOverQuantity": 6003200,
   "reportedTurnOverQuantity": 6000000,
   "turnOverValue": 2451310000000,
   "reportedTurnOverValue":
2450000000000,
   "averagePrice": 409375,
   "totalNumberOfTrades": 19
}
```


Trade Statistics – Trade report only (no price)


```
{
   "msgType": "I",
   "nanos": 111729974,
   "orderBookId": 65537,
   "openPrice": -2147483648,
   "highPrice": -2147483648,
   "lowPrice": -2147483648,
   "lastPrice": -2147483648,
   "lastAuctionPrice": -2147483648,
   "turnOverQuantity": 1000000,
   "reportedTurnOverQuantity": 1000000,
   "turnOverValue": 410000000000,
   "reportedTurnOverValue":
410000000000,
   "averagePrice": -2147483648,
   "totalNumberOfTrades": 1
}
```

TFEX Trade Statistics - Turnover value not available	Trade Statistics is reset

{
"msgType": "I",
"nanos": 348248857,
"orderBookId": 413273,
"openPrice": 6000000,
"highPrice": 6010000,
"lowPrice": 6000000,
"lastPrice": 6010000,
"lastAuctionPrice": -2147483648,
"turnOverQuantity": 141,
"reportedTurnOverQuantity": 100,
"turnOverValue": 0,
"reportedTurnOverValue": 0,
"averagePrice": 6000975,
"totalNumberOfTrades": 11
}
	{
	"msgType": "I",
	"nanos": 584043385,
	"orderBookId": 413273,
	"openPrice": -2147483648,
	"highPrice": -2147483648,
	"lowPrice": -2147483648,
	"lastPrice": -2147483648,
	"lastAuctionPrice": -2147483648,
	"turnOverQuantity": 0,
	"reportedTurnOverQuantity": 0,
	"turnOverValue": 0,
	"reportedTurnOverValue": 0,
	"averagePrice": -2147483648,
	"totalNumberOfTrades": 0
	}

7.6.5 iNAV Message (f)

The INAV Message is used to publish iNAV (Indicative NAV) Prices. iNAV is available only for ETF (Exchanged Traded Fund).

Name	Offset	Length	Value	Notes
Msg Type	0	1	"f"	f – iNAV Message.
Nanos	1	4	Nmeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
INAV	9	4	Price	Indicative NAV
Change	13	4	Price	iNAV change from previous day
Percentage Change	17	4	Price	Percentage of iNAV change from previous day (2 decimal digits)
Timestamp	21	8	Timestamp	Time of INAV data

7.6.6 Index Price Message (J)

The Index Price Message is used to publish Index Prices for index instruments.

Normally SET provides Index at 15-second intervals. Index message begins to transmit after market opens. If clients would like to get closing index of previous day before market opens, clients can get it from closing price (priceType=11) in Reference Price message.

There are some special cases of index transmission as follows:

- For Index without constituents such as MINE index, there is no index messages sent out during the day. However, after the market closes, the most recent index value will be transmitted.
- There may be Index message sent out (with the most recent index value) during Pre-open period if all constituents of the index are halted/suspended at instrument level.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"J"	J – Index Price Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
Value	9	8	Price	Index last value of current day.
High Value	17	8	Price	Index high value of current day
Low Value	25	8	Price	Index low value of current day
Open value	33	8	Price	Index open value of current day
Traded Volume	41	8	Numeric	Index traded volume of current day
Traded Value	49	8	Price	Index traded value of current day
Change	57	8	Price	Index change compared to previous close
Change Percent	65	4	Price	Index change percent compared to previous close (2 decimal digits).
Previous Close	69	8	Price	Index close value of previous day
Close	77	8	Price	Index Close value of current day
Timestamp	85	8	Timestamp	The time the Index value

7.6.7 Market Statistic Message (g)

The Market Statistics Message is used to publish Market Statistics. Normally SET provides Market Statistics at 15-second intervals.

- For TFEX Night markets, Market Statistics of previous day will be carried to next trading day. This means market statistics on day T is calculated from trading data of Night session on date T-1 and Day session on day T. When the clients connect to the system on day T, you will receive carried market statistics of Night session of day T-1.
- Resetting Market statistics flow (same as instrument trade statistics)
  - Market statistics of TFEX Night market will be reset to zero after entering INTERMISSION3_D session (before Pre-Night session).
  - Market statistics of Equity market and TFEX Day market will be reset as zero after entering RESET_STAT session.

Name	Offset	Length	Value	Notes	Equity	TFEX
Msg Type	0	1	"g"	g – Market Statistics Message.	✓	✓
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.	✓	✓
Market Statistics ID	5	12	Alpha	Unique ID for market statistics record e.g. SET, MAI, TXI, TXS and etc. The possible values of Market Statistics ID might not the same as those of Market ID or Market Segment ID because the system has capability to calculate market statistics across multiple markets.	✓	✓
Currency	17	3	Alpha	Trading currency e.g. THB, USD and etc. For TFEX markets, this field can be blank in case that there are multiple currencies in the same Market Statistics ID.	✓	✓
Market Statistics Time	20	8	Time-stamp	Time of Market statistics	✓	✓
Total Trades	28	4	Numeric	Total number of trades across the corresponding market segment and currency	✓	✓
Total Quantity	32	8	Numeric	Total quantity traded across the corresponding market segment and currency	✓	✓
Total Value	40	8	Numeric	Price * quantity for all trades in the corresponding market segment and currency ⓘ Value has no decimal place	✓	N/A
Up Quantity	48	8	Numeric	Total volume of securities traded that current price change is positive, compared to previous closing price	✓	N/A
Down Quantity	56	8	Numeric	Total volume of securities traded that current price change is negative, compared to previous closing price	✓	N/A

Name	Offset	Length	Value	Notes	Equity	TFEX
No Change Volume	64	8	Numeric	Total volume of securities traded that current price change is zero, compared to previous closing price	✓	N/A
Up Shares	72	4	Numeric	Number of securities that have a trade and current price change is positive compared to previous closing price	✓	N/A
Down Shares	76	4	Numeric	Number of securities that have a trade and current price change is negative compared to previous closing price	✓	N/A
No Change Shares	80	4	Numeric	Number of securities that have a trade and current price change is zero compared to previous closing price	✓	N/A

7.6.8 Reference Price Message (Q)

The Reference Price Message is used to publish Reference Prices per Price Types.

The clients will receive latest values of reference prices when connecting in the morning. Ever last price (price type 6) is sent out only in start of day; while the other price types are sent out both of start of day and after market is closed.

The dissemination behaviors of each price type are as follows:

- **Price type=6**: **Ever last price** is sent out only in start of day and not sent out after market is closed even though the instruments have trades.

  Updated Timestamp field of Ever last price is the actual trading date/time based on the calendar date. For example, the date/time of ever last prices of the instruments which are traded in the night session, are the previous calendar date/time.

  In case that the Equity instrument has split par action, the client will receive adjusted ever last price after the corporate action process. Timestamp field is also updated.

- **Price type=11**: **Closing price** is sent out after market closed to update latest prices. If the instruments have no trade, closing price is carried from previous closing price. Updated Timestamp field is updated to current trading day.

  In case that the Equity instrument has split par action, the client will receive adjusted closing price after the corporate action process.

- **Price type=5**: is used for different purposes as follows:
  - **Settlement price**: is sent out for TFEX series after market closed for updated settlement prices.
  - **Reference price**: In case of new TFEX series which is created from corporation action, their reference prices will be carried from the old series. The reference price is used to calculate

last trade price change of TFEX series.

- **IPO price**: is sent for Equity instrument for Initial Public Offering price. If new listed stock is from merger of companies, IPO price will be not sent out.

Moreover Ever last price, Closing price and IPO prices can be reset to -2 147 483 648 (min integer) for some cases e.g. when stock which is suspended for a long period is back to trade.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"Q"	Q – Price Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
Price Type	9	1	Numeric	Type of price published. Valid values: 5 = Settlement Price/Reference price for TFEX (only outright series) or IPO price for Equity 6 = Ever Last Price, excluding trades from the current Trading day 11= Closing Price
Price	10	4	Price	
Updated Timestamp	14	8	Timestamp	Timestamp when the Price was updated

ⓘ If Price is set to decimal -2 147 483 648 (min integer value), this means that no price is available.

7.6.9 Open Interest Message (h)

The Open Interest Message is used to publish Open Interest of Derivatives market series. The message is published 2 times a day.

- In the morning when the clients connect to the server, the client will receive updated Open interest which includes value from night session transactions of previous day.
- After market is closed (around 18.00 p.m. BKK local time), the clients will receive updated Open interest.

For new instruments created when the instrument has corporate action, the client will get updated Open interest on next trading day after login in the morning.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"h"	h– Open Interest Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	
Open Interest	9	8	Numeric	Total number of outstanding contracts
Timestamp	17	8	Timestamp	

7.7 Market Announcements

7.7.1 Market Announcement Message (N)

The Market Announcement Message is used to publish Exchange's announcements for Instruments, Markets or Exchanges.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"N"	N – Market Announcement Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	If the value is 0, the message is not applied to any specific Instrument.
Market Code	9	1	Numeric	If the value is 0, the message is not applied to any specific market.
Exchange Code	10	1	Numeric	If this field contains value, the message is applied to all products under SET and TFEX. If not, the value is 0.
Message Information Type	11	1	Numeric	Kind of message sent in announcement. 1 = CompanyAnnouncement 2 = SensitiveCompanyAnnouncement 3 = Market Message 4 = ImportantMarketMessage 5 = Static Line 6 = CriticalStaticLine 7 = Notice Received 8 = Advertisement
Message Source	12	80	Alpha	A description of the source of the message, e.g. the market control.

Name	Offset	Length	Value	Notes
Message Priority	92	1	Numeric	1 = Low Priority 2 = Medium Priority 3 = High Priority 4 = Critical Priority
Message Header	93	80	Alpha	Header of the message. Used to specify a short description of a message.
Items URL	173	1	Numeric	Characters in Document URL link.
Document URL	174	255	Alpha	The URL link to somewhere else on the Web where more information may be found, typically a document.
Number of Message Items	429	1	Numeric	Number of market message records. The value is same as Number of Message Items in Array field.
Number of Message Items in Array	430	1	Numeric	Number of message items in the following array. The maximum number is 10. Each message item is a row in the message.
Array of Message items	431 + n*80	80* Number of Message Items	Alpha	Array with rows in the message. Each row has a maximum of 80 characters. The message might contain new line character. The item counter n for the Offset is counted from 0 up to Number of Message Items -1 when Number of Message Items > 0.

Advertisement

When the Market Announcement message is used as advertisement events (*Message Information Type = Advertisement*), it has a specific defined format overlaying on the Message item as shown in the table below.

Name	Offset	Length	Value	Notes
Msg Type	0	1	"N"	N – Market Announcement Message.
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.
Orderbook ID	5	4	Numeric	

Name	Offset	Length	Value	Notes
Market Code	9	1	Numeric	0
Exchange Code	10	1	Numeric	0
Message Information Type	11	1	Numeric	8 = Advertisement
Message Source	12	80	Alpha	Participant ID who inserts or cancels this advertisement.
Message Priority	92	1	Numeric	2 = Medium Priority
Message Header	93	80	Alpha	Header of the message. Used to specify a short description of a message.
Items URL	173	1	Numeric	0
Document URL	174	255	Alpha	
Number of Message Items	429	1	Numeric	1
Number of Message Items in Array	430	1	Numeric	1
Array of Message items	431	80	Alpha	The message content has the format defined below.
Advertisement ID	*431*	*10*	*Alpha*	*Unique ID of advertisement (Left justified with blank space padding)*
Action	*441*	*1*	*Alpha*	*I = Insert (new advertisement)* *C = Cancel (the specified advertisement ID)*
Side	*442*	*1*	*Alpha*	*B = Buy* *S = Sell*
Quantity	*443*	*20*	*Alpha*	*Left justified with blank space padding*
Price	*463*	*10*	*Alpha*	*Left justified with blank space padding*
Contact Name	*473*	*10*	*Alpha*	*Left justified with blank space padding*
Contact Info	*483*	*28*	*Alpha*	*Left justified with blank space padding*

Example message (converted to json format)

Announcement for all market segments

```
{
   "msgType": "N",
   "nanos": 619635086,
   "orderBookId": 0,
   "exchangeId": 35,
   "marketId": 0,
   "messageInformationType": 4,
   "messageSource": "Market Control",
   "messagePriority": 3,
   "header": "Test Important Market
Control Message",
   "itemsUrl": 0,
   "documentUrl": "",
   "numberOfMessageItems": 1,
   "messageItems": [
      {
         "line": "Test Important Market
Control Message"
      }
   ]
}
```

Announcement for specific market segment

```
{
   "msgType": "N",
   "nanos": 42751161,
   "orderBookId": 0,
   "exchangeId": 35,
   "marketId": 104,
   "messageInformationType": 6,
   "messageSource": "Market Control",
   "messagePriority": 4,
   "header": "Critical Static Line
Message",
   "itemsUrl": 0,
   "documentUrl": "",
   "numberOfMessageItems": 2,
   "messageItems": [
      {
         "line": "Critical Static Line
Message#1
"
      },
      {
         "line": "Critical Static Line
Message#2"
      }
   ]
}
```

New advertisement

```
{
   "msgType": "N",
   "nanos": 355916737,
   "orderBookId": 65790,
   "exchangeId": 0,
   "marketId": 0,
   "messageInformationType": 8,
   "messageSource": "0001",
   "messagePriority": 2,
   "header": "Advertisement Member=0163
ID=457 (New)",
   "itemsUrl": 0,
   "documentUrl": "",
   "numberOfMessageItems": 1,
   "messageItems": [
      {
         "line": "457       IB1000000
65.123456 Mr. AAA   0891234567"
      }
   ]
}
```

Cancel advertisement

```
{
   "msgType": "N",
   "nanos": 996075847,
   "orderBookId": 65790,
   "exchangeId": 0,
   "marketId": 0,
   "messageInformationType": 8,
   "messageSource": "0001",
   "messagePriority": 2,
   "header": "Advertisement
Member=0163 ID=457 (Cancelled)",
   "itemsUrl": 0,
   "documentUrl": "",
   "numberOfMessageItems": 1,
   "messageItems": [
      {
         "line": "457       CB1000000
65.123456 Mr. AAA   0891234567"
      }
   ]
}
```

7.7.2 News Message (j)

The News Message is used to publish news announcement from companies, issuers or regulators. Clients will receive news according to subscription rights by ReportType, MarketId and Language.

Currently SET sends news and reports from each markets at different time:

- SET and mai: 6:30 – 22:40
- TFEX: 6:30 – 3:05 (next day)

Since the system does not allow members to query news from the previous day, SET recommends that clients stay connected to the system at least until 22:40 to receive news. SET will resend TFEX news after 22:40 in next trading day. This means that all clients who given permission to receive TFEX news will receive it again in the following morning. Client can discard duplicated news by checking NewsID field.

Name	Offset	Length	Value	Notes	NWS	FIN	ANN /F56
Msg Type	0	1	"j"	j – News Message.	✓	✓	✓
Nanos	1	4	Numeric	Nanoseconds portion of the timestamp.	✓	✓	✓
Report Type	5	10	Alpha	Valid values: NWS = News FIN = Financial Status ANN = Annual Report F56 = Form56-1 ⓘ The valid values are available according to member permission	NWS	FIN	ANN /F56
Market ID	15	1	Alpha	The market the news applies to Valid values: A = SET S = MAI D = TFEX ⓘ The valid values are available according to member permission ⓘ Market=D is available only for NWS report type	✓	✓	✓
Language	16	1	Alpha	The language the news is presented in Valid values: T = Thai E = English B = Both	✓	✓	✓
Sender Name	17	20	Alpha	The sender symbol including regulator, issuer and listed company	✓	✓	✓
Publish Date	37	8	Time-stamp	News publish date and time	✓	✓	✓

Name	Offset	Length	Value	Notes	NWS	FIN	ANN /F56
News Type	45	2	Alpha	The type of news Valid values for NWS: ‘ ‘ : Normal 41 : SET News Release 45 : TFEX News Release 51 : Form 45-1 / Form 45 52 : Form 45-2 53 : Form 45-3 59 : NAV 61 : Investor Alert News 62 : Reprimand News 71 : Contract Adjustments Valid values for F56: ‘ ‘ : Form 56-1 56, 58 : Form 56-1 One Report 57, 59 : Form 56-1 One Report (Change Business Type)	✓	55	✓
Head Line	47	250	Alpha	The headline for the news ⓘ Encoding is ISO8859-11 to support Thai language	✓	✓	✓
News File Name	297	50	Alpha	The News or report file name. It can be plain text or zip file	✓	✓	✓
PDF File Name	347	50	Alpha	The PDF file name	✓		
XML File Name	397	50	Alpha	The XML file name			
Period	447	4	Alpha	Report period (year)		✓	✓
Date as of	451	4	Date	Report as of date		✓	✓
Quarter	455	1	Numeric	The quarter the financial statements represent Valid values: 1=Q1 2=Q2 3=Q3 4=Q4 6=Half Year 9=Annual		✓	
Fin Stmt Type	456	1	Alpha	The financial account type Valid values: U = Separate C = Consolidated and Separate B = Equity method and Separate		✓	

Name	Offset	Length	Value	Notes	NWS	FIN	ANN /F56
Fin Stmt Status	457	1	Alpha	The financial statements status Valid values: U = Unreviewed R = Reviewed A = Audited		✓	
Status	458	1	Alpha	The status of News and reports Valid values: N=New D=Delete	✓	✓	✓
News ID	459	20	Alpha	Unique News ID, used for checking duplicated and deleted News	✓	✓	✓
News Group	479	20	Alpha	News announcement group	✓	✓	✓
Number Of Template Codes	499	1	Numeric	Number of News template codes	✓	✓	
Template Codes	500	2 * Number of Template Codes	Alpha	Array of News template codes	✓	✓	
Number of Related Symbols	500 + (2 * Number of Template Codes)	1	Numeric	Number of Instrument symbols which this news message applies	✓	✓	✓
Related Symbols	501 + (2 * Number of Template Codes)	20 * Number of Related Symbols	Alpha	Array of Instrument symbols which this news message applies ⓘ Symbols may not be available in OrderBookDirectory message e.g. News of new listed stocks	✓	✓	✓

Example message (converted to json format)

News (NWS)


```
{
   "msgType": "j",
   "status": "N",
   "nanos": 148245017,
   "reportType": "NWS",
   "marketId": "A",
   "language": "E",
   "senderName": "AAA",
   "publishDate": "16311654760000000000",
   "newsType": "",
   "headline": "Notification of Book
Closed Date and Trading Suspension of 3
DWs issued by AAA",
   "newsFileName":
"nws/en/txt/0171NWS090920211231170629E.tx
t",
   "pdfFileName":
"nws/en/pdf/0171NWS090920211231170629E.pd
f",
   "xmlFileName": "",
   "period": "",
   "dateAsOf": 0,
   "quarter": 0,
   "finStmtType": "",
   "finStmtStatus": "",
   "newsId": "6840940",
   "newsGroup": "684094",
   "templateCodes": [
      {
         "data": "90"
      }
   ],
   "relatedSymbols": [
      {
         "data": "ACE13C2109A"
      },
      {
         "data": "BANP13C2109A"
      },
      {
         "data": "BCPG13C2109A"
      }
   ]
}
```


Deleted News (NWS)


```
{
   "msgType": "j",
   "status": "D",
   "nanos": 525626507,
   "reportType": "NWS",
   "marketId": "S",
   "language": "E",
   "senderName": "MAI",
   "publishDate":
"16311654760000000000",
   "newsType": "",
   "headline": "SET adds new listed
securities : BC-W1 to be traded on
September 8, 2021",
   "newsFileName": "",
   "pdfFileName": "",
   "xmlFileName": "",
   "period": "",
   "dateAsOf": 0,
   "quarter": 0,
   "finStmtType": "",
   "finStmtStatus": "",
   "newsId": "683642E",
   "newsGroup": "683642",
   "templateCodes": [
      {
         "data": "18"
      }
   ],
   "relatedSymbols": [
      {
         "data": "BC"
      },
      {
         "data": "MAI"
      }
   ]
}
```

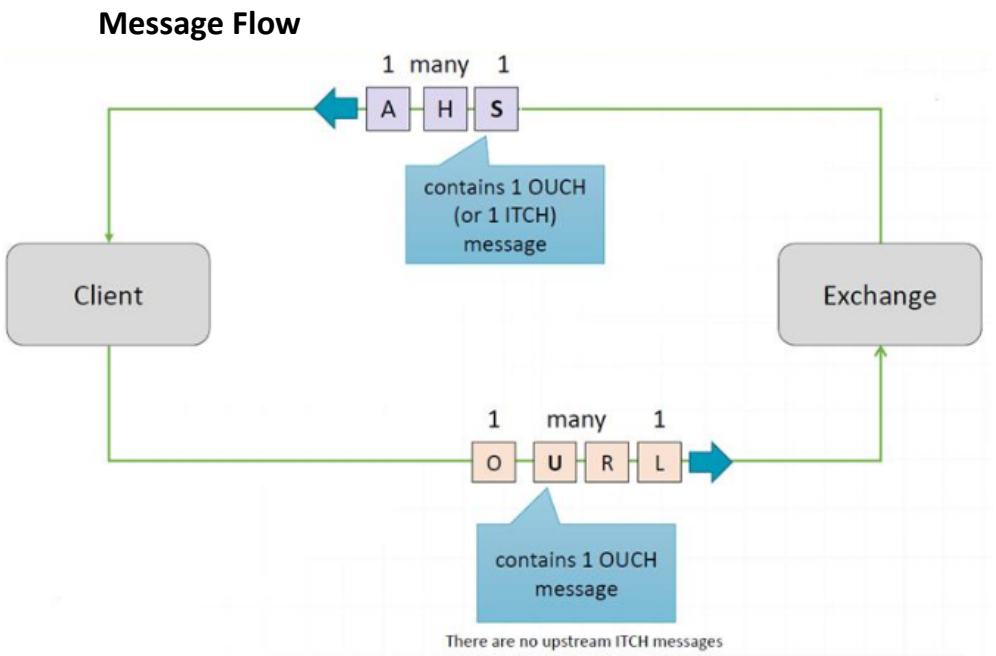
Financial Statements (FIN)	Annual Report (ANN)


```
{
   "msgType": "j",
   "status": "N",
   "nanos": 960837195,
   "reportType": "FIN",
   "marketId": "A",
   "language": "E",
   "senderName": "ABC",
   "publishDate":
"1631337061000000000000",
   "newsType": "55",
   "headline": "Financial Statement
Yearly 2020 (Audited)",
   "newsFileName":
"fin/en/1568FIN160720211619270712E.zip",
   "pdfFileName": "",
   "xmlFileName": "",
   "period": "2020",
   "dateAsOf": 20201231,
   "quarter": 9,
   "finStmtType": "C",
   "finStmtStatus": "A",
   "newsId": "672356E",
   "newsGroup": "672356",
   "templateCodes": [],
   "relatedSymbols": [
      {
         "data": "ABC"
      }
   ]
}
```

```
{
   "msgType": "j",
   "status": "N",
   "nanos": 151584483,
   "reportType": "ANN",
   "marketId": "A",
   "language": "E",
   "senderName": "ABC",
   "publishDate":
"16311654760000000000",
   "newsType": "",
   "headline": "Annual Report/ 2021",
   "newsFileName":
"ann/1568ANN260820211212580783E.zip",
   "pdfFileName": "",
   "xmlFileName": "",
   "period": "2021",
   "dateAsOf": 0,
   "quarter": 0,
   "finStmtType": "",
   "finStmtStatus": "",
   "newsId": "682089E",
   "newsGroup": "682089",
   "templateCodes": [],
   "relatedSymbols": [
      {
         "data": "ABC"
      }
   ]
}
```

8. The Example Message Flow

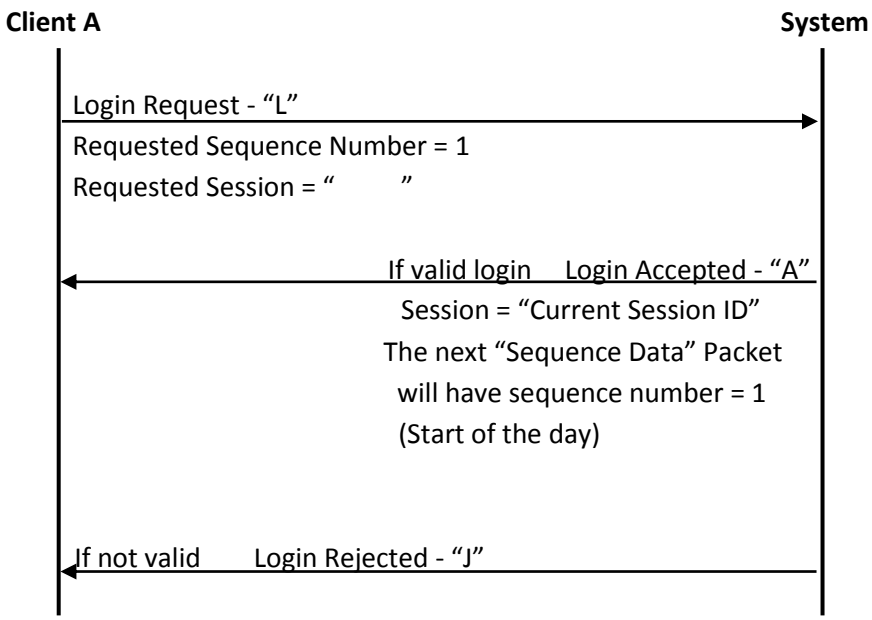
8.1 SoupBinTCP example message flow



8.1.1 Login

Client opens a TCP/IP socket and sends a Login Request Packet to the server (ITCH Gateway or Glimpse)

Scenario 1: Start of the day



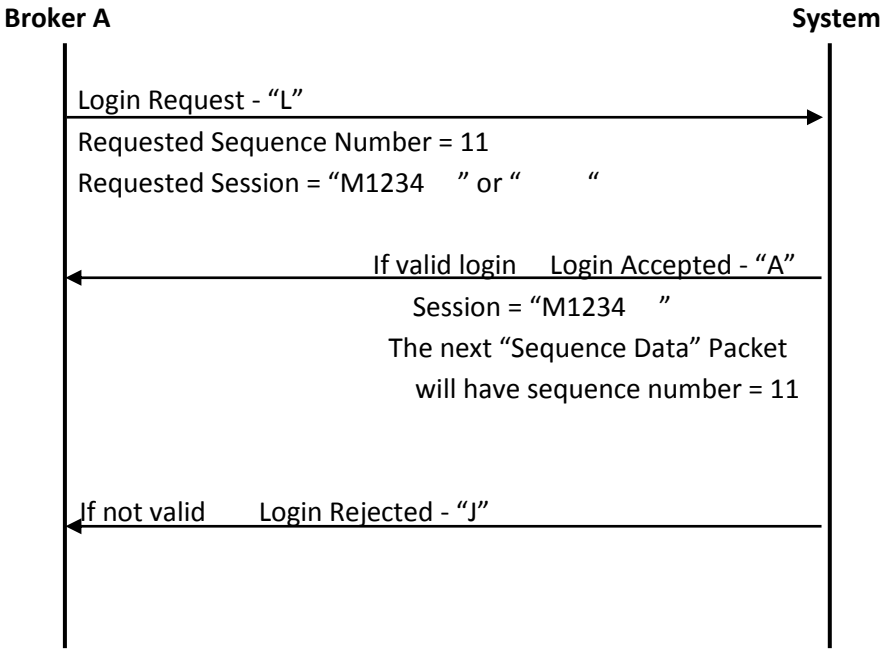
Each time a sequenced data packet arrives increment the counter by 1.

Login Request Example message

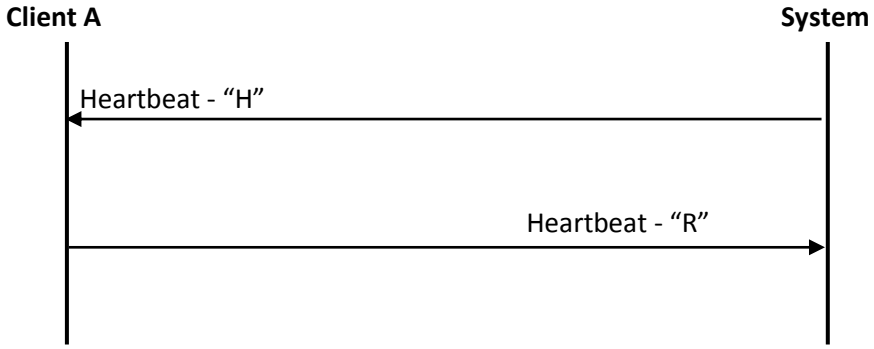
Name	Length	Value	Example Value
Packet Length	2	Integer	47
Packet Type	1	"L"	“L”
Username	6	Alphanumeric	“99F21”
Password	10	Alphanumeric	“ffffff!2 ”
Requested Session	10	Alphanumeric	“ ”
Requested Sequence Number	20	Numeric	0

Scenario 2: Disconnect when sequence number = 10 (Client login with valid sequence)

Before disconnect client connect with Session ID = “M1234 ” and Sequence number = 10.



8.1.2 Heartbeat



8.2 ITCH example message flow

8.2.1 Reference Data and Corporate Action

The corporate action process, e.g. new list, delist, change symbol, split par, XD, XR and etc. is run after market closed and the related order books information will be updated (as below tables). Please note that Equity market and Derivatives market might have some different behaviors.

Equity Market

Corporate Action	Updated Data	After market closed	On next day morning
1. New list	New order book is created and Other sign is set as "N"	✓	
2. Delist	Other sign is set as "X"	✓	
3. Change symbol	Status of old order book is Issued	✓	
	Old order book will be deleted		✓
	New order book ID of new symbol is created with Active status Originated from field refers to old symbol	✓	
	Ever last, Closing price and IPO price (if any) are carried to new order book ID	✓	
4. Split par	Other sign is set as "S"	✓	
	Ever last and Closing price are adjusted	✓	
5. Beneficial CA e.g. XD, XR	Corporate Action Code is set	✓	
	Other sign is set as "L" and related long orders are cancelled		✓

Derivatives Market

Corporate Action	Updated Data	After market closed	On next day morning
1. Generate new series of market without night session	New outrights and combination series (if any) are created		✓
2. Generate new series of market with night session	New outrights and combination series (if any) are created	✓	
3. Change symbol and Beneficial CA e.g. XD, XR	Status of old series is updated - Outright series status=Issued - Combination series status=Delist	✓	
	Old series will be deleted		✓
	New order book ID of new outright series symbol is created with Active status	✓	
	Information of the new underlying order book ID		✓
	New order book ID of new combination series is created with Active status		

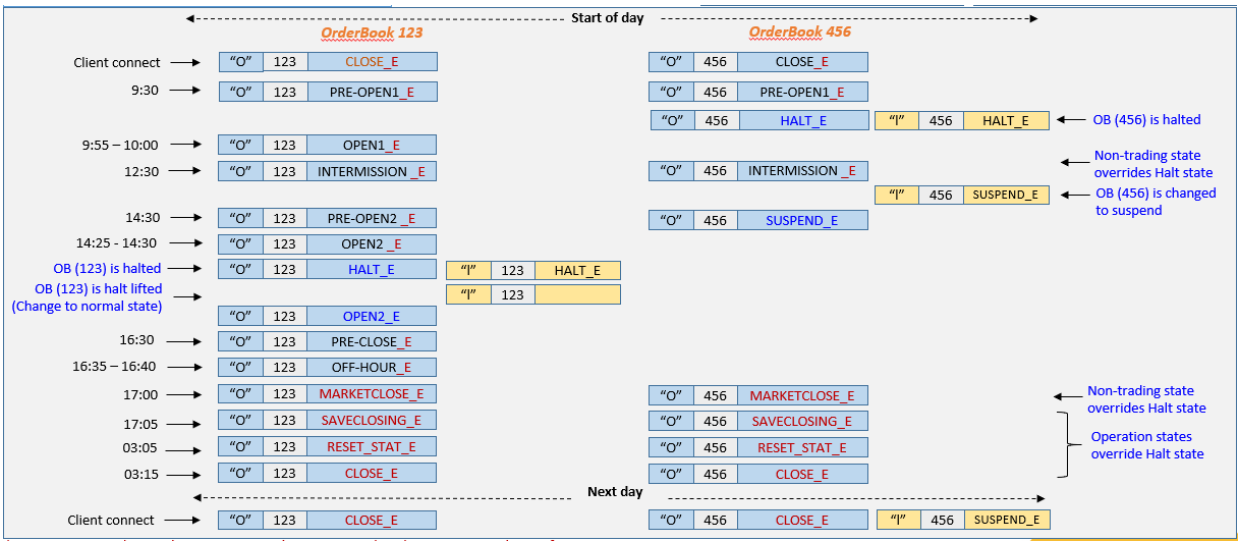
Corporate Action	Updated Data	After market closed	On next day morning
	- For markets with night session (if any) - For markets without night session	✓	✓
	Reference prices - Reference price is updated through priceType=5 - In case of changing symbol, Ever last and Closing price are carried to new order book ID	✓	
	Open interest is updated		✓

8.2.2 Trading session and instrument state example flow

The system will send trading session and halt status via Order Book State ("O") and Halt Information ("I") messages. Order Book State ("O") shows active state of the instruments. Halt Information ("I") shows specific instrument state e.g. halt status or status after halt is lifted which depending on business flow. Please see related cases of the messages as below.

- **In case of normal trading session and instruments are halted or suspended**

  Scenario:
  - *OrderBookID 123 is halted and lifted*
  - *OrderBookID 456 is halted and SP*



- **In case of SET Circuit Breaker**

  Scenario:
  - *OrderBookID 456 has been suspended since previous day*
  - *OrderBookID 789 is series in TXI or TXS*
  - *In Open1 session, SET index reaches percent threshold of CircuitBreaker. SET market and related TFEX markets (TXI and TXS) are halted with CircuitBreaker reason.*

	SET OrderBook 123			SET OrderBook 456						TFEX (TXI or TXS) OrderBook 789		
Client connect →	"O"	123	CLOSE_E	"O"	456	CLOSE_E	"I"	456	SUSPEND_E	"O"	789	CLOSE_D
9:30 →	"O"	123	PRE-OPEN1_E	"O"	456	SUSPEND_E				"O"	789	PRE-MORNING_D
9:55 – 10:00 →	"O"	123	OPEN1_E							"O"	789	MORNING_D
SET Circuit Breaker →	"O"	123	CIRCUIT_BREAKER_E	"O"	456	CIRCUIT_BREAKER_E				"O"	789	CIRCUIT_BREAKER_D
Lift CB, enter to Pre-Open state →	"O"	123	PRE-OPEN_E	"O"	456	SUSPEND_E				"O"	789	PRE-OPEN_D
Enter to normal state →	"O"	123	OPEN1_E							"O"	789	MORNING_D
12:30 →	"O"	123	INTERMISSION_E	"O"	456	INTERMISSION_E				"O"	789	INTERMISSION_D

- **In case of TFEX Price Break**

  Scenario:

  - *In Morning session, TFEX commodities markets e.g. TXD, TXM reach percent threshold of PriceBreak.*
  - *The trading session of series under related markets e.g. OrderBookID 321, will be changed to Pre-Open-CB immediately after PriceBreak is triggered without halt period and will enter normal session afterward.*
  - *When entering to normal state, state in Halt Information message will be cleared.*

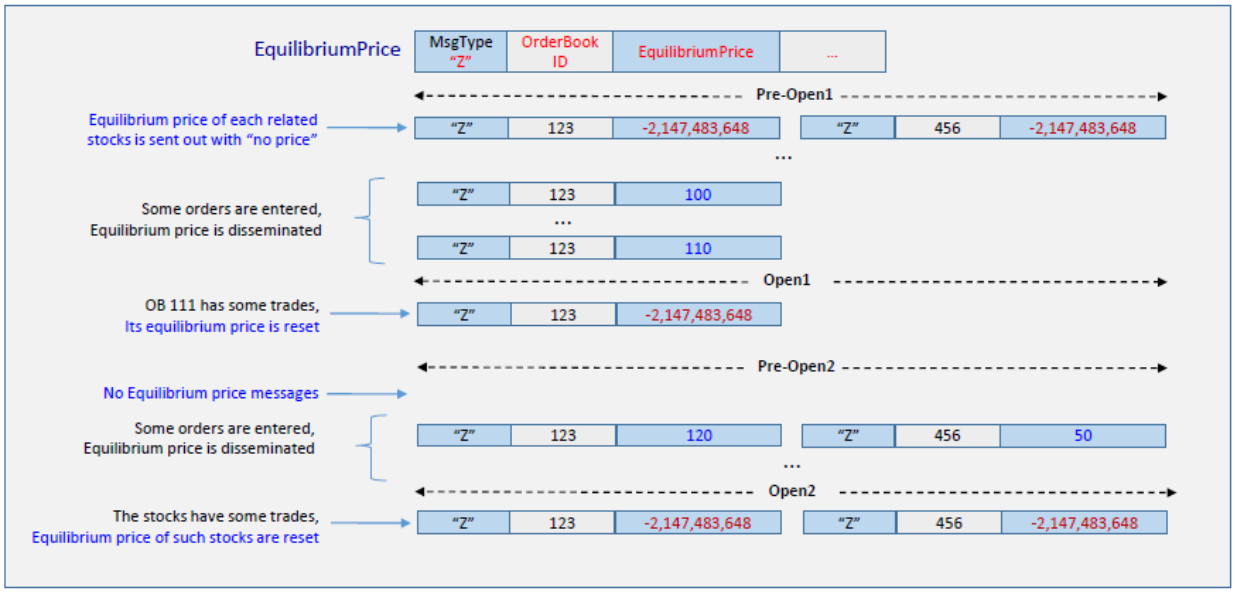
	OrderBook 321					
Client connect →	"O"	321	CLOSE_D			
9:15 →	"O"	321	PRE-MORNING_D			
9:45 →	"O"	321	MORNING_D			
TFEX Price Break and enter to Pre-Open state immediately →	"O"	321	PRE-OPEN_CB_D	"I"	321	PRE-OPEN_CB_D
Enter to normal state after Pre-open →	"O"	321	MORNING_D	"I"	321	
12:30 →	"O"	321	INTERMISSION_D			

- **In case of the instruments is on last trading date/time**

  Scenario: *At last trading time on last trading date of the series, the session status will be set to Expired.*

	OrderBook 321					
Client connect →	"O"	321	CLOSE_D			
9:15 →	"O"	321	PRE-MORNING_D			
9:45 →	"O"	321	MORNING_D			
Last trade time 11:00 →	"O"	321	EXPIRED_D	"I"	321	EXPIRED_D
12:30 →	"O"	321	INTERMISSION_D			
Pre-Afternoon 13:45 →	"O"	321	EXPIRED_D			
16:55 →	"O"	321	SETTLEMENT_D			
17:05 →	"O"	321	MARKETCLOSE_D			
3:05 →	"O"	321	RESET_STAT_D			
3:15 →	"O"	321	CLOSE_D			

8.2.3 Equilibrium price example flow



8.2.4 Market by Price (MBP) example flow

Scenario: *Receive 2 levels of bid/offer*

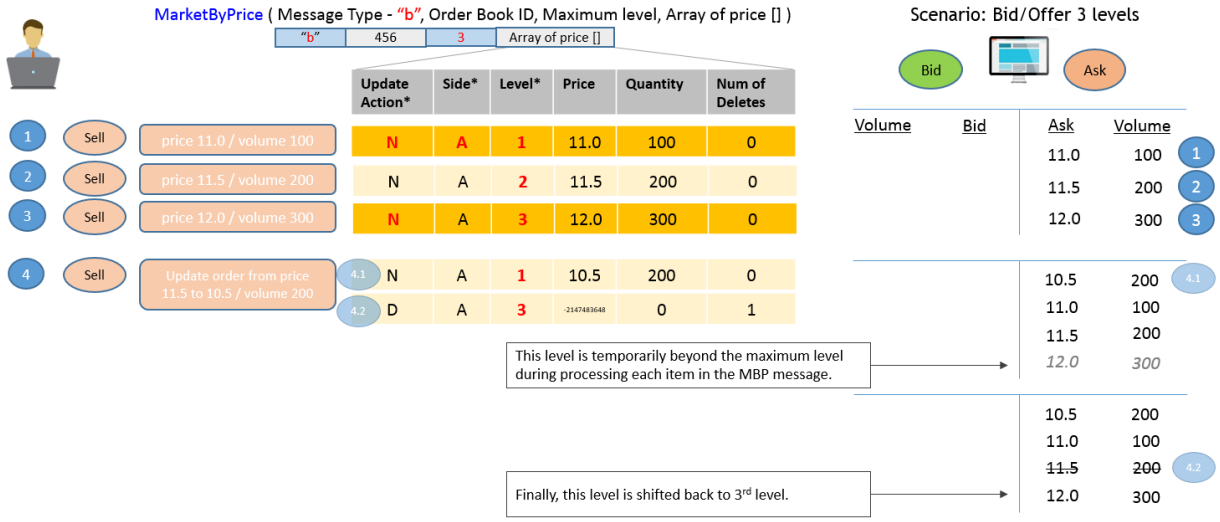
MarketByPrice (Message Type - "b", Order Book ID, Maximum level, Array of price [])

"b"	123	2	Array of price []

		Update Action*	Side *	Level *	Price	Quantity	Num of Deletes
Sell	price 110 / volume 100	N	A	1	110	100	0
Sell	price 115 / volume 200	N	A	2	115	200	0
Sell	price 110 / volume 200	C	A	1	110	300	0
Sell	price 120 / volume 200	Price Level 3 -> No MBP sent out					
Buy	price 95 / volume 200	N	B	1	95	200	0
Buy	price 100 / volume 100	N	B	1	100	100	0
Buy	price 105 / volume 200	N	B	1	105	200	0
Buy ✗	price 105 / volume 200	D	B	1	0	200	1
		N	B	2	95	200	0
Buy	price 115 / volume 600	N	B	1	115	100	0
		D	A	1	0	300	2
		N	A	1	120	200	0

(Matched 500 to Ask price 110 and 115)

Scenario: *A level can temporarily be beyond the maximum level during processing multiple updates in one MBP message*



8.3 How to maintain the local order book view

This section describes how to handle Market by Price (MBP) flow. The model for market data is to optimize bandwidth consumption by only sending delta information when a change occurs. It requires the receivers to keep a real-time database of the whole orderbook and apply the deltas in order.

Below are a couple of examples to illustrate how it works. The examples are taken from the Price Depth examples section in [2], to simplify comparison to best practices. In the message examples some fields have been removed from messages to reduce the clutter when illustrating the examples.

8.3.1 Example 1 – Initial Market

This example shows the initial market (with 2 levels bid/ask) for the following examples.

BID	ASK
2 @9740	5 @9760
5 @9730	3@9770

8.3.1.1 An order is added to the book

An order is added creating a new best price level: Buy 3 @ 9750

Field name	Value	Description
Order book ID	111	
Number of Level items	1	
Level Update Action	N	New

Field name	Value	Description
Side	B	Bid
Level	1	Price Level
Quantity	3	
Price	9750	

Note: The receiver is responsible for shifting the levels below the newly inserted one down. And price level, Bid **5 @9730,** which is out of range has to be removed from client bid queue.

Result:

BID	ASK
3@9750	5@9760
2 @9740	3@9770

8.3.1.2 Change an order

An order is changed to increase its size: **Bid order @9750 is increased by 2**.

Field name	Value	Description
Order book ID	111	
Number of Level items	1	
Level Update Action	C	Change
Side	B	Bid
Level	1	Price Level
Quantity	5	
Price	9750	

Result:

BID	ASK
5@9750	5@9760
2@9740	3@9770

8.3.1.3 Delete an order which deletes the price level

An order is deleted at a lower price level and it is the only order at that price level: **Delete Bid @9740**. And **Bid 5@9730** will be added instead.

Field name	Value	Description
Order book ID	111	
Number of Level items	2	
Level Update Action	D	Delete
Side	B	Bid
Level	2	Price Level
Number Of Deletes	1	Number of levels to delete
Level Update Action	N	New
Side	B	Bid
Level	2	Price Level
Quantity	5	
Price	9730	

Result:

BID	ASK
5@9750	5 @9760
5@9730	3@9770

8.3.2 Example 2 – A level can temporarily be beyond the maximum level during processing multiple updates in one MBP message

Initial market (with 3 levels bid/ask).

BID	ASK
	100 @11.0
	200 @11.5
	300 @12.0

The price of the order 200 @11.5 is change to 200 @10.5. This results in adding Ask 200 @10.5 and deleting 200 @11.5. However, during processing these items of update, the 3rd price level can temporarily be beyond the maximum level.

Field name	Value	Description
Order book ID	222	
Number of Level items	2	

Field name	Value	Description
Level Update Action	N	New
Side	A	Ask
Level	1	Price Level
Quantity	200	
Price	10.5	
Level Update Action	D	Delete
Side	Ask	Ask
Level	3	Price Level
Number Of Deletes	1	Number of levels to delete

During processing the item #1

BID	ASK
	200 @10.5
	100 @11.0
	200 @11.5
	300 @12.0

Finally after processing all items.

BID	ASK
	200 @10.5
	100 @11.0
	300 @12.0

Revision History

Release Date	Effective Date	Version	Change Summary
8-Oct-2020	TBD	0.1	Initial document for ITCH protocol
30-Oct-2020	TBD	0.1.1	• Order Book Directory message: add modifier field • Update some field names to be same as Java library class: Msg type, Nanos, Exchange Code, Market Code
30-Nov-2020	TBD	0.2	• Add section 7.2 – ITCH example message flow • Add section 7.3 – How to maintain the local order book view • Update some field names and more description
22-Jan-2021	TBD	0.3	• Update Architecture picture: Add ITCH-News server • Add new messages: Price Limit (k), Open Interest (h) and News (j) • Change DateTime data type to be Timestamp (nanoseconds since epoch): Reference Price, Trade Ticker, iNAV, Index value, Market Statistics and Market Announcement • Market Statistics: swap fields • Order Book Directory: - Add underlying name field - Change underlying order book ID field to be numeric - Change field length of Exchange Code and Market Code to 1 byte
5-Mar-2021	TBD	0.4	• Add Halt Information message and remove Halt State field from Order Book State message • Reference Price: remove PriceType=1 (Price limit reference price) • Market Statistics: update field length • Trade Statistics: add more description • Market Announcement: - Add new possible values of Message Information Type and to support Advertisement - Remove some unused fields - Add the format of Advertisement • News: Update possible value of report type, NWS
2-Apr-2021	TBD	0.5	• Data Type: Update Price data type able to be 8 bytes length • Order Book Directory:

Release Date	Effective Date	Version	Change Summary
			- Add NotationDate and DecimalsInContractSizePQF fields - Add possible value of Status field - Update some field names and add description • Index Price and News: update field length • Market Announcement: update field format • Add description for Reference Price and Open Interest message • Add more example flows in section 7.2.1 - Trading session and instrument state example flow
28-May-2021	TBD	0.6	• Order Book Directory: - Update AllowTTF to not applicable - Update flow of CA code and split flag to be sent out after CA process (same as current system) - Other Sign: add L flag for Long order cancellation due to CA • Order Book State: Add possible values of active state • Trade Ticker: update Aggressor N for None • iNAV: correct field length • ITCH Overview: Add Message Overview • Reference Data: Add Reference Data Overview • Add more description for Price Limit, Order Book State, Halt Information, MBP, Trade Ticker, Trade Statistics, Market Statistics and others as per highlighted • ITCH example message flow: update TFEX Price Break flow
2-Jul-2021	TBD	1.0	• SoupBin TCP Protocol: Cancel End of session message and update description • MoldUDP64 Protocol: Add picture of Downstream packet • Revise System Architecture section and add ITCH MDFA in Market Data Gateway • Add ITCH Client Development section • System Event Message: update EventCode=O not available for ITCH MDFA • Order Book Directory: - Financial products: add UL for Underlying and others to support new products - Par and Contract size: correct data type to Price

Release Date	Effective Date	Version	Change Summary
			- Status: add Issued and Delisted • Order Book State: add STARTUP_E session and update description • MBP: - Number Of Deletes: correct data type to numeric - Price: send min integer for Market Price • Trade: update possible values of Equity trade report code • Index Price: update Open index to open of the day • Open Interest: add missing Timestamp field • Market Statistics and Reference Price: Add more description • Market Announcement: Add example messages and update descriptions • ITCH example message flow: Add Reference Data and Corporate Action
13-Sep-2021	TBD	1.1	• ITCH Market Data System Overview: add ITCH Client Connection section and update descriptions • ITCH Client Development: update descriptions - Add ITCH Connection Flow - Update ITCH Message Flow - ITCH Multicast Client: add recommended configuration • Add ITCH Message Overview • Order Book Directory: - Remove financial products CBC and CBP - Update more descriptions • Trade Ticker: Classify Derivatives trade report code • Reference Price: add sending IPO price via priceType=5 • Add more descriptions and example messages for System Event, Equilibrium Price, Trade Statistics and News • SoupBinTCP example message flow: update values in example
1-Oct-2021	TBD	1.2	• Enhancement: Glimpse server can support snapshot data for MDFA Price Feed (no impact on MDFA News) - System Architecture

Release Date	Effective Date	Version	Change Summary
			- ITCH MDFA Client Connection - ITCH Client Development for MDFA Price Feed and News - ITCH Message Flow for MDFA Price Feed and News - ITCH Message Overview • Multicast Receiver: Add more example codes • Order Book State: Add possible value, DAYCLOSE_D, for TFEX market • Reference Price: Add more descriptions
15-Nov-2021	TBD	1.3	• Update using Sequence Number from "G" message by removing +1 - Glimpse Message Format - ITCH Client Development: ICH MDF and MDFA for Price Feed Client, and ITCH Multicast Client • System Architecture: Add Redundant Connection pictures • Update UDP payload size maximum to 64KB - MoldUDP64 Protocol – Downstream packet - ITCH Client Development: ITCH Multicast Client • ITCH Java Example – SoupBinTCP: Add example of logout method • Reference Price: Add more description for Ever last price • Index Price: Update Traded value field sent out as 2 decimal digits
4-Feb-2022	TBD	1.4	• ITCH Client Development – Add more descriptions about reconnecting • Order Book State: - Add possible values, TRANSITION1_D, TRANSITION1_2 and TRANSITION3_D; PAUSE_E and PAUSE_D - Add missing possible values, CIRCUIT_BREAKER_E and CIRCUIT_BREAKER_D - Correct EXPIRED_E code • Halt Information: - Add PAUSE_E and PAUSE_D - Correct EXPIRED_E code - And more details for instrument state • MBP: Update description of Maximum level field • Equilibrium price and Trade Ticker: Add more description

Release Date	Effective Date	Version	Change Summary
			• Trade Ticker: Update trade report code possible values for Derivatives market • Trade Statistics and Index Price: Correct turnover value fields to be Price data type (meaning that value fields can contain decimal places) • Index Price: Correct missing timestamp field and add more descriptions • News: Add possible values of newsType for F56 report type • ITCH example message flow: Update some CA cases
14-Feb-2022	TBD	1.5	• MoldUDP64 Protocol: Recall End of session in Downstream packet
23-May-2022	TBD	1.6	• Snapshot: Seconds message when switch from Snapshot to real-time market data
6-Jun-2022	TBD	1.6.1	• Clarify MBP processing with multiple updates at 7.6.1, 8.2.4, and 8.3.2
13-Jun-2022	TBD	1.6.2	• Correct MBP processing with multiple updates at 8.3.2
18-July-2022	TBD	1.6.3	• Reference Price: Add more description
15-Aug-2022	TBD	1.6.4	• Add more description about information of new underlying order book ID at 7.4.1 and 8.2.1
19-Sep-2022	TBD	1.6.5	• Order Book Directory: Update usage of expiration date field for equity market
4-Nov-2022	TBD	1.6.6	• Order Book Directory: Add guideline for Sector Code field • Clarification for start of the day procedure at 6.1.1
27-Feb-2023	TBD	1.6.7	• Clarify the state of index orderbooks