Document title

NYSE OPENBOOK AGGREGATED EXCHANGE DATA PUBLISHER (XDP) CLIENT SPECIFICATION

NYSE OPENBOOK AGGREGATED & NYSE MKT OPENBOOK AGGREGATED

Version	Date
1.3a	15 Oct 2015

PREFACE

DOCUMENT HISTORY

The following table provides a description of all changes to this document.

VERSION NO.	DATE	CHANGE DESCRIPTION
0.1	12/1/09	Initial draft
0.2	1/25/10	Changes based on development feedback
0.3	1/26/10	Changes to the Heartbeat messages
0.3	3/8/10	Changes to delivery flag Addition of the Time Reference Message Changes to Control Messages
0.4	3/22/10	Changes to Heartbeat Delivery flag value Clarification on the Symbol Index Map Assignment on Channel ID on IP addresses
0.5	3/23/10	Addition of Retrans Request Source IPs
0.6	3/26/10	Changes to the Packet Header & TCP Messages
0.7	3/30/10	Changes to the control messages
1.0	4/19/10	Initial version
1.0b	6/23/2010	■ Removed Firm index mapping messages. ■ Added "LastSeqNum" field to the Refresh Header message ■ Corrected Bloomberg code field sizes on the Symbol Index mapping message
1.2	12/15/2010	■ Removed Order ID description ■ Removed Time Reference description ■ Removed control message types not used on this feed ■ Corrected update message example on 2.1.6 ■ Added Symbol Index Mapping FTP file
1.2a	08/12/2011	Updated artwork and minor edits throughout
1.3	05/30/2012	Changed name to NYSE MKT throughout
	09/03/2012	Minor correction made to 'Date' column for previous entry.
1.3 a	10/15/2015	Updated the location of Symbol Index Mapping FTP

REFERENCE MATERIAL

The following lists the associated documents, which either should be read in conjunction with this document or which provide other relevant information for the user:

- SFTI US Technical Specification
- SFTI US Customer Guide
- NYSE Symbology

CONTACT INFORMATION

For technical support please contact the Service Desk:

- Telephone: +1 212 383 3640 (International)
- Telephone: 866 873 7422 (Toll free, US only)
- Email: service.desk@nyx.com

FURTHER INFORMATION

- For additional product information, visit:

  http://www.nyxdata.com/Data-Products/NYSE-OpenBook

  http://www.nyxdata.com/Data-Products/NYSE-MKT-OpenBook
- For updated capacity figures, visit our capacity pages at: http://www.nyxdata.com/capacity
- For details of IP addresses, visit our IP address pages at: http://www.nyxdata.com/ipaddresses
- For a full glossary, visit: http://www.nyxdata.com/glossary/

CONTENTS

1. MARKET DATA PROCESSING INFORMATION

1.1 XDP MARKET DATA OVERVIEW

The XDP for the OpenBook Aggregated data feed has the following high-level features:

- Multicast technology
- High Availability
- Ultra-low latency
- Reliable network solution
- High level of scalability

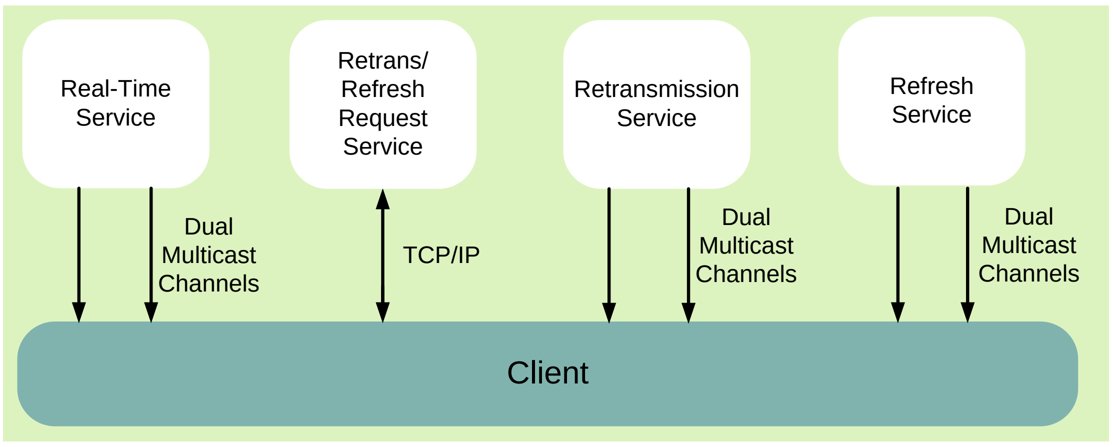
This chapter provides detailed information about the features of the feed, to support the development of client applications by Traders, Independent Software Vendors, and Quote Vendors.

The following chapters of this document provide details that are specific to each of the market data sets, including formats for each message type.

1.2 ACCESS TO MARKET DATA

Clients connect to multicast addresses for the real time market data messages and refresh data, and can also connect to a TCP/IP server for packet retransmissions (only for Symbol Mapping data – to be introduced in the near future). Requests for retransmission and refresh are performed via TCP/IP.

Figure 1 Access to Market Data



1.2.1 Real Time Market Data

Real-time market data is message-based over the UDP IP protocol with fixed-length binary and ASCII fields.

It uses the push-based publishing model. This means that data will be published based on its availability. Once an update is available, it will be published to the appropriate multicast group.

For capacity reasons, market data can be split across a number of multicast groups which are organized into predefined data sets (Channels).

Each multicast group will deliver a set of data for a certain market segment.

The client application will be responsible for issuing multicast subscriptions to one or more of the multicast groups assigned to each product.

The process of subscribing to a multicast group ID is also known as ‘joining’ a multicast group. Upon session termination, the client’s host system should issue an ‘unjoin’ message. This will terminate delivery of data to that host’s local network. If a client application terminates without issuing an ‘unjoin’ message, the network will eventually issue a ‘timeout’ for the multicast group subscription that will automatically terminate delivery of the multicast packets to the host’s local network.

The ‘join’ and ‘unjoin’ processes are standard functions. No specific instructions are provided here, as they are specific to the user’s operating system and programming language.

1.2.2 Retransmission Functionality

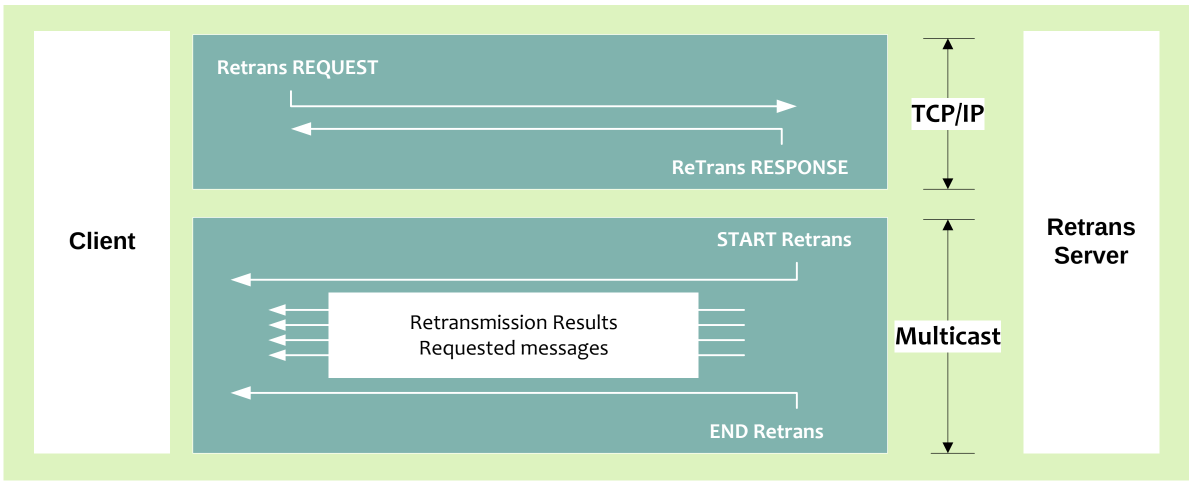
The retransmission functionality is designed to allow the user to recapture a small number of missed messages.

It is not intended that clients use the retransmission functionality to recover data after long outages or on late start up. Accordingly, the number of messages that the user can request by each Source ID is limited. The number of retransmission requests permitted per user is also limited per day.

The client makes a TCP/IP connection with the retransmission server, and receives the requested messages also via the Multicast channel.

The following diagram shows the sequence of messages and the transport protocols employed when making a retransmission request.

Figure 2 Retransmission Functionality



The retransmission request will include a Source ID (username) which will be validated by the XDP system. It is important to note that only one Source ID can be used per application session.

The retransmission request may be rejected for any of the following reasons:

- Invalid Source ID (username)
- Invalid requested sequence numbers
- Incorrectly formatted request packet
- Maximum number of packets by request exceeded
- Messages are no longer in cache
- Total number of messages requested in the current day exceeds the predefined system limit

- Number of retransmission requests in the current day exceeds the predefined system limit

In the case of such a failure, the user will receive an error message to advise of the reason for failure.

If the reason for failure is exceeding a predefined system limit, clients are asked to not make any further requests. If further retransmissions are required, the client should contact NYSE .

1.2.3 Refresh Functionality

The Refresh Server supplies (on demand) a snapshot of the current state of the security.

The refresh server is designed to allow the user to update the market state within their applications before restarting in real-time, following a data outage or late start.

The client makes a TCP/IP connection to the Refresh Server for requesting the refresh, while also joining the refresh multicast channels for receiving the refresh messages.

The retrans/refresh server will respond to a request with a Request Response message via TCP/IP to indicate whether the request was accepted or rejected.

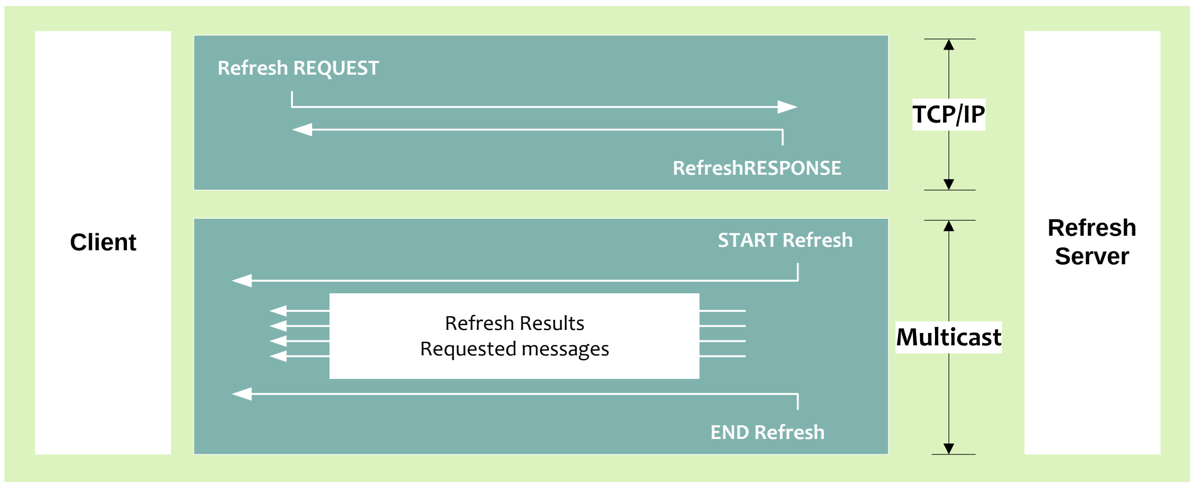
The refresh messages will then be sent on the multicast channels separate from the retransmission channels. This will be preceded by a Start of Refresh indicator in the Delivery flag field of the Packet Header and followed by an End of Refresh indicator in the Delivery flag field of the Packet Header. Each refresh packet will begin with a Refresh Header (Msg Type ‘35’) that identifies what the current packet is in the sequence of the refresh updates, and what the total number of refresh packets is. No dedicated retransmission service is available for the refresh, if message loss is detected, clients should submit another refresh request.

The refresh request will include a Source ID (username), which will be validated by the exchange system. It is important to note that only one Source ID can be used per application session.

A pair of refresh multicast channels will be provided for each corresponding real-time service. The contents of the refresh and message formats will correspond to the contents and message formats contained in the appropriate real-time service.

The following diagram shows the sequence of messages and the transport protocols employed when making a refresh request.

Figure 3 Refresh Functionality



The refresh request may be rejected for any of the following reasons:

- Invalid Source ID
- Invalid Product ID
- Incorrectly formatted request packet
- Incorrect message type sent
- Total number of refreshes requested in the current day exceeds the predefined system limit
- Rejected due to unavailability of refresh data

In the case of such a failure, the user will receive an error message to advise of the reason for failure.

If the reason for failure is exceeding a predefined system limit, clients are asked to not make any further requests. If further refreshes are required, the client should contact NYSE .

1.3 PROCESSING GUIDELINES

1.3.1 General Processing Notes

The following processing notes apply to all messages:

- All fields will be sent for every message.
- Only field values will appear in the published messages (e.g. no names or 'tags' will appear in the message).
- The field names that appear in the message format documents are for reference purposes only.
- All the fields are contiguous, with reserved fields for alignment issues.
- All field sizes are fixed and constant.
- The message sizes may vary.
- Binary fields are provided in Little-Endian format.
- ASCII string fields are left aligned and null padded.
- Segmentation of messages across packets will not be supported. This means a message will never straddle a packet boundary.

1.3.2 FAST Optimization

FAST optimization will **not** be used for OpenBook Aggregated.

1.3.3 Packet Structure

All packets of data sent on the XDP feed will have a common packet header followed by one or more messages (with the exception of some technical packets that do not contain any messages).

The packet header format is the same for all packets, and contains packet length, number of messages within the packet, packet sequence number, etc.

The format of each message in the packet depends on message type, but each message will start with message size and message type.

The maximum length of a packet is 1500 bytes.

A packet will only ever contain complete messages. A single message will never straddle multiple packets.

The message size will never exceed the maximum packet length (less the packet header size).

PACKET HEADER	MESSAGE 1	MESSAGE 2	...	MESSAGE N

The packet header provides information including the total packet length, a packet sequence number, the number of messages within the packet. The format is as follows:

Table 1 Packet Header Fields

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	DESCRIPTION
PktSize	0	2	Binary Integer	This field indicates the size of the packet including the 16 -byte packet header in bytes
DeliveryFlag	2	1	Binary Integer	A flag that indicates whether this is an original, retransmitted, or 'replayed' message. Valid values include: '1' – Heartbeat Message '11' – Original Message '12' – Sequence Number Reset Message '13' – Only one packet in retransmission update Uncompressed '14' – Start of retrans Update Uncompressed '15' – Part of a retransmission sequence Uncompressed '16' – End of retrans Update Uncompressed '17' – Only one packet in Refresh update Uncompressed '18 – Start of Refresh Update Uncompressed '19' – Part of a Refresh sequence Uncompressed '20' – End of Refresh Update Uncompressed '21' – Message Unavailable '31' – Original Message compressed (Fast) '32' – Sequence Number Message (Fast) '33' - Only one packet in retransmission update compressed (Fast) '34' – Start of retrans Update compressed

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	DESCRIPTION
				(Fast) ‘35’ – Part of a retransmission sequence compressed (Fast) ‘36’ – End of retrans Update compressed (Fast) ‘37’ – Only one packet in Refresh update compressed (Fast) ‘38’ – Start of Refresh Update compressed (Fast) ‘39’ – Part of a Refresh sequence compressed (Fast) ‘40’ – End of Refresh Update compressed (Fast) ‘41’ – Message Unavailable
NumberMsgs	3	1	Binary Integer	This field contains the number of messages in the packet and also used to determine the next sequence number. See Sequence Numbers for more information.
SeqNum	4	4	Binary Integer	This field contains the message sequence number assigned by XDP for each product. It is used for gap detection, unique for each broadcast stream (except if reset during the day). See Sequence Numbers for more information.
SendTime	8	4	Binary Integer	This field specifies the time when the message was generated in XDP. The number represents the number of seconds in UTC time (EPOCH)
SendTimeNS	12	4	Binary Integer	This field specifies the number represents the nanosecond portion of UTC time (since EPOCH)

The format of each message within a packet will vary according to message type.

However, regardless of the message type, each message will start with a message header consisting of 2 fields: a 2-byte message length, followed by a 2-byte message type.

Table 2 Message Header Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	-	2	Binary Integer	This field indicates the size of the message body in bytes including this field.
MsgType	-	2	Binary Integer	This field identifies the type of message

1.3.4 Message Size Field Processing

Customers should not hard-code message sizes in feed handlers; instead the feed handler should use the MsgSize field to determine where the next message in the packet begins. This allows the XDP format to accommodate the different market needs for data content and allow the format to be more agile.

For example: The following table demonstrates a message type used by NYSE, NYSE MKT and NYSE Arca. Each market shares the same fields until byte 28. If you were only taking the NYSE Arca version of the feed, you would read "28 bytes" in the Msg Size field, and the next message will begin on "Byte 29". If you were reading an NYSE message, then the Msg Size field will indicate that the next message begin after the 35th Byte.

Table 3 Message Size Field Processing

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message. **NYSE – 34 Bytes** **NYSE MKT – 34 bytes** **NYSE Arca - 28 bytes**
Msg Type	2	2	Binary Integer	This field identifies the type of message. Message '100' – Add Order Message
SourceTimeNS	4	4	Binary Integer	This field represents the nanosecond offset from the time reference second in UTC time (EPOCH).
SymbolIndex	8	4	Binary Integer	This field identifies the numerical representation of the symbol.
OrderID	12	4	Binary Integer	The Order ID identifies a unique order.
Price	16	4	Binary Integer	This field specifies the price of the order, see Price Formats. Use the

Look at the Msg Size field to know where the next record will be.

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
				Price scale from the symbol mapping index.
Volume	20	4	Binary Integer	This field contains the size of the order.
Side	24	1	ASCII Character	This field indicates the side of the order Buy/Sell. Valid values: ■ 'B' – Buy ■ 'S' – Sell
OrderIDGTCIndicator	25	1	Binary Integer	This field specifies if Trade Order ID is a GTC order: ■ '0' – Day Order ■ '1'- GTC Order
TradeSession	26	1	Binary Integer	Bit Shift values: 0x01 Ok for morning hours 0x02 Ok for national hours (core) 0x04 Ok for late hours
QuoteCondition	27	1	Binary Integer	The current quote condition for the symbol. The quote condition shall be blank if no quote condition exists (example when the Book is fast).
AggregatedVolume	28	4	Binary Integer	This field is the Total Volume at the Price Point after the event has been applied.
NumOrders	32	2	Binary Integer	This field contains the number of orders at the current price point.

Market-specific content

The variable message size allows that the feed handler can also insulate your code from any future field additions that you may not want.

Table 4 Example 1

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message. NYSE – 16 Bytes NYSE MKT– 16 bytes NYSE Arca - 16 bytes
Msg Type	2	2	Binary Integer	This field identifies the type of message Message '999' – Price message example
SourceTimeNS	4	4	Binary Integer	This field represents the nanosecond offset from the time reference second in UTC time (EPOCH).
SymbolIndex	8	4	Binary Integer	This field identifies the numerical representation of the symbol.
Price	12	4	Binary Integer	This field specifies the price of the order see Price Formats. Use the Price scale from the symbol mapping index.

Look at the Msg Size field to know where the next record will be.

Now the new format adds a new 4-byte volume field adjusting the message size to 20 bytes.

The feed handler code automatically is prepared for the 20-byte format without any work, allowing for the receiving application to either continue to read on the first 16 bytes passed to it or develop to read the new field at your own pace.

Table 5 Example 2

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message. NYSE – 20 Bytes NYSE MKT– 20 bytes NYSE Arca – 20 bytes
Msg Type	2	2	Binary Integer	This field identifies the type of message Message '999' – Price message

Look at the Msg Size field to know where the next record will be.

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
				example
SourceTimeNS	4	4	Binary Integer	This field represents the nanosecond offset from the time reference second in UTC time (EPOCH).
SymbolIndex	8	4	Binary Integer	This field identifies the numerical representation of the symbol.
Price	12	4	Binary Integer	This field specifies the price of the order, see Price Formats. Use the Price scale from the symbol mapping index.
Volume	16	4	Binary Integer	This field contains the size of the order.

1.3.5 Date and Time Conventions

Dates and Times use UTC (Universal Time, Coordinated) EPOCH. For example Wednesday 12/1/09 22:05:17.000 UTC is indicated as 1259791537.

1.3.6 Product Ids

Listed below are the Product IDs for the existing feeds throughout NYSE . The NYSE, MKT, and NYSE Arca IDs have been changed so that they are unique.

Table 6 Product Ids

EXCHANGE	PRODUCTID	DESCRIPTION
NYSE	1	NYSE OpenBook Aggregated
MKT	50	NYSE MKT OpenBook Aggregated
NYSE Arca	150	ArcaBook Aggregated (when available)

1.3.7 Sequence Numbers

All messages conform to the message level sequencing.. Each channel A, B, C, D, etc shall have its own sequence number. This will allow recipients to identify 'gaps' or duplicates in each message sequence number and, if appropriate, reconcile them (line arbitrage) with the primary/secondary multicast groups or request retransmission of the missing/corrupted data packets.

Clients can use sequence numbers to determine the following:

The next SeqNumbers are calculated by adding the current SeqNum+NumMsgs:

Table 7 Calculating Sequence Numbers

PACKET	SEQNUM	NUMMSGS
Packet 1	1	4
Packet 2	5	2
Packet 3	7	1
Packet 4	8	3
Packet 5	11	1

If the client drops the first five packets they would request a gap fill for messages 1-11. Instead of sending the original five packets we would send one packet with 11 messages (assuming the total number of messages fit within the 1500 byte packet size, otherwise the delivery flag will indicate that the update will span multiple packets).

1.3.8 Line Arbitration

Client applications should check the Sequence Number (SN) for every packet received.

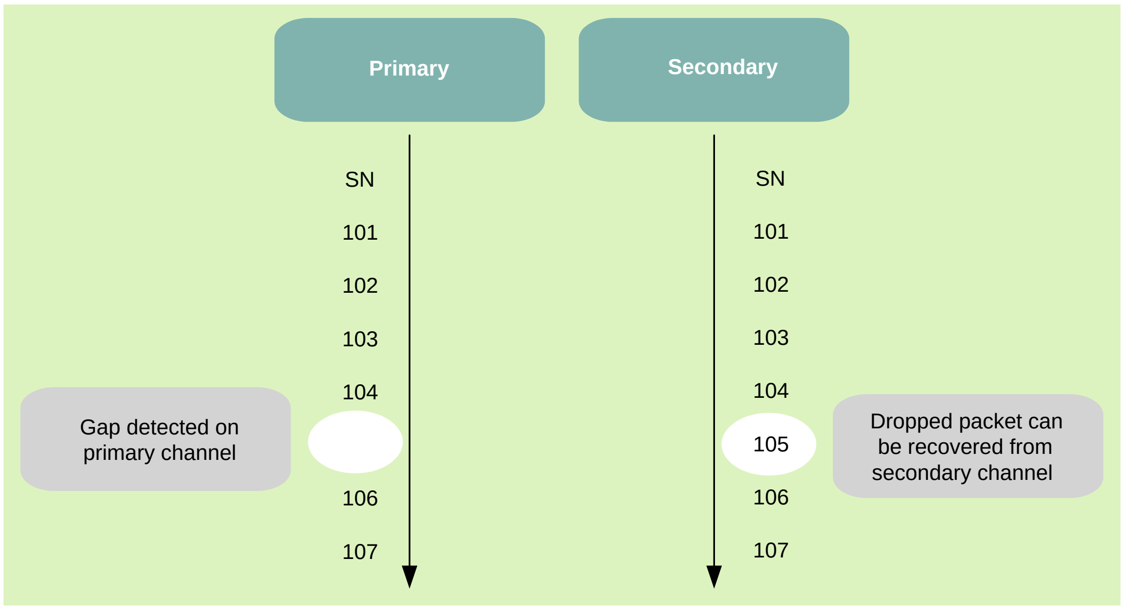
SNs are unique for each channel however they do not increase monotonically. The sequence number increase as shown in the table above.

Line A and line B are identical in terms of:

- Packet contents
- SNs
- Sequence in which packets are sent.

Client applications should listen to both channels in real time. Clients should look at packets coming from both lines and process the ones that arrive first, regardless of whether they came from line A or line B. It is advisable to apply the 'first come – first served' rule.

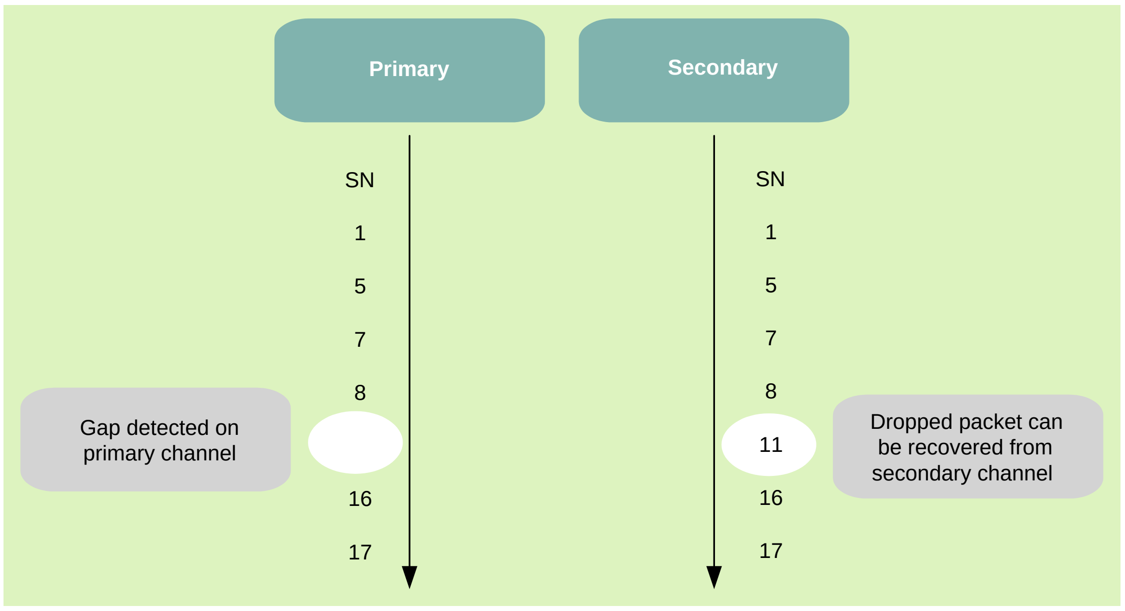
Figure 4 Packet Header Sequence Numbers (ssuming one message per packet)


Primary
Secondary
SN
101
102
103
104
106
107
SN
101
102
103
104
105
106
107
Gap detected on primary channel
Dropped packet can be recovered from secondary channel

Assuming the message sequence below:

FIELD NAME	SEQNUM	NUMMSGS	NEXT EXPECTED SEQNUM
Message 1	1	4	5
Message 2	5	2	7
Message 3	7	1	8
Message 4	8	3	11
Message 5	11	1	12
Message 6	12	4	16
Message 7	16	1	17

Figure 5 Detecting Missed Packets


Primary
Secondary
SN
1
5
7
8
16
17
SN
1
5
7
8
11
16
17
Gap detected on primary channel
Dropped packet can be recovered from secondary channel

1.3.9 Detecting and Recovering Missed Data

UDP is an 'unreliable' protocol and therefore may drop packets. All multicast data is provided over dual channels (line A and line B).

The OpenBook Aggregated feed provides three different mechanisms for recovering missed data:

- Line arbitration – using dual multicast channels
- Retransmission server – recovery of a limited number of packets
- Refresh server – snapshot of current market state

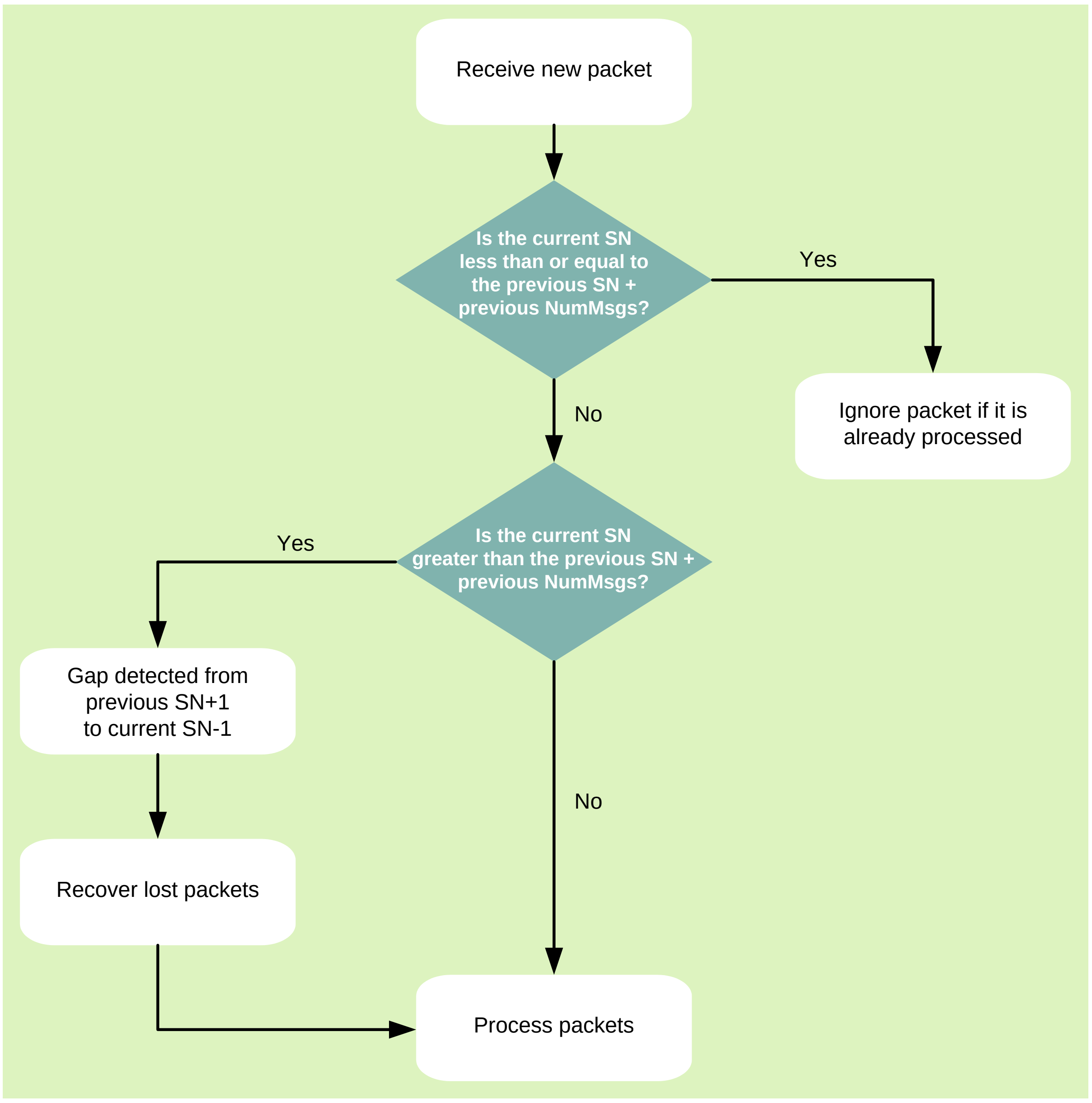
These mechanisms should be used as follows:

Table 8 Recovery Mechanisms

EVENT	ACTION
Packet lost on one of the two lines	Try to recover data from the other line with a configurable timeout
Dropped packet(s) on both line A and line B	Recover dropped packet(s) from retransmission server
Late start up or extended intraday outage	Request a refresh of the current market state and then continue with real time messages

The following diagram illustrates how the SN should be used to detect gaps in the feed.

Figure 6 Gap Detection


Receive new packet
Is the current SN less than or equal to the previous SN + previous NumMsgs?
Yes
Ignore packet if it is already processed
No
Is the current SN greater than the previous SN + previous NumMsgs?
Yes
Gap detected from previous SN+1 to current SN-1
Recover lost packets
No
Process packets

1.3.10 Retransmission Server

If a packet is lost from both line A and line B, clients then make a TCP/IP request to have the packets resent. Packets are resent from the Retransmission Server.

After a client establishes a TCP/IP connection, the Retransmission Server will periodically send heartbeat request messages to the client. Clients must respond to this request with a heartbeat response within a specific timeframe – otherwise the Retransmission Server will close the connection.

The client makes a TCP/IP connection to the Retransmission Server for both requesting and receiving retransmitted packets (only applies to Symbol Index mapping – to be available in the near future).

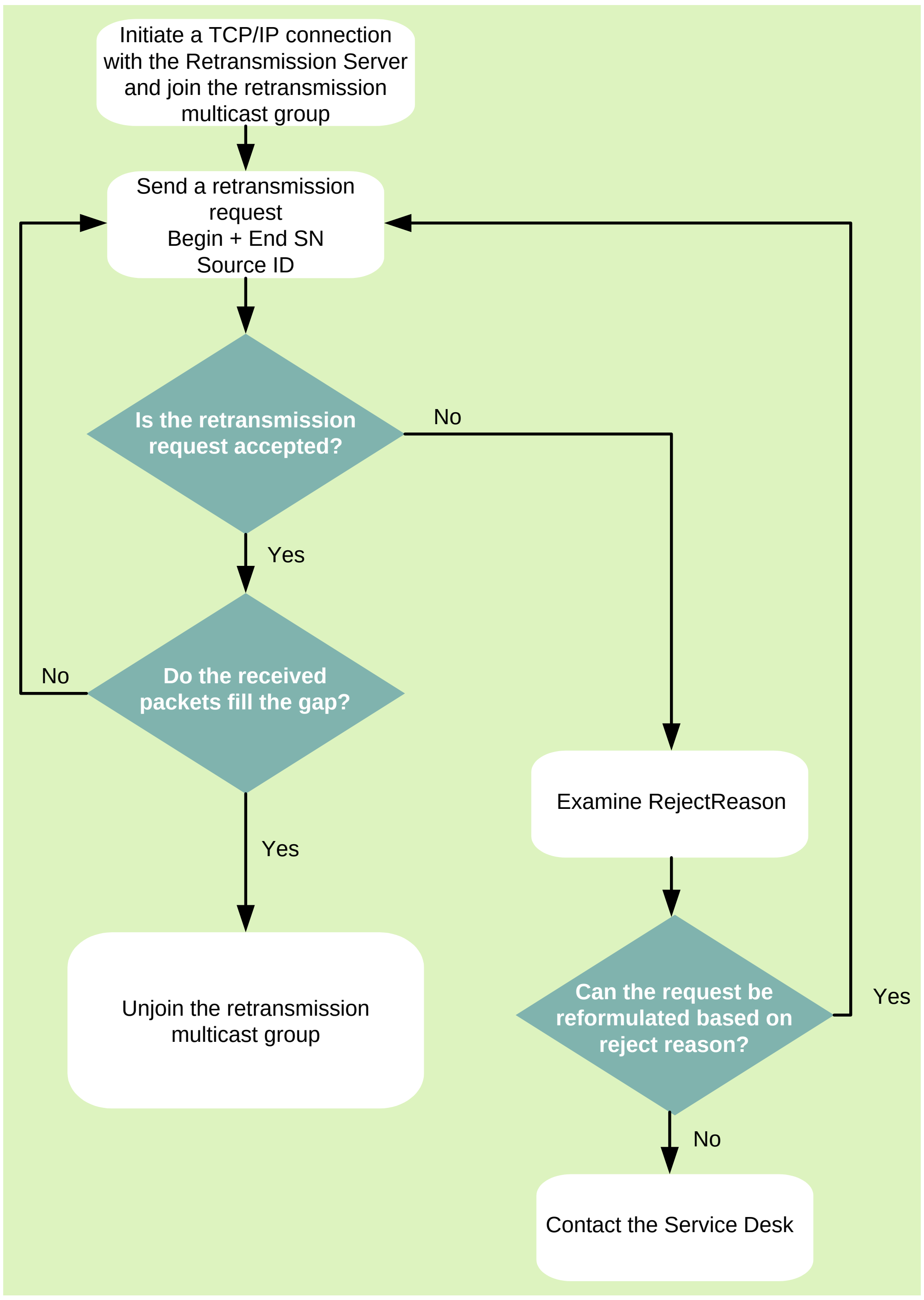
Retransmission requests should contain a Start SN, an End SN and a Source ID. The Source ID identifies the client application, and will be supplied by the exchange. The request can be rejected for a number of reasons as defined later in this document.

The number of retransmissions allowed per client per day is limited.

The length of each retransmission is limited to a pre-defined number of messages.

The following diagram illustrates the process of requesting dropped packets from the retransmission server:

Figure 7 Requesting Retransmission of Dropped Packets


Initiate a TCP/IP connection with the Retransmission Server and join the retransmission multicast group
Send a retransmission request Begin + End SN Source ID
Is the retransmission request accepted?
No
Yes
Do the received packets fill the gap?
No
Yes
Examine RejectReason
Can the request be reformulated based on reject reason?
Yes
No
Unjoin the retransmission multicast group
Contact the Service Desk

1.3.11 Refresh Server

The Refresh Server supplies (on demand) a snapshot of the current state of the security.

The refresh server is designed to allow the user to update the market state within their applications before restarting in real time, following a data outage or late start.

The client makes a TCP/IP connection to the Refresh Server for requesting the refresh, while also joining the refresh multicast channels for receiving the refresh messages.

The retrans/refresh server will respond to a request with a Request Response message via TCP/IP to indicate whether the request was accepted or rejected.

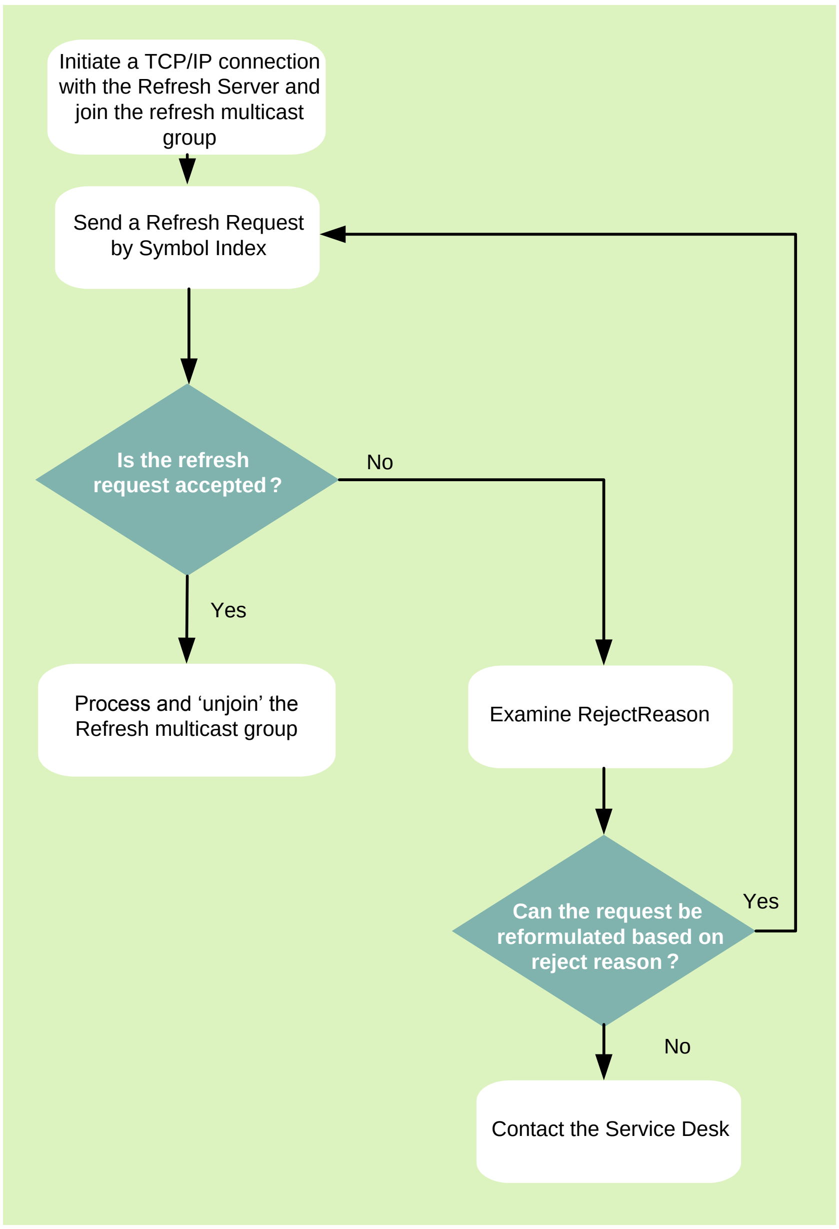
The refresh messages will then be sent on the multicast channels. This will be preceded by a Start of Refresh indicator in the Delivery flag field of the Packet Header and followed by an End of Refresh indicator in the Delivery flag field of the Packet Header. Each refresh packet will begin with a Refresh Header (Msg Type '35') that identifies what the current packet is in the sequence of the refresh updates, and what the total number of refresh packets is. No dedicated retransmission service is available for the refresh, if message loss is detected, clients should submit another refresh request.

The refresh request will include a Source ID (username), which will be validated by the exchange system. It is important to note that only one Source ID can be used per application session.

A pair of refresh multicast channels will be provided for each corresponding real-time service. The contents of the refresh and message formats will correspond to the contents and message formats contained in the appropriate real-time service.

The following diagram shows the sequence of messages and the transport protocols employed when making a refresh request.

Figure 8 Requesting Refresh Information


Initiate a TCP/IP connection with the Refresh Server and join the refresh multicast group
Send a Refresh Request by Symbol Index
Is the refresh request accepted ?
No
Yes
Process and 'unjoin' the Refresh multicast group
Examine RejectReason
Can the request be reformulated based on reject reason ?
Yes
No
Contact the Service Desk

1.3.12 Price Formats

All price fields are sent in integer format.

Prices in this feed are represented by two fields, separating the denominator and the numerator. All prices in the feed share a common denominator (unless otherwise specified), which is represented in the PriceScaleCode.

The PriceScaleCode field value represents the common denominator for the following formula:

$$\text{Price} = \frac{Numerator}{10^{PriceScaleCode}}$$

For example, a price of 27.56 is represented by a Numerator of 2756 and a PriceScaleCode equals to 2.

1.3.13 Symbol Mapping

To ensure high throughput and low latency, symbols are identified using a Symbol Index message (msg type '3'). This is an ordered list from 1 to N of all symbols per multicast group. Symbol Indices are unique for every symbol and do not change each trading day. New symbols are appended to the end of the symbol mapping index and symbols that are removed do not have their index number reused.

1.3.14 Symbol Index Mapping FTP

For customer would prefer to download the symbol index mapping from an FTP server, a subset of the index mapping information is made available via FTP at 12:30am EST every trading day at the following location.

- Pipe-delimited: ftp://ftp.nyxdata.com/NYSESymbolMapping/NYSESymbolMapping.txt
- XML format: ftp://ftp.nyxdata.com/NYSESymbolMapping/NYSESymbolMapping.xml
- File Format: See the following table.

Table 9 File Format

FIELD NAME	FORMAT	DESCRIPTION
Symbol	Text	This field contains the full symbol in NYSE Symbology. A sequence of characters representing the symbol, padded with NULLs. For more information, see *NYSE Symbology*
CQS Symbol	Text	This field contain the full symbol in CTS and UTP line format
SymbolIndex	Numeric	This field identifies the numerical representation of the symbol. See Symbol Mapping.
NYSE Market	Character	This field represents the exchange within NYSE trading the symbol. 'N' – NYSE 'P' – NYSE Arca 'A' – NYSE MKT
Listed Market	Character	This field represents the exchange where the symbol is listed 'N' – NYSE 'P' – NYSE Arca 'Q' – NASDAQ 'A' – NYSE MKT
UOT	Numeric	This field represents the stock symbol's unit of trade
SystemID	Numeric	This field represents the matching engine ID where the symbol is trading

FIELD NAME	FORMAT	DESCRIPTION
Bloomberg BSID	Alphanumeric	Unique Bloomberg identifier (BSID) including the Bloomberg source for all Bloomberg securities as an integer value. This field is used for B Pipe and subscription services.
Bloomberg Global ID	Alphanumeric	Unique global identifier assigned by Bloomberg for all securities across every asset class.

1.4 OPERATIONAL INFORMATION

The following measures are in place to safeguard against unexpected system failures.

1.4.1 Exchange System Failure

1.4.1.1 Dual Multicast Lines

Under normal operating conditions, the exchange system will send real-time messages to two unique multicast addresses. This provides clients with two redundant data feeds. The client application should be designed to handle the loss of one of the two multicast channels without any interruption to service.

1.4.1.2 TCP/IP Channels

TCP/IP channels are made available for retransmission and refresh requests and responses.

The user can choose to disconnect/reconnect in between requests. However if choosing to remain connected, the user will need to respond to heartbeat requests from the exchange.

1.4.1.3 High Availability

The High Availability (HA) functionality of the market data publisher is set up to ensure there is no loss of service for clients if there is any kind of outage in the exchange on the primary publisher, for example a hardware failure. The HA failover has been designed to be as transparent as possible for clients, as the connectivity in terms of multicast groups and ports will not change. However, clients should note that there are specific technical details that should be considered.

For details of retransmissions and refresh behavior that should be included as part of application logic, please refer to Processing.

1.4.2 Disaster Recovery Site

In order to mitigate any serious outage in the primary data center, a secondary data center is online in standby mode, in case of a serious incident.

Clients should ensure all configurations surrounding the secondary data center are included.

1.4.3 Client System Failure

Real-time market data will be made available on two different multicast groups. This offers clients the possibility to set up more than one receiving system processing the same data. In the event of a client system failure, the backup client system should continue to process the real-time data sent on the second multicast group.

1.4.4 Gap Detection

The XDP feed provides a unique, sequential packet sequence number for each multicast channel. This will allow recipients to identify 'gaps' in the message sequence and, if appropriate, reconcile them 'locally' with an alternate channel or request retransmission of the missing/corrupted data packet.

For more details on gap detection, see Line Arbitration.

1.4.5 System Behavior on Start and Restart

At the start of the day, the feed will send the following messages:

- 10 Heartbeat Messages with Delivery Flag set to 1, sequence number is 1 (next expected seqnum)
- Sequence Number Reset message (Msg Type '1'), sequence number is set to 1
- Symbol Mapping messages for securities traded on the market; (Msg Type '3')
- A snapshot message for each instrument: (Msg Type '110')

Note that this sequence will also be followed on system recovery following a failure. Therefore in exceptional circumstances a user may see this during the trading day.

2. OPENBOOK AGGREGATED MESSAGE SPECIFICATIONS

2.1 ORDERBOOK INFORMATION

2.1.1 Overview

OpenBook Aggregated provides a one-second snapshot of the Exchange's limit-order book for all traded securities. OpenBook Aggregated lets traders see aggregate limit-order volume at every bid and offer price. The data feed uses the push-based publishing model. This means that data will be published based on its availability. Once information is available, it will be published to clients.

2.1.2 Publication Times

Table 10 Normal Trading Days

MSGTYPE	DESCRIPTION	NYSE	MKT
110	Snapshot Message	1:00am – 4:00pm	1:00am – 4:00pm
111	Delta Message	7:30am – 4:00pm	7:30am – 4:00pm

Table 11 Early Closing Days

MSGTYPE	DESCRIPTION	NYSE	MKT
110	Snapshot Message	1:00am – 1:00pm	1:00am – 1:00pm
111	Delta Message	7:30am – 1:00pm	7:30am – 1:00pm

2.1.3 Control Message Types Used in the Data Feed

See Operational Information for message type details.

Table 12 Control Message Types

MSGTYPE	DESCRIPTION	NYSE	MKT
1	Sequence Number Reset	x	x
2	Time Reference Message		
3	Symbol Index Mapping	x	x
5	Option Series Index Mapping		
10	Retransmission Request Message	x	x
11	Request Response Message	x	x
12	Heartbeat Response Message	x	x

MSGTYPE	DESCRIPTION	NYSE	MKT
13	Symbol Index Mapping Request Message	x	x
15	Refresh Request Message	x	x
31	Message Unavailable	x	x
32	Symbol Clear		
33	Trading Session Change		
34	Security Status Message		
35	Refresh Header Message	x	x

2.1.4 OrderBook Snapshot/Refresh Message (Msg Type '110')

2.1.4.1 Message Overview

A 'Snapshot Update' message provides the full depth of book for the symbol.

2.1.4.2 Message Sending Rules

A Snapshot message 110 message is sent as a result of one of the following events:

- Start of day
- Refresh Request (sent only over refresh multicast group)
- System Restart

2.1.4.3 Message Structure

The table below describes the body fields of an OrderBook Snapshot message, MsgType = '110'.

Table 13 OrderBook Snapshot Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message. NYSE – 49 Bytes NYSE MKT – 49 Bytes
Msg Type	2	2	Binary Integer	This field identifies the type of message '110 – Snapshot Message
SourceTime	4	4	Binary Integer	This field specifies the time when the msg was generated in the order book. The number represents the number of seconds in UTC time (EPOCH)
SourceTimeNS	8	4	Binary Integer	This field represents the nanosecond portion within the second in UTC time (EPOCH)

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SymbolIndex	12	4	Binary Integer	This field identifies the numerical representation of the symbol. See Symbol Mapping
Ultra LastSeqNum	16	4	Binary Integer	To ease recovery of OpenBook Ultra customers, This field contains the last sequence number of the last message sent out of the OpenBook Ultra feed.
Symbol	20	11	ASCII String	This field contains the full symbol in NYSE Symbology. A sequence of characters representing the symbol, padded with NULLs. For more information, see *NYSE Symbology*
PriceScaleCode	31	1	Binary Integer	Price scale for price conversion of the symbol. See Price Formats.
TradingStatus	32	1	ASCII Character	This field indicates the market condition of the security. Valid Values: 'P' - Pre-Opening for messages sent before the stock is opened on a trade or quote 'O' - The stock has opened or re-opened 'C' - The stock was closed from the Closing template 'H' - The stock is halted during a trading halt and has not resumed
RemainingCount	33	2	Binary Integer	Number of updates remaining in the event.
MPV	35	2	Binary Integer	This field contains the minimum price variation
UpdateCount	37	1	Binary Integer	Number of updates. Indicates number of times the following group of fields (Price, Volume, Side, NumOrders) will be repeated in the message. Default value is **"1"**
>Price	38	4	Binary Integer	This field contains the price point.. See Price Formats. Use the Price scale from the symbol mapping index.
>Volume	42	4	Binary Integer	This field contains the total interest quantity in shares at a price point

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
>Side	46	1	ASCII Character	This field indicates the side of the order Buy/sell. Valid Values: 'B' – Buy 'S' – Sell
>NumOrders	47	2	Binary Integer	This field contains the number of orders at the current price point

2.1.4.4 Message Processing

A snapshot will be sent starting at a pre-defined time such as the start of day to provide details of any orders such as GTC orders (Good 'Til Cancelled), and exceptionally following an intraday service restart.

The following notes provide general guidelines for processing snapshot Messages. The client should not ascribe any importance to the order in which they are presented.

- Snapshot update Messages that span multiple packets must be processed as one complete message.
- For Snapshot update Messages that span multiple packets, if a packet is lost, then the whole message should be considered lost.
- Snapshot update messages contain all active price points regardless of prior period activity.

2.1.5 OrderBook Delta Update Message (Msg Type '111')

2.1.5.1 Message Overview

A 'Delta Update' message provides information about price updates for symbol.

2.1.5.2 Message Sending Rules

A Delta Update 111 message is sent as a result of one of the following events:

- New order
- New cancel
- New execution

2.1.5.3 Message Structure

The table below describes the body fields of a Delta Update message, MsgType = '111'.

Table 14 Delta Update Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message. NYSE– 35 Bytes NYSE MKT – 35 Bytes
Msg Type	2	2	Binary Integer	This field identifies the type of message '111 – Event Update Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceTime	4	4	Binary Integer	This field specifies the time when the msg was generated in the order book. The number represents the number of seconds in UTC time (EPOCH)
SourceTimeNS	8	4	Binary Integer	This field represents the nanosecond portion within the second in UTC time (EPOCH)
SymbolIndex	12	4	Binary Integer	This field identifies the numerical representation of the symbol. See Symbol Mapping
Ultra LastSeqNum	16	4	Binary Integer	To ease recovery of OpenBook Ultra customers, This field contains the last sequence number of the last message sent out of the OpenBook Ultra feed.
TradingStatus	20	1	ASCII Character	This field indicates the market condition of the security. Valid Values: 'P' - Pre-Opening for messages sent before the stock is opened on a trade or quote 'O' - The stock has opened or re-opened 'C' - The stock was closed from the Closing template 'H' - The stock is halted during a trading halt and has not resumed
RemainingCount	21	2	Binary Integer	Number of updates remaining in the event.
UpdateCount	23	1	Binary Integer	Number of updates. Indicates number of times the following group of fields (Price, Volume, Side, and NumOrders) will be repeated in the message. Default value is "1"
>Price	24	4	Binary Integer	This field contains the price point. See Price Format. Use the Price scale from the symbol mapping index.
>Volume	28	4	Binary Integer	This field contains the total interest quantity in shares at a price point
>Side	32	1	ASCII Character	This field indicates the side of the order Buy/sell.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				Valid Values: 'B' – Buy 'S' – Sell
>NumOrders	33	2	Binary Integer	This field contains the number of orders at the current price point

2.1.5.4 Message Processing

The following notes provide general guidelines for processing Delta Update Messages. The client should not ascribe any importance to the order in which they are presented.

- Delta Update Messages that span multiple packets must be processed as one complete message.
- For Delta Update Messages that span multiple packets, if a packet is lost, then the whole message should be considered lost.
- All price points containing a zero (0) quantity should be removed as an active price point.
- If no changes have occurred for a given symbol (e.g., an inactive stock) since the last publication, no Delta Update Message is generated.

2.1.6 Scenarios

2.1.6.1 Scenario 1 – single price point update for the same symbol

New events:

A 100 share Buy Order entered at $49.99 for stock ABC at 9:30:00am.

Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	500	49.99		
1	300	49.98		
3	600	49.97		

New State of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	1
2	600	49.99		
1	300	49.98		
3	600	49.97		

OpenBook Update Message

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	VALUE
PktSize	0	2	Binary Integer	51
DeliveryFlag	2	1	Binary Integer	1
NumberMsgs	3	1	Binary Integer	1
SeqNum	4	4	Binary Integer	1000
SendTime	8	4	Binary Integer	1259832600
SendTimeNS	12	4	Binary Integer	
Msg Size	0	2	Binary Integer	35
Msg Type	2	2	Binary Integer	111
SourceTime	4	4	Binary Integer	1259832600
SourceTimeNS	8	4	Binary Integer	0
SymbolIndex	12	4	Binary Integer	24005
Ultra LastSeqNum	16	4	Binary Integer	40000
TradingStatus	20	1	ASCII Character	O
RemainingCount	21	2	Binary Integer	0
UpdateCount	23	1	Binary Integer	1
Price	24	4	Binary Integer	4999
Volume	28	4	Binary Integer	600
Side	32	1	ASCII Character	B
NumOrders	33	2	Binary Integer	2

2.1.6.2 Scenario 2 – Multiple price point updates for the same symbol

New events:

A 100 share Buy Order entered at $49.99 for stock ABC at 9:30:00am.

A 400 share Sell Order entered at $50.00 for stock ABC at 9:30:00am.

Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	500	49.99		
1	300	49.98		
3	600	49.97		

New state of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	700	2
2	600	49.99		
1	300	49.98		
3	600	49.97		

OpenBook Update Message

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	VALUE
PktSize	0	2	Binary Integer	62
DeliveryFlag	2	1	Binary Integer	1
NumberMsgs	3	1	Binary Integer	1
SeqNum	4	4	Binary Integer	1000
SendTime	8	4	Binary Integer	1259832600
SendTimeNS	12	4	Binary Integer	
Msg Size	0	2	Binary Integer	46
Msg Type	2	2	Binary Integer	111
SourceTime	4	4	Binary Integer	1259832600
SourceTimeNS	8	4	Binary Integer	0
SymbolIndex	12	4	Binary Integer	24005
Ultra LastSeqNum	16	4	Binary Integer	40000
TradingStatus	20	1	ASCII Character	O
RemainingCount	21	2	Binary Integer	0
UpdateCount	23	1	Binary Integer	2
Price	24	4	Binary Integer	4999

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	VALUE
Volume	28	4	Binary Integer	600
Side	32	1	ASCII Character	B
NumOrders	33	2	Binary Integer	2
Price	35	4	Binary Integer	5000
Volume	39	4	Binary Integer	700
Side	43	1	ASCII Character	S
NumOrders	44	2	Binary Integer	2

2.1.6.3 Scenario 3 –single price point updates for the different symbols

New events:

100 share Buy Order entered at $49.99 for stock ABC.

400 share Sell Order entered at $30.00 for stock XYZ.

ABC - Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	**500**	49.99		
1	**300**	49.98		
3	**600**	49.97		

ABC - New state of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	2
2	**600**	49.99		
1	**300**	49.98		
3	**600**	49.97		

XYZ - Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		30.02	900	3
		30.01	600	2
		30.00	800	4
1	**100**	29.99		
1	**200**	29.98		
3	**300**	29.97		

XYZ - New state of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		30.02	900	3
		30.01	600	2
		30.00	1200	5
1	**100**	29.99		
1	**200**	29.98		
3	**300**	29.97		

OpenBook Update Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
PktSize	0	2	Binary Integer	86
DeliveryFlag	2	1	Binary Integer	1
NumberMsgs	3	1	Binary Integer	2
SeqNum	4	4	Binary Integer	1000
SendTime	8	4	Binary Integer	1259832600
SendTimeNS	12	4	Binary Integer	
Msg Size	0	2	Binary Integer	35
Msg Type	2	2	Binary Integer	111
SourceTime	4	4	Binary Integer	1259832600
SourceTimeNS	8	4	Binary Integer	0
SymbolIndex	12	4	Binary Integer	24005
Ultra LastSeqNum	16	4	Binary Integer	40000
TradingStatus	20	1	ASCII Character	O
RemainingCount	21	2	Binary Integer	0
UpdateCount	23	1	Binary Integer	1
Price	24	4	Binary Integer	4999
Volume	28	4	Binary Integer	600
Side	32	1	ASCII Character	B
NumOrders	33	2	Binary Integer	2
Msg Size	35	2	Binary Integer	35
Msg Type	37	2	Binary Integer	111
SourceTime	39	4	Binary Integer	1259832600
SourceTimeNS	43	4	Binary Integer	0
SymbolIndex	47	4	Binary Integer	18006
Ultra LastSeqNum	51	4	Binary Integer	28569
TradingStatus	55	1	ASCII Character	O
RemainingCount	56	2	Binary Integer	0
UpdateCount	58	1	Binary Integer	1
Price	59	4	Binary Integer	3000

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
Volume	63	4	Binary Integer	1200
Side	67	1	ASCII Character	S
NumOrders	68	2	Binary Integer	5

2.1.6.4 Scenario 4 –multiple price point updates for the different symbols

New events:

100 share Buy Order entered at $49.99 for stock ABC.

200 share Buy Order entered at $49.98 for stock ABC.

400 share Sell Order entered at $30.00 for stock XYZ.

100 share Sell Order entered at $30.02 for stock XYZ.

ABC - Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	500	49.99		
1	300	49.98		
3	600	49.97		

ABC - New state of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		50.02	400	4
		50.01	200	1
		50.00	300	2
2	600	49.99		
2	500	49.98		
3	600	49.97		

XYZ - Existing Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		30.02	900	3
		30.01	600	2
		30.00	800	4
1	100	29.99		
1	200	29.98		
3	300	29.97		

XYZ - New state of the Book

#ORDERS	BUY	PRICE POINT	SELL	#ORDERS
		30.02	1000	4
		30.01	600	2
		30.00	1200	5
1	100	29.99		
1	200	29.98		
3	300	29.97		

OpenBook Update Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
PktSize	0	2	Binary Integer	108
DeliveryFlag	2	1	Binary Integer	1
NumberMsgs	3	1	Binary Integer	2
SeqNum	4	4	Binary Integer	1000
SendTime	8	4	Binary Integer	1259832600
SendTimeNS	12	4	Binary Integer	
Msg Size	0	2	Binary Integer	46
Msg Type	2	2	Binary Integer	111
SourceTime	4	4	Binary Integer	1259832600
SourceTimeNS	8	4	Binary Integer	0
SymbolIndex	12	4	Binary Integer	24005
Ultra LastSeqNum	16	4	Binary Integer	40000
TradingStatus	20	1	ASCII Character	O
RemainingCount	21	2	Binary Integer	0
UpdateCount	23	1	Binary Integer	2
Price	24	4	Binary Integer	4999
Volume	28	4	Binary Integer	600
Side	32	1	ASCII Character	B
NumOrders	33	2	Binary Integer	2
Price	35	4	Binary Integer	4998
Volume	39	4	Binary Integer	500
Side	43	1	ASCII Character	B
NumOrders	44	2	Binary Integer	2
Msg Size	46	2	Binary Integer	46
Msg Type	48	2	Binary Integer	111
SourceTime	50	4	Binary Integer	1259832600
SourceTimeNS	54	4	Binary Integer	0
SymbolIndex	58	4	Binary Integer	18006
Ultra LastSeqNum	62	4	Binary Integer	28569

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
TradingStatus	66	1	ASCII Character	O
RemainingCount	67	2	Binary Integer	0
UpdateCount	69	1	Binary Integer	2
Price	70	4	Binary Integer	3000
Volume	74	4	Binary Integer	1200
Side	78	1	ASCII Character	S
NumOrders	79	2	Binary Integer	5
Price	81	4	Binary Integer	3002
Volume	85	4	Binary Integer	1000
Side	89	1	ASCII Character	S
NumOrders	90	2	Binary Integer	4

2.1.6.5 Scenario 5 – Price point delete

New events:

A 500 share execution at $49.99 for stock ABC at 9:30:00am.

Existing Book

#orders	Buy	Price Point	Sell	#orders
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	**500**	49.99		
1	**300**	49.98		
3	**600**	49.97		

New state of the Book

#orders	Buy	Price Point	Sell	#orders
		50.02	400	4
		50.01	200	1
		50.00	300	1
1	**300**	49.98		
3	**600**	49.97		

OpenBook Update Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
PktSize	0	2	Binary Integer	49
DeliveryFlag	2	1	Binary Integer	1
NumberMsgs	3	1	Binary Integer	1
SeqNum	4	4	Binary Integer	1000
SendTime	8	4	Binary Integer	1259832600

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	VALUE
SendTimeNS	12	4	Binary Integer	
Msg Size	0	2	Binary Integer	35
Msg Type	2	2	Binary Integer	111
SourceTime	4	4	Binary Integer	1259832600
SourceTimeNS	8	4	Binary Integer	0
SymbolIndex	12	4	Binary Integer	24005
Ultra LastSeqNum	16	4	Binary Integer	40000
TradingStatus	20	1	ASCII Character	O
RemainingCount	21	2	Binary Integer	0
UpdateCount	23	1	Binary Integer	1
Price	24	4	Binary Integer	4999
Volume	28	4	Binary Integer	0
Side	32	1	ASCII Character	B
NumOrders	33	2	Binary Integer	0

3. CONTROL MESSAGE SPECIFICATIONS

3.1 INTRODUCTION

There are two types of messages transmitted as part of this protocol: control and data. Control messages do not contain data, they allow conversing parties to exchange session-specific information (e.g. 'reset sequence number'). Data messages are product-specific and control messages apply to all products.

3.2 PACKET HEADER FORMAT

All messages will contain a common packet header. See Packet Structure for product specific headers. The design is intended to minimize the development burden on behalf of clients. Meaning that, all clients may implement line-level protocol processing once, and then only need develop parsing algorithms for their choice of message.

3.3 CONTROL MSG TYPES

The table below shows all of the messages types used by the XDP system. To determine which message types are contained in the specific product, please view the Message Type section in the product definition.

Table 15 Control Msg Types

MSGTYPE	DESCRIPTION	NYSE	MKT	ARCA	DELIVERY
1	Sequence Number Reset	x	x		Multicast
2	SourceTime Reference Message				Multicast
3	Symbol Index Mapping	x	x		Multicast
5	Option Series Index Mapping				Multicast
10	Retransmission Request Message	x	x		TCP/IP
11	Request Response Message	x	x		TCP/IP
12	Heartbeat Response Message	x	x		TCP/IP
13	Symbol Index Mapping Request Message	x	x		TCP/IP
15	Refresh Request Message	x	x		TCP/IP
31	Message Unavailable	x	x		Multicast
32	Symbol Clear				Multicast
33	Trading Session Change				Multicast
34	Security Status Message				Multicast
35	Refresh Header Message	x	x		Multicast

3.4 SEQUENCE NUMBER RESET (MSG TYPE ‘1’)

This message is sent to ‘reset’ the Packet Sequence Number at start of day, in response to failures, etc. Note that this message will contain a valid sequence number.

3.4.1 Sequence Number Reset Message Fields

Table 16 Header Fields

FIELD	VALUE
PktSize	’30 Bytes’
DeliveryFlag	Valid values include: ‘12’ – Sequence Number Reset
NumberMsgs	1
SeqNum	1
SendTime	
SendTimeNS	

Table 17 Message Body Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. ’14 Bytes’
MsgType	2	2	Binary Integer	This field identifies the type of message. ‘1’ – Sequence Number Reset message
SourceTime	4	4	Binary Integer	This field specifies the time when the message was generated in the order book. The number represents the number of seconds in **UTC** time (EPOCH)
SourceTimeNS	8	4	Binary Integer	This field specifies the number represents the nanosecond portion of **UTC** time (since EPOC)
ProductID	12	1	Binary Integer	The product ID used in the XDP header to identify the NYSE feed. See Product Ids.
ChannelID	13	1	Binary Integer	This field contains the multicast channel ID over which the packet was sent.

3.4.2 Sequence Number Reset Processing Notes

Sequence numbers normally begin at one (1) with a sequence number reset message and increase by calculating SeqNum+NumberMsgs with each subsequent packet. There are three scenarios where the sequence number is reset:

- A start of day a sequence number reset message is sent
- If the value should exceed the maximum value that the SeqNum field may contain, it will be reset to one (1).
- If XDP has a failure and it recovers, it sends a sequence number reset message. The SeqNum field of that message will be set to one (1)

3.5 HEARTBEAT

Heartbeat messages are sent only over both TCP/IP and multicast connection.

3.5.1 Heartbeat Message Fields

Table 18 Header Fields

FIELD	VALUE
PktSize	’16 Bytes’
DeliveryFlag	Valid values include: ‘1’– Heartbeat Message Only
NumberMsgs	
SeqNum	Next expected sequence number
SendTime	
SendTimeNS	

3.5.2 General Heartbeat Processing Notes (TCP and Multicast)

The following applies to the TCP channels for retransmissions and refresh, and also the multicast channels for real time and refresh data.

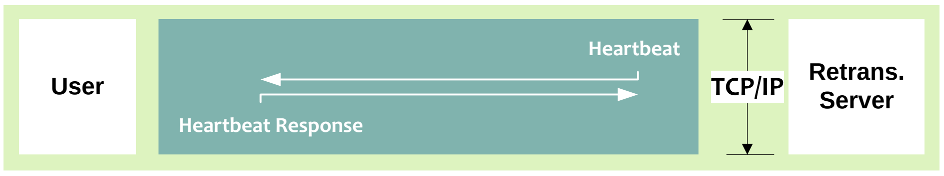
- Heartbeat messages will only contain the packet header (with a DeliveryFlag = ‘1’).
- Heartbeats will sent over on the TCP/IP Retrans Request connection and over the production multicast
- Heartbeat frequency since the last packet, is:
  - 60 seconds in the active TCP/IP retransmission sessions
  - 60 seconds on the multicast lines when there is no data content being sent
  - Heartbeat messages will be sent with the next expected sequence number. Please see Sequence Numbers.

3.5.3 Retransmission and Refresh Heartbeat Processing Notes (TCP)

- Clients may receive a heartbeat message if they have an active TCP/IP session with the retransmission or refresh server.
- To determine the health of the user connection on the TCP/IP channel, the retransmission or refresh server will send regular heartbeat messages to the user. The heartbeat frequency is 60 seconds. The time out for this heartbeat response message is set at 5 seconds. If no response is received by the server within this timeframe, the TCP/IP session will be disconnected.

- Clients that choose to establish and remain connected to the retransmission or refresh server intraday must respond to a heartbeat message with a heartbeat response message. Users can choose to either disconnect following each retransmission or refresh request, or remain connected to the retransmission or refresh server.

Figure 9 Retransmission/Refresh Server Heartbeats



3.6 HEARTBEAT RESPONSE (MSG TYPE ‘12’)

Clients that choose to establish and remain connected to the retransmission server intraday must respond to a heartbeat message with a heartbeat response message.

3.6.1 Heartbeat Response Message Field

Table 19 Header Fields

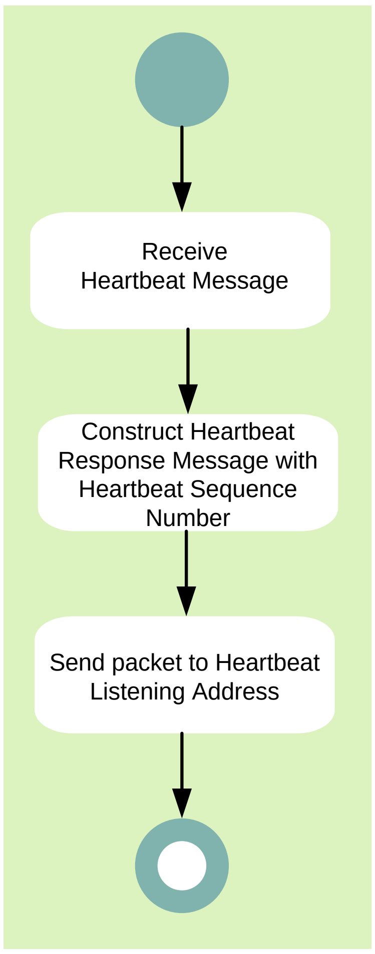
FIELD	VALUE
PktSize	’30 Bytes’
DeliveryFlag	Valid values include: ‘11’ – Original Message Only
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Table 20 Message Body Fields

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. ’14 Bytes’
MsgType	2	2	Binary Integer	This field identifies the type of message. ‘12’ – Heartbeat Response message
SourceID	4	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 9 characters, null terminated.

3.6.2 Processing

Figure 10 Processing


Receive Heartbeat Message
Construct Heartbeat Response Message with Heartbeat Sequence Number
Send packet to Heartbeat Listening Address

3.7 RETRANSMISSION REQUEST (MSG TYPE ‘10’)

This message is sent by clients requesting missing messages identified by a sequence number gap. Upon receipt of a valid retransmission request message, the requested message(s) will be sent. The requested message(s) have the same message format and content as the original sent by the system.

3.7.1 Retransmission Request Message Fields

Table 21 Header Fields

FIELD	VALUE
PktSize	’40 Bytes’
DeliveryFlag	Valid values include: ‘11’ – Original Message uncompressed ‘31’ – Original Message compressed (FAST)
NumberMsgs	
SeqNum	

FIELD	VALUE
SendTime	
SendTimeNS	

Table 22 Message Body Fields

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '24 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '10' – Retransmission Request message
BeginSeqNum	4	4	Binary Integer	The beginning sequence number of the requested range of messages to be retransmitted.
EndSeqNum	8	4	Binary Integer	The end sequence number of the requested range of messages to be retransmitted.
SourceID	12	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 9 characters, null terminated.
ProductID	22	1	Binary Integer	The product ID used to identify the NYSE feed. See Product Ids.
ChannelID	23	1	Binary Integer	This field contains the multicast channel ID where you would like to request data from

3.8 REFRESH REQUEST (MSG TYPE '15')

This message is sent by clients requesting a refresh. The system will provide the appropriate message(s) in response.

3.8.1 Refresh Request Message Field

Table 23 Header Fields

FIELD	VALUE
PktSize	'36 Bytes'
DeliveryFlag	Valid values include: '11' – Original Message uncompressed '31' – Original Message compressed (FAST)
NumberMsgs	1
SeqNum	

FIELD	VALUE
SendTime	
SendTimeNS	

Table 24 Message Body Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. ’20 Bytes’
MsgType	2	2	Binary Integer	This field identifies the type of message. ‘15’ – Refresh Request Message
SymbolIndex	4	4	Binary Integer	This field identifies the numerical representation of the symbol. Note: If this field is set to zero, then XDP will generate a refresh for all symbols
SourceID	8	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 9 characters, null terminated.
ProductID	18	1	Binary Integer	The product ID used to identify the NYSE feed. See Product Ids.
ChannelID	19	1	Binary Integer	This field contains the multicast channel ID where you would like to request data from

3.9 REQUEST RESPONSE MESSAGE (MSG TYPE ‘11’)

This message will be sent immediately via TCP/IP in response to the client’s request for retransmission/refresh messages.

3.9.1 Request Response Message Fields

Table 25 Header Fields

FIELD	VALUE
PktSize	
DeliveryFlag	Valid values include: ‘11’ – Original Message uncompressed ‘31’ – Original Message compressed (FAST)
NumberMsgs	1

FIELD	VALUE
SeqNum	
SendTime	
SendTimeNS	

Table 26 Message Body Fields

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '21' Bytes
MsgType	2	2	Binary Integer	This field identifies the type of message. '11' – Request Response Message
RequestSeqNum	4	4	Binary Integer	This field contains the request message sequence number assigned by the client. It is used by the client to couple the request with the response message.
SourceID	8	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 10 characters, null terminated.
ProductID	18	1	Binary Integer	The product ID used in the XDP header to identify the NYSE feed. See Product Ids.
ChannelID	19	1	Binary Integer	This field contains the multicast channel ID where you requested data from
Status	20	1	ASCII Character	This is a flag that indicates the reason why the request was rejected. Valid values: '0' – Message was accepted '1' – Rejected due to an Invalid source ID '2' – Rejected due to invalid sequence range '3' – Rejected due to maximum sequence range (see threshold limits) '4' – Rejected due to maximum request in a day '5' – Rejected due to maximum number of refresh requests in a day '6' – Rejected. Request message seqnum TTL (Time to live) is too old. Use refresh to recover

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				current state if necessary. '7' – Rejected due to an Invalid Channel ID '8' – Rejected due to an Invalid Product ID

3.10 RETRANSMISSION MESSAGES

Upon receipt of a valid retransmission request message, the requested message(s) will be sent. This message(s) has the same message format and content as the original messages sent by XDP but may be grouped in a single packet for maximum packet efficiency (see Sequence Numbers). These messages will flow through the retransmission IP address and channel.

3.10.1 Retransmission Message Packet Header

Table 27 Retransmission Message Fields

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	13' – Only one packet in retransmission update Uncompressed '14' – Start of retrans Update Uncompressed '15' – Part of a retransmission sequence Uncompressed '16' – End of retrans Update Uncompressed '17' – Only one packet in Refresh update Uncompressed '18 – Start of Refresh Update Uncompressed '19' – Part of a Refresh sequence Uncompressed '20' – End of Refresh Update Uncompressed '33' - Only one packet in retransmission update compressed (Fast) '34' – Start of retrans Update compressed (Fast) '35' – Part of a retransmission sequence compressed (Fast) '36' – End of retrans Update compressed (Fast) '37' – Only one packet in Refresh update compressed (Fast) '38' – Start of Refresh Update compressed (Fast) '39' – Part of a Refresh sequence compressed (Fast) '40' – End of Refresh Update compressed (Fast)
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

3.10.2 Processing

All Subscribers will receive retransmission messages through the retransmission channel.

Due to the multicast nature, subscribers will receive ‘all’ retransmission messages, including messages that were not requested by them.

Note that when a message for a particular symbol is retransmitted, a new message **for the same symbol** may be sent through the regular channel. This scenario is very likely to occur with busy symbols and may cause confusion as to which message contains the latest information on that symbol.

In order to resolve the conflict, the following qualification method should be applied:

- Check the SeqNum field. A retransmitted message retains the same sequence number as the original message
- The most current sequence number (SEQNUM) contains the latest information.
- If the SEQNUMS are the same: messages are the same, any of the two messages contains the same information.

3.11 SYMBOL INDEX MAPPING REQUEST MESSAGES (MSG TYPE ‘13’)

This message is sent by Subscribers via TCP/IP requesting the Symbol index mapping.

Table 28 Symbol Index Mapping Request Message Packet Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ‘11’ – Original Message uncompressed ‘31’ – Original Message compressed (FAST)
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Table 29 Symbol Index Mapping Request Msg Body

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. ‘21 Bytes’
MsgType	2	2	Binary Integer	This field identifies the type of message. ‘13’ – Symbol Index Mapping Request Message
SymbolIndex	4	4	Binary	This field identifies the numerical representation of the symbol. SymbolIndex

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	value can be zero, which is to request all symbol mapping for the multicast group.*
SourceID	8	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 9 characters, null terminated.
ProductID	18	1	Binary Integer	The product ID used in the XDP header to identify the NYSE feed. See Product Ids.
ChannelID	19	1	Binary Integer	This field contains the multicast channel ID where you requesting the index map from.
RetransmitMethod**	20	1	Binary Integer	This field identifies the Retransmission method for the symbol index mapping. Valid Values: '0' – retransmit via UDP (this is default in NYSE/MKT)** '1' – retransmit via TCP/IP connection (This is default on Arca)**

* To request all symbols for a specific multicast group, specify '0' (zero).

** RetransmitMethod is a field that gives customers the ability to specify if they want Symbol Index Mapping sent to them via TCP/IP or UDP. This feature is not yet available.

3.12 SYMBOL INDEX MAPPING REQUEST MESSAGES (MSG TYPE '3')

This message is sent to Subscribers via TCP/IP or Multicast (**Please note that for NYSE/MKT this message is available ONLY over Multicast at this time)** upon request of the Symbol index mapping.

Table 30 Symbol Index Mapping Message Packet Header

FIELD	DESCRIPTION
PktSize	This field indicates the size of the message body in bytes.
DeliveryFlag	Valid values include: '11' – Original Message Uncompressed '31' – Original Message Compressed '17' – Only one packet in Refresh update Uncompressed '18 – Start of Refresh Update Uncompressed '19' – Part of a Refresh sequence Uncompressed '20' – End of Refresh Update Uncompressed '37' – Only one packet in Refresh update compressed (Fast)

FIELD	DESCRIPTION
	‘38’ – Start of Refresh Update compressed (Fast) ‘39’ – Part of a Refresh sequence compressed (Fast) ‘40’ – End of Refresh Update compressed (Fast)
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Table 31 Symbol Index Mapping Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message
Msg Type	2	2	Binary Integer	This field identifies the type of message ‘3’ – Symbol Index Map Message
SymbolIndex	4	4	Binary Integer	This field identifies the numerical representation of the symbol.
Symbol	8	11	ASCII String	This field contains the full symbol in NYSE Symbology. A sequence of characters representing the symbol, padded with NULLs. For more information, see *NYSE Symbology*
Filler	19	1	Binary Integer	This field is reserved for future use
Market ID	20	2	Binary Integer	ID of the Originating Market. 1 - NYSE Cash 2 - Europe Cash 3 - Arca Cash 4 - NYSE/Arca Options 5 - NYSE/Arca Bonds 6 - ArcaEdge 7 - LIFFE 8 - AMEX Options 9 - MKT Cash
System ID	22	1	Binary Integer	ID of the Originating System. This field is not yet applicable to the NYSE & NYSE MKT equities market
Exchange Code	23	1	ASCII Character	Exchanges where it is Listed. N – NYSE

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				P – ARCA Q – NASDAQ A - MKT
PriceScaleCode	24	1	Binary Integer	Price scale for price conversion of the symbol. See Price Formats.
Security Type	25	1	ASCII String	Type of Security A – ADR C - COMMON STOCK E – ETF F – FOREIGN I – UNITS M - MISC/LIQUID TRUST P - PREFERRED STOCK R – RIGHTS S - SHARES OF BENEFICIARY INTEREST T – TEST U – UNITS W – WARRANT
UOT	26	2	Binary Integer	Lot Size requirement
PrevClosePrice	28	4	Binary Integer	This field contains the previous day's closing price for the security
PrevCloseVolume	32	4	Binary Integer	This field contains the previous day's closing volume for the security
Price Resolution	36	1	Binary Integer	0 = All Penny 1 = Penny/Nickel 5 = Nickel/Dime
Round Lot	37	1	ASCII String	Round Lot Accepted Y – Yes N – No
BloombergCompany Id (ID_BB_GLOBAL)	38	12	Binary Integer	Unique global identifier assigned by Bloomberg for all securities across every asset class. '0' - information is currently unavailable **Default Value:** 0

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
BloombergSecurityId (ID_BB_SEC_NUM_SRC)	50	20	Binary Integer	Unique Bloomberg identifier (BSID) including the Bloomberg source for all Bloomberg securities as an integer value. This field is used for B Pipe and subscription services.
Bloomberg Symbol (ID_BB_SEC_NUM_DES)	62	30	ASCII String	Unique Bloomberg name for all Bloomberg securities.

Note: a Binary zero in any field (including ASCII) unless specified represents that the information is not currently available.

3.13 MESSAGE UNAVAILABLE MESSAGE (MSG TYPE '31')

This message will be sent to inform the subscribers of unavailability of a range of messages for which they may have requested retransmission via the Retransmission Multicast channels. Below is the message format.

Table 32 Message Unavailable Message Packet Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: '21' – Message Unavailable '41' – Message Unavailable
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Table 33 Message Unavailable Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '14 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '31' – Message Unavailable
BeginSeqNum	4	4	Binary Integer	The beginning sequence number of the requested range of messages to be retransmitted.
EndSeqNum	8	4	Binary Integer	The end sequence number of the requested range of messages to be retransmitted.
ProductID	12	1	Binary	The product ID used in the XDP header to identify

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
			Integer	the NYSE feed. See Product Ids.
ChannelID	13	1	Binary Integer	This field contains the multicast channel ID where you requesting the index map from.

3.14 REFRESH HEADER (MSG TYPE '35')

This message is the first message type sent each refresh packet and will never been sent out as its own packet.

3.14.1 Refresh Header Message Field

Table 34 Header Fields

FIELD	VALUE
PktSize	
DeliveryFlag	Valid values include: '17' – Only one packet in Refresh update Uncompressed '18 – Start of Refresh Update Uncompressed '19' – Part of a Refresh sequence Uncompressed '20' – End of Refresh Update Uncompressed '37' – Only one packet in Refresh update compressed (Fast) '38' – Start of Refresh Update compressed (Fast) '39' – Part of a Refresh sequence compressed (Fast) '40' – End of Refresh Update compressed (Fast)
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Table 35 Message Body Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '12 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '35' – Refresh Header Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
CurrentRefreshPkt	4	2	Binary Integer	This field represents the current refresh packet in the update
TotalRefreshPkts	6	2	Binary Integer	This field represents the total number of refresh packets you should expect in the update
LastSeqNum	8	4	Binary Integer	This field contains the last sequence number sent on the feed. The refresh is the state of the book as of this sequence number.

3.14.2 Refresh Example

Assuming the refresh of symbol xyz requires 3 packets.

The first Packet structure will look as follows:

PACKET HEADER	REFRESH HEADER	MESSAGE 1	MESSAGE 2	...	MESSAGE N

The Packet header will look as follows:

FIELD	VALUE
PktSize	
DeliveryFlag	Valid values include: '17' – Only one packet in Refresh update Uncompressed '18 – Start of Refresh Update Uncompressed '19' – Part of a Refresh sequence Uncompressed '20' – End of Refresh Update Uncompressed '37' – Only one packet in Refresh update compressed (Fast) '38' – Start of Refresh Update compressed (Fast) '39' – Part of a Refresh sequence compressed (Fast) '40' – End of Refresh Update compressed (Fast)
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

The first message in the body will be the refresh header.

FIELD NAME	OFFSET	SIZE (BYTES)	VALUE
MsgSize	0	2	12 Bytes
MsgType	2	2	35' – Refresh Header Message
CurrentRefreshPkt	4	2	1
TotalRefreshPkts	6	2	3
LastSeqNum	8	4	5000

Followed by message type 110 – Snapshot message.

4. PRODUCTION CONFIGURATION

4.1 INTRODUCTION

4.1.1 Data Content

The information supplied in this chapter applies to the production environment only.

4.1.2 Data Delivery

NYSE market data is only available via the SFTI network.

4.2 XDP PRODUCTION HOURS

The following table is the overview of the main daily event-generating activity and indicative time on the OpenBook Aggregated Feed.

Table 36 Production Hours

EVENT	TIME (GMT)	COMMENT
Sequence Number Reset	~1:00am	
Symbol Mapping	~1:00am	
Snapshot Message	~4:00am	
Pre-Open	7:30am	Open time and close time are for the first financial products to open or close
Open	9:30am	
Early Closing Time	1:00pm	
Close	4:00pm	

4.3 MULTICAST/TCP SETUP

4.3.1 Joining Multicast Groups

Recipient's applications/hosts will be responsible for issuing Multicast subscriptions to one or more of the Multicast Groups assigned to the product. In response to each authorized subscription request, SFTI network will complete the tasks associated with delivering the Multicast packets from the data source to the recipient's network.

The process of subscribing to a Multicast Group ID is also known as 'joining' a Multicast Group. Upon session termination, the subscriber's host system should issue an 'unjoin' message. This will terminate delivery of data to that host's local network. If an application/host terminates without issuing an 'unjoin' message, the network will eventually issue a 'timeout' for the Multicast Group subscription that will automatically terminate delivery of the Multicast packets to the host's local network.

4.3.2 Feeds

All data is published using two sets of unique IP Multicast Group IDs ("Primary" and "Secondary")—the two will originate from the same distribution site. The feeds are redundant to each other, i.e., they are synchronized with each other. Each message from each feed contains the same packet sequence number.

4.3.3 Data Feed IP Addresses

The following link contains all of the IP addresses, TCP Source IP addresses and channel assignments for the feed: http://www.nyxdata.com/ipaddresses

4.3.4 Heartbeat Mechanism

The heartbeat message frequency is set to 60 seconds on the TCP/IP connection.

A heartbeat message response has to be sent within 5 seconds to stay connected to the server.

4.3.5 Source ID

The Source ID allows clients to perform retransmission and refresh requests. Please note that the Source IDs for retransmissions and refresh are identical. NYSE will provide each client with a default of 1 Source ID for Production with a limit of up to 4 Source IDs per client.

Each Source ID may only be logged on to a server once at a given time.

4.3.6 Number of Source IDs

Clients are allowed a max of 4 Source IDs. Please contact NYSE Technical support services to setup a Source ID.

4.3.7 Parallel Sessions

Clients may file several concurrent requests on the server at the same time with the same Source ID; there is no need to wait for the active retransmission to be closed to ask for another one.

Responses to these requests are sent in the same order as the initial requests.

4.3.8 Retransmission/Refresh Request Limitations

The recommendations below apply to production and test.

Table 37 Limitations

CAPABILITY	DESCRIPTION	THRESHOLD	ACTION
Prevention of invalid subscribers	Incoming requests from subscribers that are not in the enabled subscriber's source ID list will not be honored. XDP subscribers will need a source ID, which is a string that uniquely identifies the subscriber of the retransmission requests. Please contact the NYSE Service Desk to get a unique source ID.	N/A	Request will not be processed.
Limitation of Requests for a large number of packets	Only retransmission requests for 1000 packets or less will be honored.	1000	Request will not be processed.
Limitation of Generic Requests Time Interval	If the generic request on a message which is not within this threshold, the request will not be honored.	75000	Request will not be honored.
Limitation of Generic Requests	Generic requests for messages not within the threshold number of requests per day, will	500	Subsequent retransmissions requests from

CAPABILITY	DESCRIPTION	THRESHOLD	ACTION
	not be honored during that particular day.		that subscriber will be blocked.
Limitation of requests for refresh messages	Up to 5000 refresh requests will be honored.	5000	Request will not be honored.
Limitation of Index Mapping Requests	Up to 500 Symbol Index Mapping requests will be honored.	N/A	Request will not be honored.

5. UAT DATA FEED CONFIGURATION

5.1 INTRODUCTION

5.1.1 Data Content

The information supplied in this chapter applies to the UAT environment only.

5.1.2 Data Delivery

NYSE market data is only available via the SFTI network.

5.2 XDP UAT HOURS

The following table is the overview of the main daily event-generating activity and indicative time on the OpenBook Aggregated Feed.

Table 38 UAT Hours

EVENT	TIME (GMT)	COMMENT
Sequence Number Reset	~8:00am	
Symbol Mapping	~8:00am	
Pre-Open	8:30am	The data is replayed throughout the trading day.
Open	9:30am	
Close	5:00pm	

5.3 MULTICAST/TCP SETUP

5.3.1 Joining Multicast Groups

Recipient's applications/hosts will be responsible for issuing Multicast subscriptions to one or more of the Multicast Groups assigned to the product. In response to each authorized subscription request, SFTI network will complete the tasks associated with delivering the Multicast packets from the data source to the recipient's network.

The process of subscribing to a Multicast Group ID is also known as 'joining' a Multicast Group. Upon session termination, the subscriber's host system should issue an 'unjoin' message. This will terminate delivery of data to that host's local network. If an application/host terminates without issuing an 'unjoin' message, the network will eventually issue a 'timeout' for the Multicast Group subscription that will automatically terminate delivery of the Multicast packets to the host's local network.

5.3.2 Feeds

All data is published using two sets of unique IP Multicast Group IDs ("Primary" and "Secondary")—the two will originate from the same distribution site. The feeds are redundant to each other, i.e., they are synchronized with each other. Each message from each feed contains the same packet sequence number.

5.3.3 Data Feed IP Addresses

The following link contains all of the Production and CERT IP addresses, TCP Source IP addresses and channel assignments for the feed.

http://www.nyxdata.com/ipaddresses

5.3.4 Heartbeat Mechanism

The heartbeat message frequency is set to 60 seconds on the TCP/IP connection.

A heartbeat message response has to be sent within 5 seconds to stay connected to the server.

5.3.5 UAT Source ID

The Source ID allows clients to perform retransmission and refresh requests. Please note that the Source IDs for retransmissions and refresh are identical. NYSE will provide each client with a default of 1 Source ID for Production with a limit of up to 4 Source IDs per client.

Each Source ID may only be logged on to a server once at a given time.

5.3.6 Number of Source IDs

The Source ID in the UAT environment is identical to the production environment. Clients are allowed a max of 4 Source IDs. Please contact NYSE Technical support services to setup a Source ID.

5.3.7 Parallel Sessions

Clients may file several concurrent requests on the server at the same time with the same Source ID; there is no need to wait for the active retransmission to be closed to ask for another one.

Responses to these requests are sent in the same order as the initial requests.

5.3.8 UAT Retransmission/Refresh Request Limitations

The recommendations below apply to production and test.

Table 39 UAT Limitations

CAPABILITY	DESCRIPTION	THRESHOLD	ACTION
Prevention of invalid subscribers	Incoming requests from subscribers that are not in the enabled subscriber's source ID list will not be honored. XDP subscribers will need a source ID, which is a string that uniquely identifies the subscriber of the retransmission requests. Please contact the NYSE Service Desk to get a unique source ID.	N/A	Request will not be processed.
Limitation of Requests for a large number of sequence numbers	Only retransmission requests for 1000 packets or less will be honored.	1000	Request will not be processed.
Limitation of Generic Requests Time Interval	If the generic request on a message which is not within this threshold, the request will not be honored.	75000	Request will not be honored.
Limitation of Generic Requests	Generic requests for messages not within the threshold number of requests per day will not be honored during that particular	500	Subsequent retransmissions requests from that subscriber will be

CAPABILITY	DESCRIPTION	THRESHOLD	ACTION
	day.		blocked.
Limitation of requests for refresh messages	Up to 5000 refresh requests will be honored.	5000	Request will not be honored.
Limitation of Index Mapping Requests	Up to 500 Symbol Index Mapping requests will be honored.	N/A	Request will not be honored.

6. TEST DATA FEED CONFIGURATION

6.1 INTRODUCTION

6.1.1 Data Content

The information supplied in this chapter applies to the XDP Test environment only.

6.1.2 Data Delivery

NYSE market data is only available via the SFTI network.

6.2 XDP TEST HOURS

The following table is the overview of the main daily event generating activity and indicative time on the OpenBook Aggregated Feed. Normal testing hours are **Tuesday and Thursday nights from 7pm to 9pm. Please note that the Production test IP addresses can also be used during release rollouts and if so, will follow the same hours as Production.**

6.3 MULTICAST/TCP SETUP

6.3.1 Joining Multicast Groups

Recipient's applications/hosts will be responsible for issuing Multicast subscriptions to one or more of the Multicast Groups assigned to the product. In response to each authorized subscription request, SFTI network will complete the tasks associated with delivering the Multicast packets from the data source to the recipient's network.

The process of subscribing to a Multicast Group ID is also known as 'joining' a Multicast Group. Upon session termination, the subscriber's host system should issue an 'unjoin' message. This will terminate delivery of data to that host's local network. If an application/host terminates without issuing an 'unjoin' message, the network will eventually issue a 'timeout' for the Multicast Group subscription that will automatically terminate delivery of the Multicast packets to the host's local network.

6.3.2 Feeds

All data is published using two sets of unique IP Multicast Group IDs ("Primary" and "Secondary")—the two will originate from the same distribution site. The feeds are redundant to each other, i.e., they are synchronized with each other. Each message from each feed contains the same packet sequence number.

6.3.3 Data Feed IP Addresses

The following link contains all of the Production and CERT IP addresses, TCP Source IP addresses and channel assignments for the feed: http://www.nyxdata.com/ipaddresses

6.3.4 Heartbeat Mechanism

The heartbeat message frequency is set to 60 seconds on the TCP/IP connection.

A heartbeat message response has to be sent within 5 seconds to stay connected to the server.

6.3.5 Test Source ID

The Source ID allows clients to perform retransmission and refresh requests. Please note that the Source IDs for retransmissions and refresh are identical. NYSE will provide each client with a default of 1 Source ID for Production with a limit of up to 4 Source IDs per client. Each Source ID may only be logged on to a server once at a given time.

6.3.6 Number of Source IDs

The Source ID in the Test environment is identical to the production environment. Clients are allowed a max of 4 Source IDs. Please contact NYSE Technical support services to setup a Source ID.

6.3.7 Parallel Sessions

Clients may file several concurrent requests on the server at the same time with the same Source ID; there is no need to wait for the active retransmission to be closed to ask for another one.

Responses to these requests are sent in the same order as the initial requests.

6.3.8 Test Retransmission/Refresh Request Limitations

The recommendations below apply to production and test.

Table 40 Test Limitations

CAPABILITY	DESCRIPTION	THRESHOLD	ACTION
Prevention of invalid subscribers	Incoming requests from subscribers that are not in the enabled subscriber's Source ID list will not be honored. XDP subscribers will need a Source ID, which is a string that uniquely identifies the subscriber of the retransmission requests. Please contact the NYSE Service Desk to get a unique Source ID.	N/A	Request will not be processed.
Limitation of Requests for a large number of sequence numbers	Only retransmission requests for 1000 packets or less will be honored.	1000	Request will not be processed.
Limitation of Generic Requests Time Interval	If the generic request on a message which is not within this threshold, the request will not be honored.	75000	Request will not be honored.
Limitation of Generic Requests	Generic requests for messages not within the threshold number of requests per day, will not be honored during that particular day.	500	Subsequent retransmissions requests from that subscriber will be blocked.
Limitation of requests for refresh messages	Up to 5000 refresh requests will be honored.	5000	Request will not be honored.
Limitation of Index Mapping Requests	Up to 500 Symbol Index Mapping requests will be honored.	N/A	Request will not be honored.