Document title

XDP COMMON CLIENT SPECIFICATION

Version	Date
1.6a	3 Jun 2014

PREFACE

DOCUMENT HISTORY

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

VERSION NO.	DATE	CHANGE DESCRIPTION
0.1	09/01/11	– Split original spec into two and created a new “XDP Common Client Spec” – Initial distribution
0.2	09/29/11	Formatting changes throughout
0.3	09/30/11	Updated the descriptions of the BeginSeqNum and EndSeqNum fields in Table 18 (Request Response Message Fields)
0.4	11/07/11	Updated: – Table 13 (Time Reference Message Fields) – Table 16 (Retransmission Request Message Fields) – Table 17 (Refresh Request Message Field) – Table 18 (Request Response Message Fields)
0.5	02/02/12	Updated: – Section 1.3.3 (Packet Structure; ‘DeliveryFlag’ field) – Section 1.3.4 (XDP Failover) – Section 1.3.15 (Symbol Index Mapping FTP) – Section 2.13 (Symbol Index Mapping Message; description of ‘Refreshes of a Single Symbol’) – Artwork throughout
0.6	02/16/12	Updated: – Description of the ‘TickerDesignation’ field in Table 10 (Symbol Index Mapping FTP File Format) – FTP file paths in Section 1.3.15 (Symbol Index Mapping FTP)
0.7	03/09/12	Added values of ‘D’, ‘H’, ‘L’ and ‘O’ to the description of the Security Type field in Table 21 (Symbol Index Mapping Message Fields – Msg Body)
0.8	05/16/2012	Changed name to NYSE MKT throughout.
0.9	07/23/2012	Added value ‘T’ to description of the Listed Market field in Table 10 (Symbol Index Mapping).
1.0	08/08/2012	Rebranded document with new NYSE Technologies template
1.1	20/06/2013	Update TickerDesignation Q for tape C.
1.2	08/07/2013	Added values to exchange code description in Table 21 (Msg Body)
1.3	09/24/2013	Removal of section 1.3.2 Fast Optimization. Added SSR fields to the

VERSION NO.	DATE	CHANGE DESCRIPTION
		Security Status Message – MSG Type ‘34’.
1.4	10/14/2013	Added ‘Z’ as exchange code for BATS in section 2.13 Table 21
1.5	11/6/2013	Section 1.3.14 Symbol Index Mapping FTP – Update to Listed Markets, Section 2.13 Message Body – Update to Security Types Section 2.18 MSG Body – Updates to various fields
1.6	4/9/2014	Updated Security Status and Symbol Index messages to include notes reflecting current implementation Updated Security Status fields to have standalone SSRState, MarketState, and SessionState fields
1.6a	6/3/2014	Updated section 2.13 Table 21 to include ‘Z’ exchange code erroneously removed in previous version Updated minor typos not affecting data content

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

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 information about market data products, visit http://www.nyxdata.com/Data-Products
- 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 INTRODUCTION

1.1.1 Market Data Overview

The data feed has the following high-level features:

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

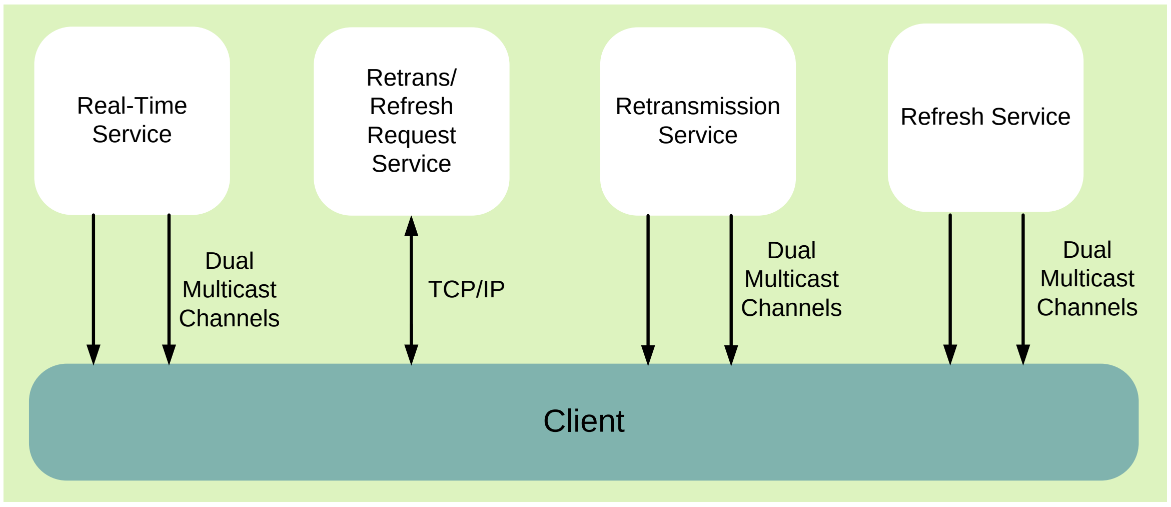
This chapter provides detailed information about the features of the feed, to support the development of client applications by Traders, Independent Software Vendors and Quote Vendors.

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 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

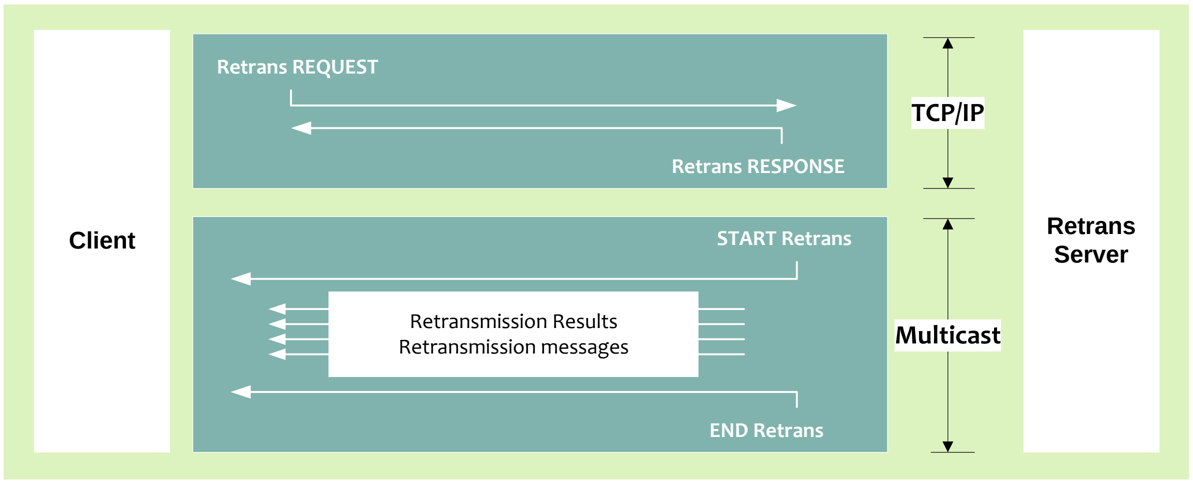
The retransmission functionality is designed to allow the user to recapture a small number of missed messages.

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 should join the retransmission multicast channels first before making a TCP/IP connection and requesting messages for retransmission.

Figure 2 Retransmission Request shows the sequence of messages and the transport protocols employed when making a retransmission request.

Figure 2 Retransmission Request



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 Product ID
- Invalid Channel ID
- 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 advice 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 Euronext.

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 should join the Refresh multicast channels first before making a TCP/IP connection and requesting messages for refresh.

The Retransmission/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 as follows:

- **Refreshes of a Single Symbol** The Delivery Flags of all refreshes of a single symbol that span multiple packets contain 'Part' indications only. You use the PktNum and NumPkts fields to know when the complete refresh has been received.
- **Refreshes of All Symbols** Start, Part and End Delivery Flags are used for refreshes of all symbols. All packets of the first symbol are marked Start and all packets of the last symbol are marked End. This applies also to refreshes of all Symbol Index Mapping messages.

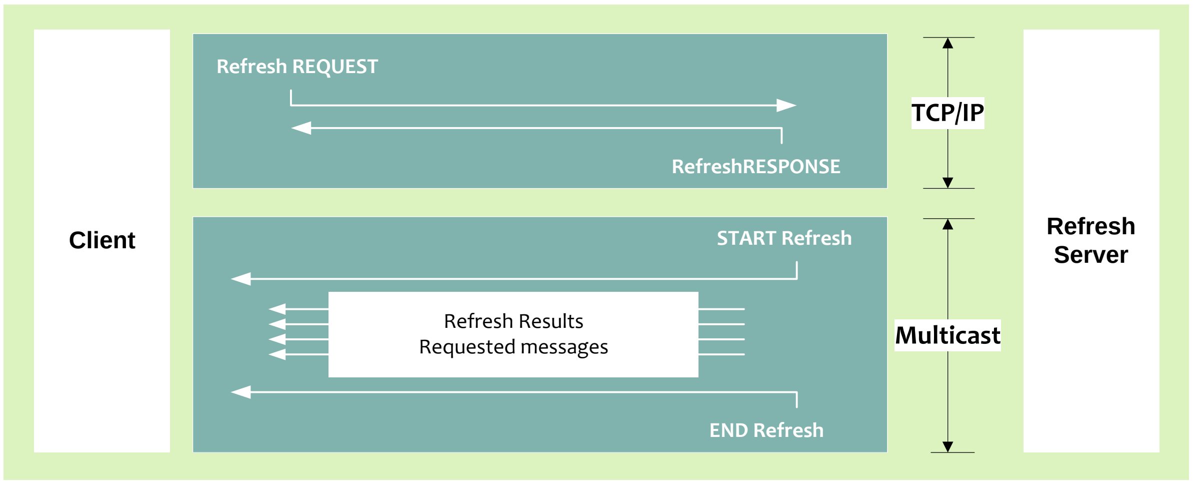
Each refresh packet begins with a Refresh Header (Msg Type '35') that identifies what the current packet is in the sequence of the refresh updates, and what the total number of refresh packets is. No dedicated retransmission service is available for the refresh; if message loss is detected, clients should 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.

Figure 3 Refresh Request shows the sequence of messages and the transport protocols employed when making a refresh request.

Figure 3 Refresh Request



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

- Invalid Source ID
- Invalid Product ID
- Invalid Channel 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 Euronext.

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 (for example, 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
- All messages are on a 1 byte boundary
- 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 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, and so forth.

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 and 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 ■ 10 – XDP Failover (see XDP Failover) ■ 11 – Original Message ■ 12 – Sequence Number Reset Message ■ 13 – Only one packet in retransmission update Uncompressed ■ 15 – Part of a retransmission sequence Uncompressed ■ 17 – Only one packet in Refresh update Uncompressed ■ 18 – Start of Refresh Update Uncompressed

FIELD	OFFSETS	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ 19 – Part of a Refresh sequence Uncompressed ■ 20 – End of Refresh Update Uncompressed ■ 21 – Message Unavailable ■ 30 – Compacted XDP Failover ■ 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.
SeqNum	4	4	Binary Integer	This field contains the message sequence number assigned by XDP for each channel. It is used for gap detection, unique for each broadcast stream (except if reset during the day), see Sequence Numbers.
SendTime	8	4	Binary Integer	This field specifies the time when the packet was sent to the multicast channel for publication. 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 two fields: a two-byte message length, followed by a two-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.

1.3.3 XDP Failover

When failing over, the DeliveryFlag field is set to either 10 or 30 at the start of failover. The following then occurs:

1. The sequence reset is sent first and by itself
2. Each symbol is looped and sent
3. Symbol Mapping occurs
4. Symbols are cleared, with the next expected sequence number for the symbol
5. The Last Security status message is sent
6. The Last Trading Session Change message is sent
7. All open refresh messages are sent
8. Last Source time reference message is sent

Once all symbols have been processed the DeliveryFlag field is set back to 11 or 31.

1.3.4 Msg Size Field Processing

Customers should not hard code msg sizes in feed handlers; instead the feed handler should use the Msg Size 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, **Table 3 Msg Size Field Processing** 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 Msg 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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
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 Price scale from the symbol mapping index.
Volume	20	4	Binary Integer	This field contains the size of the order.

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

FIELD	OFFSET	SIZE	FORMAT	DESCRIPTION
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.

For example, let’s say the original format had the following 16-byte message:

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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
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 four-byte volume field adjusting the Msg 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 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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
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.

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

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.

XDP uses the concept of a time reference that identifies the whole number of seconds in UTC time, and each data message contains the nanosecond offset from that time reference value.

Each symbol will send a Time Reference Message (Msg Type '2') every x number of seconds containing the symbol index to identify the source timestamp.

The XDP System itself will send only a packet header Send Time that is in UTC time that does not require a time reference message.

1.3.6 Sequence Numbers

All messages conform to the message level sequencing.. Each channel A, B, C, D, and so forth 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:

- Missing (gapped) packets
- Unordered packets
- Duplicate packets

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

Table 6 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 5 packets they would request a gap fill for messages 1-11. Instead of sending the original five packets, one packet is sent with eleven messages (assuming the total number of messages fits within the 1500-byte packet size, otherwise the delivery flag will indicate that the update will span multiple packets).

1.3.7 Line Arbitration

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

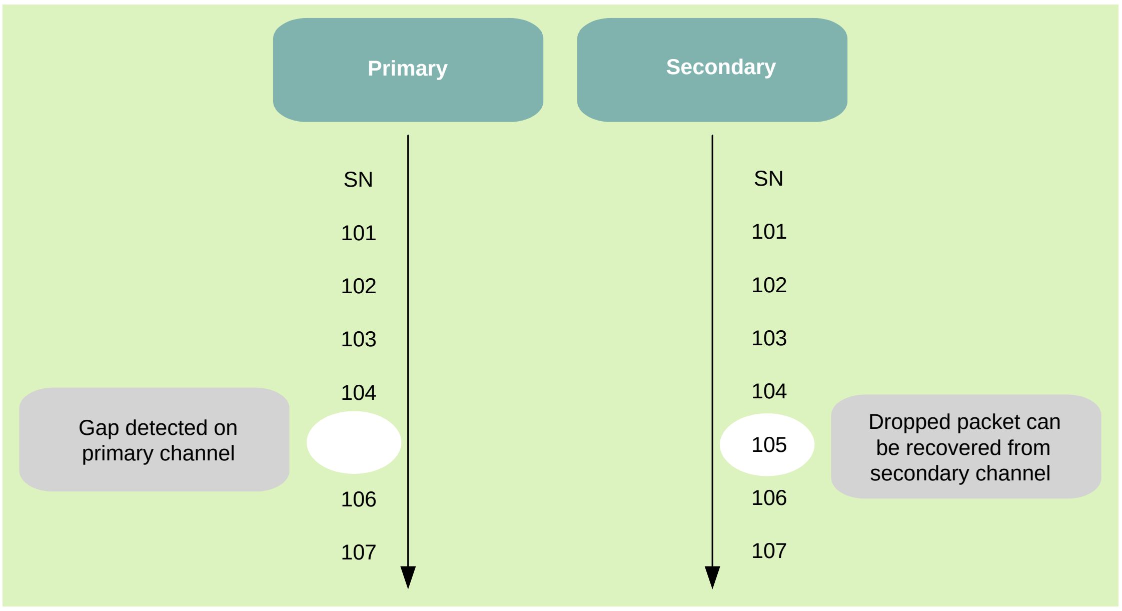
SNs are unique for each channel however they do not increase monotonically. The sequence number increase as shown in the table above.

Line A and line B are identical in terms of:

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

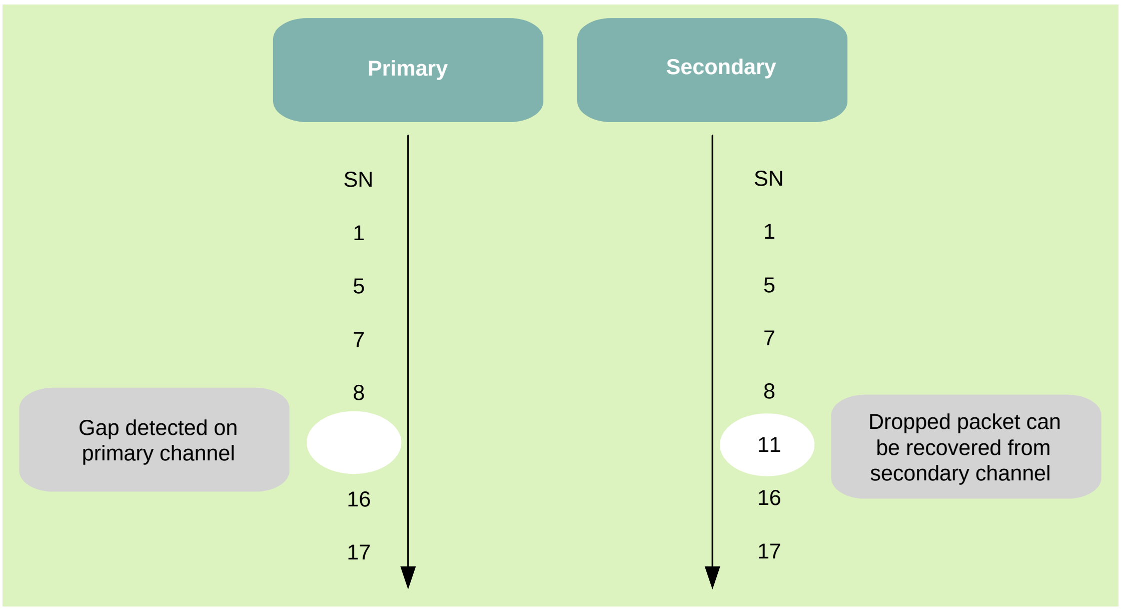
Client applications should listen to both channels in real-time. Clients should look at packets coming from both lines and process the ones that arrive first, regardless of whether they came from line A or line B. It is advisable to apply the ‘first come – first served’ rule.

Figure 4 Packet Header Sequence Numbers (Assuming 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


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

1.3.8 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 XDP Integrated feed provides the following 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

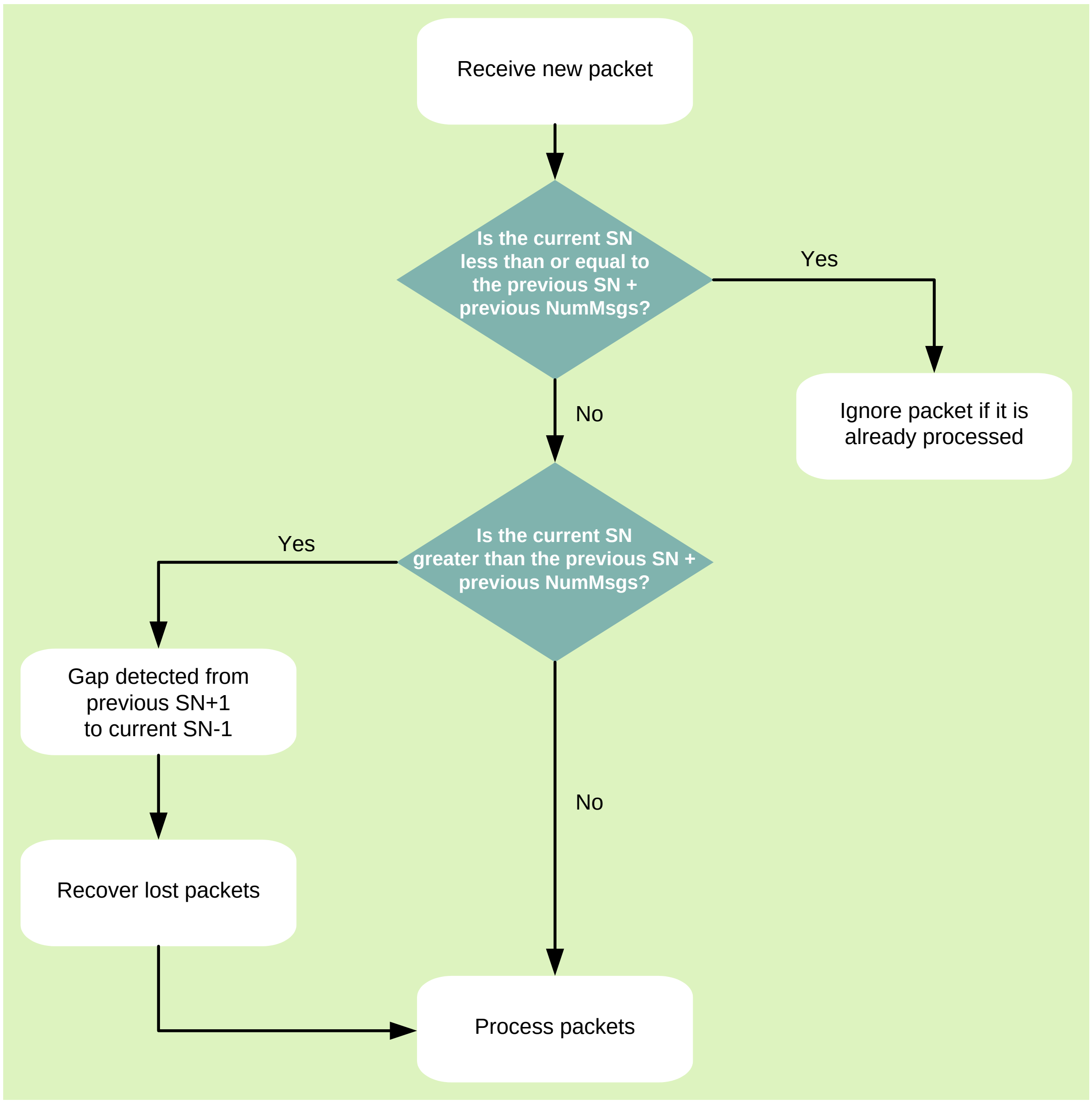
These mechanisms should be used as follows:

Table 7 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

Figure 5 illustrates how the SN should be used to detect gaps in the feed.

Figure 5 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

1.3.9 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*).

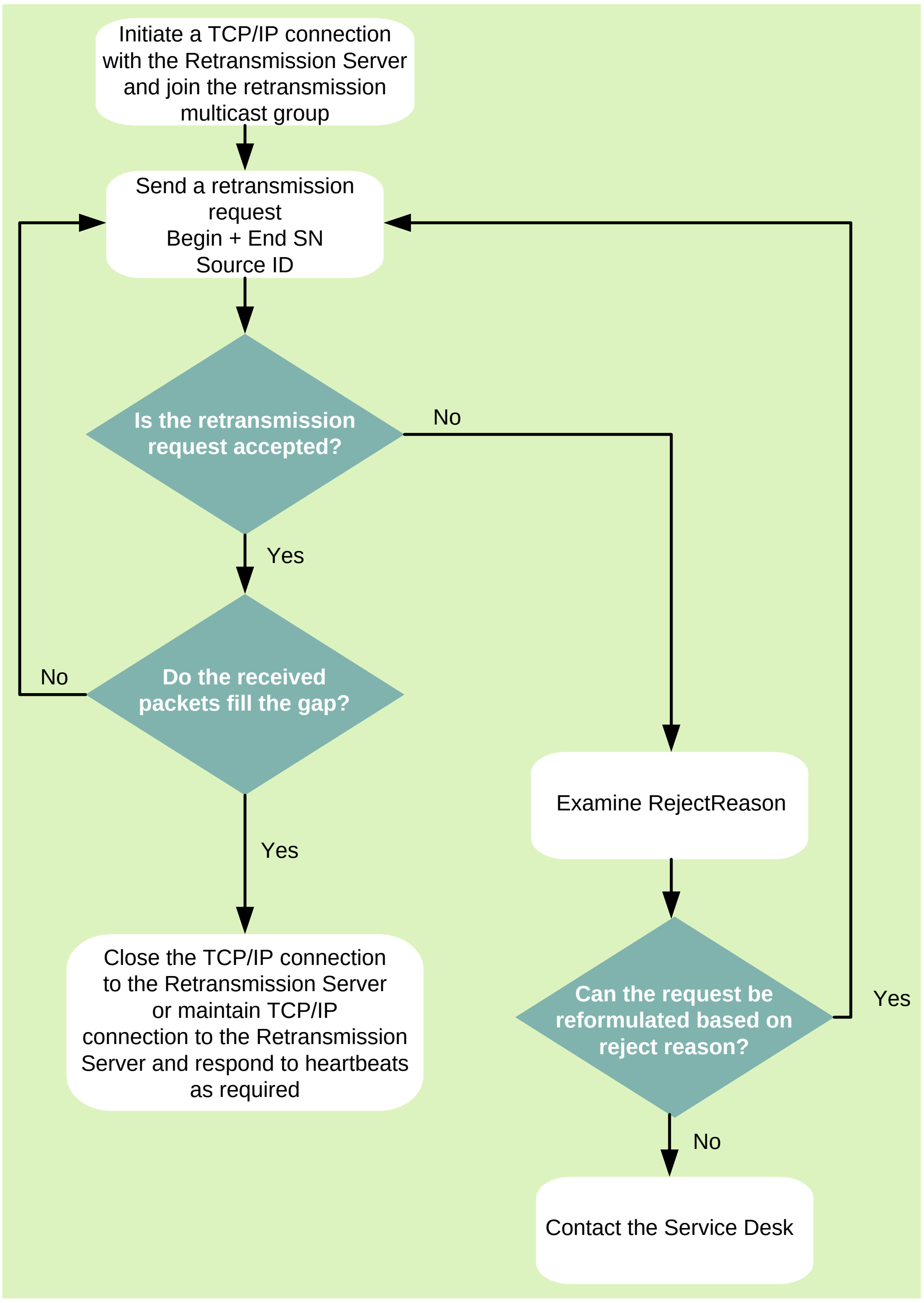
Retransmission requests should contain a Start SN, an End SN and a Source ID. The Source ID identifies the client application, and will be supplied by the exchange. The request can be rejected for a number of reasons, see Request Response Message (Msg Type ‘11’).

The number of retransmissions allowed per client per day is limited, see Retransmission/Refresh Request Limitations.

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

Figure 6 illustrates the process of requesting dropped packets from the Retransmission Server.

Figure 6 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
Close the TCP/IP connection to the Retransmission Server or maintain TCP/IP connection to the Retransmission Server and respond to heartbeats as required
Can the request be reformulated based on reject reason?
Yes
No
Contact the Service Desk

1.3.10 Refresh Server

If a client starts their application late, or experiences an outage, the Refresh Server should be used to provide the means to get back in synchronization with the real-time market. Clients must join the refresh multicast group first before sending and receiving the refresh request and response messages (ack/nack) over a TCP/IP connection.

After a client establishes a TCP/IP connection, the request 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 Refresh Server will close the connection.

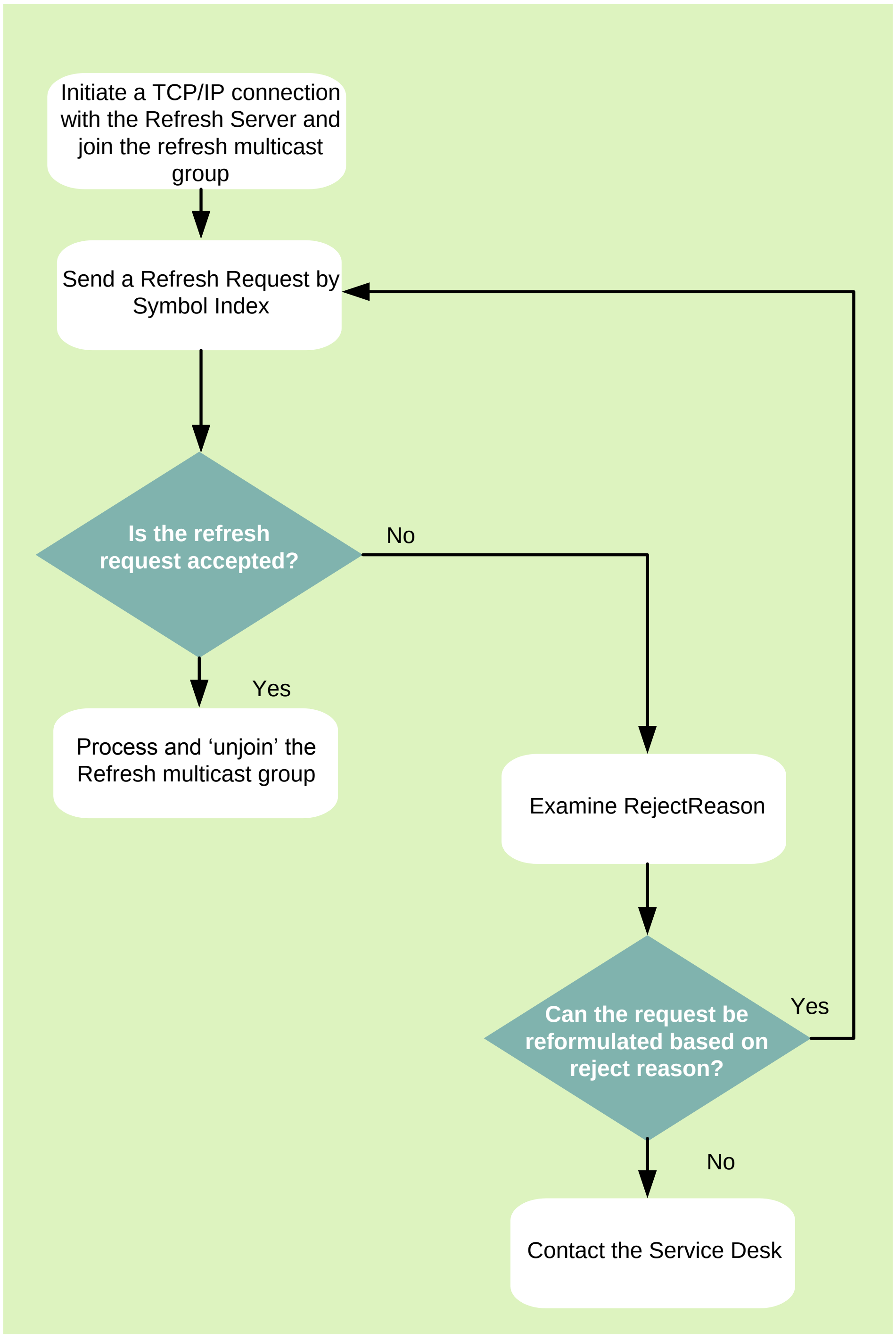
The client makes a TCP/IP connection to the Refresh Server to request a refresh of packets and after joining the multicast refresh groups. Refresh requests should contain a Product ID, Channel ID and a Source ID. The Product ID identifies the Product you are requesting a refresh for. The Channel ID will define for which multicast channel a refresh is required. The Source ID identifies the client application, and will be supplied by the exchange. This will be identical to the Source IDs allocated for the Retransmission Server. The request can be rejected for a number of reasons as defined in the Request Response message (see Msg Type '11').

Once a successful request has been received by the server, a refresh response will be sent to the client. Clients should then process the refresh content from the refresh multicast groups and synchronize this with the real time data.

The number of refreshes allowed per client per day is limited, Retransmission/Refresh Request Limitations.

Figure 7 illustrates the process of requests from the Refresh Server.

Figure 7 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

1.3.11 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:

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

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

1.3.12 Order IDs

The UTP matching engine uses a unique 64-bit order ID and consists of four concatenated data values - the Order ID, Market ID, System ID, and GTCIndicator. In all XDP feeds, the Market ID, and System ID are provided in the Symbol Index Mapping, since these values are static for the trading day. The Order ID and GTCIndicator are contained in the XDP data messages.

Depending upon the host byte alignment, the data structure will be different. **Table 8 Big-Endian System** and
Table 9 Little-Endian System depict the structure to properly combine the Order ID, Market ID, System ID, and GTCIndicator to obtain the same UTP matching engine Order ID sent on order acknowledge messages.

Table 8 Big-Endian System

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Order ID	0	4	Binary Integer	Order ID
Market ID	4	2	Binary Integer	ID of the Originating market in the Symbol Index Mapping
System ID	6	1	Binary Integer	ID of the Originating System in the Symbol Index Mapping
GTCIndicator	7	1	Binary Integer	This field specifies if Order ID is a GTC order: ■ 0 – Day Order ■ 1- GTC Order

Table 9 Little-Endian System

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
GTCIndicator	7	1	Binary Integer	This field specifies if Trade Order ID is a GTC order: ■ 0 – Day Order ■ 1- GTC Order
System ID	6	1	Binary Integer	ID of the Originating System in the Symbol Index Mapping
Market ID	4	2	Binary Integer	ID of the Originating market in the Symbol Index Mapping
OrderID	0	4	Binary Integer	Order ID

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 customers that 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.

- NYSE
  - Pipe-delimited: ftp://ftp.nyxdata.com/NYSESymbolMapping/NYSESymbolMapping_NMS.txt
  - XML format: ftp://ftp.nyxdata.com/NYSESymbolMapping/NYSESymbolMapping_NMS.XML
- NYSE ARCA
  - Pipe-delimited: ftp://ftp.nyxdata.com/ARCASymbolMapping/ARCASymbolMapping.txt
  - XML format: ftp://ftp.nyxdata.com/ARCASymbolMapping/ARCASymbolMapping.XML

The file format is as follows:

Table 10 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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
NYSE Market	Character	This field represents the exchange within NYSE Euronext trading the symbol: ■ 'N' – NYSE ■ 'P' – NYSE Arca ■ 'A' – NYSE MKT
Listed Market	Character	This field represents the exchange where the symbol is listed: ■ 'A' – NYSE MKT ■ 'N' – NYSE ■ 'P' – NYSE Arca ■ 'Q' – NASDAQ

FIELD NAME	FORMAT	DESCRIPTION
		■ 'T' – NASDAQ stock traded on NYSE Arca ■ 'U' – OTCBB symbol for Global OTC ■ 'V' – Other OTC symbols for Global OTC ■ 'Z' - BATS
TickerDesignation	Character	This represents on which ticker the symbol will be printed: ■ 'A' – CTA Tape A ■ 'B' – CTA Tape B ■ 'Q' – CTA Tape C
UOT	Numeric	This field represents the stock symbol's unit of trade.
PriceScaleCode	Numeric	This field contains the price scale for price conversion of the symbol. 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: $Price = \frac{Numerator}{10^{PriceScaleCode}}$ For example, a price of 27.56 is represented by a Numerator of 2756 and a PriceScaleCode equals to 2.
SystemID	Numeric	This represents the matching engine ID where the symbol is trading.
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, refer to Retransmission Message 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 that all configurations surrounding the secondary data center are included as described in Retransmission/Refresh Request Limitations.

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')

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. CONTROL MESSAGE SPECIFICATIONS

2.1 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 (for example, 'reset sequence number').
- Data messages are product-specific and control messages apply to all products.

2.2 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.

2.3 CONTROL MSG TYPES

The following table shows all of the messages types used by the XDP system.

Table 11 Control Message Types

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

2.4 SEQUENCE NUMBER RESET (MSG TYPE '1')

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

Table 12 Sequence Number Reset Message Fields

Header

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

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. '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 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 specifies the number represents the nanosecond portion of UTC time (since EPOCH)
ProductID	12	1	Binary Integer	The product ID used in the XDP header to identify the NYSE Euronext feed.
ChannelID	13	1	Binary Integer	This field contains the multicast channel ID over which the packet was sent.

2.4.1 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).

2.5 SOURCE TIME REFERENCE MESSAGE (MSG TYPE '2')

Time Reference messages are sent **only** in the production multicast streams and represent the whole number of seconds in UTC time from the Matching Engines. These messages are sent out when required, not during specific time intervals.

The Source Time Reference message is sent inline per symbol and has a valid sequence number. The messages will not be sent if there is a long period of inactivity for a symbol. Note that, unlike typical control messages, Source Time Reference messages can come in packets containing market data messages.

Table 13 Time Reference Message Fields

Header

FIELD	VALUE
PktSize	'32 Bytes'
DeliveryFlag	
NumberMsgs	
SeqNum	Valid sequence number
SendTime	
SendTimeNS	

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. '16 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '2' – Time Reference Message
SymbolIndex	4	4	Binary Integer	This field identifies the numerical representation of the symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
SymbolSeqNum	8	4	Binary Integer	This field contains the symbol sequence number
TimeReference	12	4	Binary	This field represents the UTC time (EPOCH) in

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	seconds

2.6 HEARTBEAT

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

Table 14 Heartbeat Message Fields

Header

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

2.6.1 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, see Sequence Numbers.

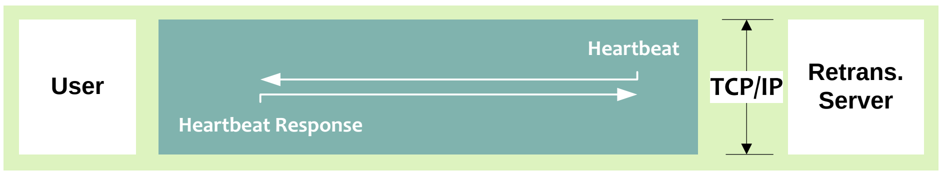
2.6.2 Retransmission and Refresh Heartbeat Processing Notes (TCP)

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 8 Retransmission/Refresh Server Heartbeats



2.7 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.

Table 15 Heartbeat Response Message Field

Header

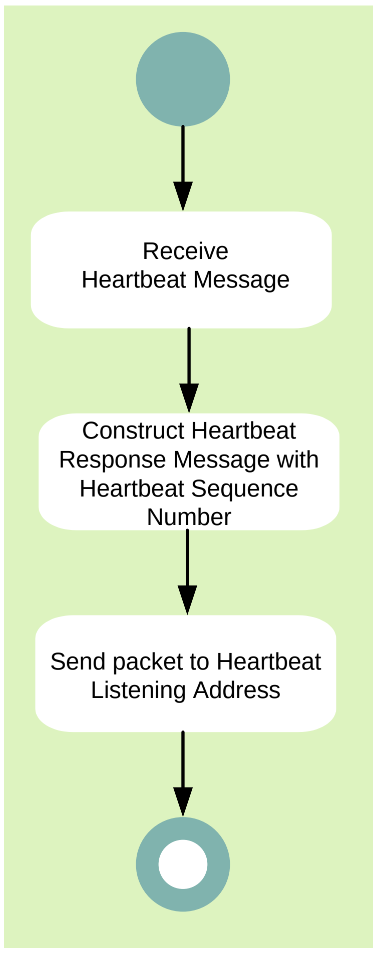
FIELD	VALUE
PktSize	'30 Bytes'
DeliveryFlag	Valid values include: '11' – Original Message Only
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

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. '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.

2.7.1 Heartbeat Message Processing

Figure 9 Processing of Heartbeat Messages


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

2.8 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.

Table 16 Retransmission Request Message Fields

Header

FIELD	VALUE
PktSize	’40 Bytes’
DeliveryFlag	Valid values include: ‘11’ – Original Message uncompressed
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Msg Body

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 Euronext feed.
ChannelID	23	1	Binary Integer	This field contains the multicast channel ID where you would like to request data from

2.9 REFRESH REQUEST (MSG TYPE '15')

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

Table 17 Refresh Request Message Field

Header

FIELD	VALUE
PktSize	'36 Bytes'
DeliveryFlag	Valid values include: '11' – Original Message uncompressed
NumberMsgs	1
SeqNum	
SendTime	
SendTimeNS	

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. '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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets. 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 Euronext feed.
ChannelID	19	1	Binary Integer	This field contains the multicast channel ID where you would like to request data from

2.10 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 or symbol mapping messages.

Table 18 Request Response Message Fields

Header

FIELD	VALUE
PktSize	
DeliveryFlag	Valid values include: '11' – Original Message uncompressed
NumberMsgs	1
SeqNum	
SendTime	
SendTimeNS	

Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. ‘29’ Bytes
MsgType	2	2	Binary Integer	This field identifies the type of message. ‘11’ – Retransmission 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.
BeginSeqNum	8	4	Binary Integer	The beginning sequence number of the requested range of messages to be retransmitted. **Note**: This field applies only to retransmission requests. If the response is for a refresh or symbol mapping request, it defaults to 0.
EndSeqNum	12	4	Binary Integer	The end sequence number of the requested range of messages to be retransmitted. **Note**: This field applies only to retransmission requests. If the response is for a refresh or symbol mapping request, it defaults to 0.
SourceID	16	10	ASCII String	This field represents the name of the source requesting retransmission. This field is 10 characters, null terminated.
ProductID	26	1	Binary Integer	The product ID used in the XDP header to identify the NYSE Euronext feed.
ChannelID	27	1	Binary Integer	This field contains the multicast channel ID where you requested data from
Status	28	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)

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				■ '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 current state if necessary. ■ '7' – Rejected due to an Invalid Channel ID ■ '8' – Rejected due to an Invalid Product ID ■ '9' – Rejected due to: 1) Invalid MsgType, or 2) Mismatch between MsgType and MsgSize

2.11 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.

Table 19 Retransmission Message Packet Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	■ '13' – Only one packet in retransmission update Uncompressed ■ '15' – Part of a retransmission sequence 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
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

2.11.1 Retransmission Message 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.

2.12 SYMBOL INDEX MAPPING REQUEST MESSAGE (MSG TYPE '13')

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

Table 20 Symbol Index Mapping Request Message Fields

Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ■ '11' – Original Message uncompressed
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

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 Integer	This field identifies the numerical representation of the symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets. SymbolIndex value can be zero, which is to request all symbol mapping for the multicast group.*

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
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 Euronext feed.
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

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

2.13 SYMBOL INDEX MAPPING MESSAGE (MSG TYPE '3')

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

- **Refreshes of a Single Symbol** The refresh of a single symbol index mapping message spans only a single packet and contains a Delivery Flag of 17 or 37 to indicate a single update only.
- Refreshes of All Symbols Start, Part and End Delivery Flags are used for refreshes of all symbols. All packets of the first symbol are marked Start and all packets of the last symbol are marked End. This applies also to refreshes of all Symbol Index Mapping messages.

Table 21 Symbol Index Mapping Message Fields

Header

FIELD	DESCRIPTION
PktSize	This field indicates the size of the message body in bytes.
DeliveryFlag	Valid values include: ■ '11' – Original Message 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
NumberMsgs	
SeqNum	
SendTime	

FIELD	DESCRIPTION
SendTimeNS	

Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
Msg Size	0	2	Binary Integer	Size of the message '44 bytes'
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. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
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 – NYSE Arca Cash ▪ 4 - NYSE/Arca Options ▪ 5 - NYSE/Arca Bonds ▪ 6 – Global OTC ▪ 7 – LIFFE ▪ 8 – NYSE Amex Options ▪ 9 - NYSE MKT Cash
System ID	22	1	Binary Integer	ID of the Originating System.
Exchange Code	23	1	ASCII Character	Exchanges where it is listed: ▪ 'N' – NYSE ▪ 'P' – NYSE Arca ▪ 'Q' – NASDAQ

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				■ ‘A’ – NYSE MKT ■ ‘U’ – OTCBB symbol for Global OTC ■ ‘V’ – Other OTC symbols for Global OTC ■ ‘Z’ – BATS Note: ‘N’, ‘Q’ and ‘A’ are applicable to NYSE and NYSE 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 applicable only for Arca: ■ ‘A’ – ADR ■ ‘C’ - COMMON STOCK ■ ‘D’ – DEBENTURES ■ ‘E’ – ETF ■ ‘F’ – FOREIGN ■ ‘H’ – AMERICAN DEPOSITARY SHARES ■ ‘I’ – UNITS ■ ‘L’ – INDEX LINKED NOTES ■ ‘M’ - MISC/LIQUID TRUST ■ ‘O’ – ORDINARY SHARES ■ ‘P’ - PREFERRED STOCK ■ ‘R’ – RIGHTS ■ ‘S’ - SHARES OF BENEFICIARY INTEREST ■ ‘T’ – TEST ■ ‘U’ – UNITS ■ ‘W’ – WARRANT Type of Security applicable only for NYSE and NYSE MKT: ■ ‘A’ – COMMON STOCK ■ ‘B’ – PREFERRED STOCK ■ ‘C’ – WARRANT

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				■ 'D' – RIGHT ■ 'E' – CORPORATE BOND ■ 'F' – TREASURY BOND ■ 'G' – STRUCTURED PRODUCT ■ 'H' – ADR COMMON ■ 'I' – ADR PREFERRED ■ 'J'-ADR WARRANTS ■ 'K' – ADR RIGHTS ■ 'L' – ADR CORPORATE BOND ■ 'M' – NY REGISTERED SHARE ■ 'N' – GLOBAL REGISTERED SHARE ■ 'O' – INDEX ■ 'P' – FUND ■ 'Q' – BASKET ■ 'R' – UNIT ■ 'S' – LIQUIDATING TRUST ■ 'U' - UNKOWN
Lot Size	26	2	Binary Integer	Round lot size.
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
MPV	38	2	Binary Integer	Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
Unit of Trade	40	2	Binary	This field specifies the security Unit of Trade.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
			Integer	■ 1 – 1 Share Unit ■ 10 – 10 Share Unit ■ 50 – 50 Share Unit ■ 100 – 100 Share Unit Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
LRP	42	2	Binary Integer	This field contains the LRP percentage value to calculate the High and Low LRPs. Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.

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

2.14 VENDOR MAPPING MESSAGE (MSG TYPE '4')

This message will be sent to inform the subscribers of other vendor symbology that maps to NYSE Symbols

Table 22 Vendor Mapping Fields

Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ■ '11' – Original Message 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
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '37 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '4' – Vendor Mapping Message
SymbolIndex	4	4	Binary Integer	This field identifies the numerical representation of the NYSE symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
VendorID	8	8	Binary Integer	**See Vendor Mapping Table below**
VendorID2	16	20	ASCII	**See Vendor Mapping Table below**
Vendor	36	1	Binary Integer	This field identifies which vendor this mapping applies to: ■ 0 – Comstock ■ 1 – Bloomberg

Vendor Mapping Table

VENDOR	VENDORID	VENDORID2
Comstock	Not Used	This field contains the Comstock symbol
Bloomberg	Unique global identifier assigned by BloombergGlobalID (ID_BB_Global) for all securities across every asset class	Unique Bloomberg Security identifier (BSID) including the Bloomberg source for all Bloomberg securities as an integer value. This field is used for B Pipe and subscription services. (ID_BB_SEC_NUM_SRC)

Note: The vendor mapping message will be a future enhancement

2.15 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.

Table 23 Message Unavailable Message Fields

Header

FIELD	DESCRIPTION
PktSize	

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

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 Integer	The product ID used in the XDP header to identify the NYSE Euronext feed.
ChannelID	13	1	Binary Integer	This field contains the multicast channel ID where you requesting the index map from.

2.16 SYMBOL CLEAR MESSAGE (MSG TYPE '32')

The following table describes the body fields of a Symbol Clear message. This message is sent by NYSE Euronext in response to issues with a specific symbol, i.e. system problems that corrupt order book, a local failover, or DR situation.

Table 24 Symbol Clear Message Fields

Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ■ '11' – Original Message Uncompressed
NumberMsgs	
SeqNum	

FIELD	DESCRIPTION
SendTime	
SendTimeNS	

Msg Body

FIELD NAME	OFFSET	SIZE	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. '32' – Symbol Clear
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 specifies the number represents the nanosecond portion of UTC time (since EPOCH)
SymbolIndex	12	4	Binary Integer	This field identifies the numerical representation of the symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
NextSourceSeqNum	16	4	Binary Integer	This field contains the source sequence number value that the recipient should expect in the immediately succeeding data packet for the specific symbol

2.17 TRADING SESSION CHANGE MESSAGE (MSG TYPE '33')

This message is sent in response to a trading session change for a specific symbol. All open order that is not in the same Trading Session should be deleted for that symbol.

Table 25 Trading Session Change Message Fields

Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ■ 11 – Original Message Uncompressed
NumberMsgEntries	
SeqNum	
SendTime	
SendTimeNS	

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. 33 – Symbol Trading Session Change
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 specifies the number represents the nanosecond portion of UTC time (since EPOCH)
SymbolIndex	12	4	Binary Integer	This field identifies the numerical representation of the symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
SymbolSeqNum	16	4	Binary Integer	This field contains the symbol sequence number
Trading Session	20	1	Binary Integer	A Trading session can be a combination of the three-bit values: ▪ 0x01 - Ok for morning hours ▪ 0x02 - Ok for national hours (core) ▪ 0x04 - Ok for late hours

2.17.1 Message Processing Rules

1. When a trading session message is received, all orders that are not eligible for the current or future trading sessions should be deleted from the book. No explicit deletes will be sent.
2. The NYSE will only use value 0x02 – National Hours(core market hours)

2.18 SECURITY STATUS MESSAGE (MSG TYPE '34')

This message will be sent to inform the subscribers of Trading Halts on the securities traded.

Table 26 Security Status Message Fields

Header

FIELD	DESCRIPTION
PktSize	
DeliveryFlag	Valid values include: ■ 11' – Original Message Uncompressed
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

Msg Body

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. Current State: 22 Bytes After Future Enhancements (Noted in each field): 46 bytes
MsgType	2	2	Binary Integer	This field identifies the type of message. 34 – Security Status 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 specifies the number represents the nanosecond portion of UTC time (since EPOCH)
SymbolIndex	12	4	Binary Integer	This field identifies the numerical representation of the symbol. This field is unique for products within each respective market and cannot be used to cross reference a security between markets.
SymbolSeqNum	16	4	Binary Integer	This field contains the symbol sequence number
Security Status	20	1	ASCII Character	The following are Halt Status Codes: ■ '3' - Opening Delay

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				■ ‘4’ - Trading Halt ■ ‘5’ - Resume ■ ‘6’ - No open/no resume The following are Short Sale Restriction Codes: ■ ‘A’ – Short Sale Restriction Activated (Day 1) ■ ‘C’ – Short Sale Restriction Continued (Day 2) ■ ‘D’ - Short Sale Restriction Deactivated ■ ‘E’ – Short Sale Restriction in Effect (The following condition will appear only in the event of a prior day correction/cancel affecting the Short Sale restriction. Applicable only to Arca) **Note:** For NYSE Arca, only the following values are applicable: ‘4’, ‘5’ and ‘E’. NYSE Market State values : ■ ‘O’ – Opened ■ ‘P’ – Pre-opening ■ ‘X’ -- Closed The following values are the Price Indication values: ■ ‘T’ – T - Time ■ ‘I’ – Price Indication ■ ‘G’ – Pre-Opening Price Indication ■ ‘R’ – Rule 15 Indication.
Halt Condition	21	1	ASCII Character	■ ‘0x20’ – Not applicable ■ '~' - Security not delayed/halted ■ 'D' - News dissemination ■ 'I' - Order imbalance ■ 'P' - News pending ■ 'M' – LULD pause ■ 'S' - Related security (not used) ■ 'X' - Equipment changeover

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				■ 'Z' - No open/No resume Market Wide Circuit Breakers are as follows: ■ '1' - Market Wide Circuit Breaker (MWCB) Halt Level 1. ■ '2' - Market Wide Circuit Breaker (MWCB) Halt Level 2. ■ '3' - Market Wide Circuit Breaker (MWCB) Halt Level 3.
Transaction ID	22	4	Binary Integer	This is a unique key per symbol across all messages within a transaction. Users will be able to map the Transaction ID across the respective depth of book, trades, BBO, and Integrated Book products for each market. Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
Price 1	26	4	Binary Integer	Default value is 0. ■ If securityStatus = 'A', then this is the SSR Triggering Trade Price ■ If securityStatus = ‘G’, then this is Pre-Opening Low Price Indication. ■ If securityStatus = ‘I’, then this is Low Price Indication ■ If securityStatus = ‘R’, then this is Rule 15 Low Indication Price. Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
Price 2	30	4	Binary Integer	Default value is 0 ■ If securityStatus = ‘I’, then this is High Price Indication ■ If securityStatus = ‘G’, then this is Pre-Opening Price Indication ■ If securityStatus = ‘R’, then this is Rule 15 High Price Indication Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
SSR Triggering Exchange ID	34	1	Binary Integer	Default value is 0. This field is only populated when securityStatus = 'A'. This field is the Binary representation of the character values mentioned below. Valid Values are: ■ 'N' – NYSE ■ 'P' – NYSE Arca ■ 'Q' – NASDAQ ■ 'A' – NYSE MKT ■ 'U' – OTCBB symbol for Global OTC ■ 'V' – Other OTC symbols for Global OTC ■ 'B' – NASDAQ OMX BX ■ 'C' – NSX ■ 'D' – FINRA ■ 'I' – ISE ■ 'J' – EDGA ■ 'K' – EDGX ■ 'M' – CHX ■ 'N' – NYSE ■ 'S' – CTS ■ 'T' – NASDAQ OMX ■ 'W' – CBSX ■ 'X' – NASDAQ OMX PSX ■ 'Y' – BATS Y ■ 'Z' – BATS Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability..
SSR Triggering Volume	35	4	Binary Integer	Default value is 0. This field is only populated when securityStatus = 'A' Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
Time	39	4	Binary Integer	Default value is 0. ■ If securityStatus = 'A' , then this is SSR Trigger Time ■ If securityStatus = 'T', then it is T-Time Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability..
SSRState	43	1	ASCII Character	Short Sale Restriction values: ■ '~' – No Short Sale in Effect ■ 'E' – Short Sale Restriction in Effect Note: For NYSE Arca, only the following values are applicable: 'E'. Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
MarketState	44	1	ASCII Character	This field is applicable to NYSE and NYSE MKT Market State values: ■ 'O' – Opened ■ 'P' – Pre-Opening ■ 'X' – Closed Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.
SessionState	45	1	ASCII Character	This field is applicable to NYSE Arca and is for future use. Session State values: ■ 'X' – Early Session State ■ 'Y' – Core Session State ■ 'Z' – Late Session State Note: This field is not yet implemented and is left as a future release. Clients will be notified upon availability.

2.19 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. Note the following:

- **Refreshes of a Single Symbol** The Delivery Flags of all refreshes of a single symbol that span multiple packets contain 'Part' indications only. You use the PktNum and NumPkts fields to know when the complete refresh has been received.
- **Refreshes of All Symbols** Start, Part and End Delivery Flags are used for refreshes of all symbols. All packets of the first symbol are marked Start and all packets of the last symbol are marked End. This applies also to refreshes of all Symbol Index Mapping messages.

Table 27 Refresh Header Message Fields

Header

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
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

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. '16 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '35' – Refresh Header Message
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

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
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.
LastSymbolSeqNum	12	4	Binary Integer	This field contains the last symbol sequence number sent on the feed. The refresh is the symbol state of the book as of this sequence number.

2.19.1 Message Sending Rules

1. The first packet will contain the full 16-byte Refresh header message; every packet thereafter will contain an 8byte refresh header. The LastSeqNum and the LastSymbolSeqNum fields are removed to avoid sending duplicate information with every packet.
2. After the first packet the refresh header will look as follows:

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	This field indicates the size of the message body in bytes. '8 Bytes'
MsgType	2	2	Binary Integer	This field identifies the type of message. '35' – Refresh Header Message
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

2.19.2 Refresh Example

Assuming the refresh of symbol xyz requires three packets.

The first Packet structure will look as follows:

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

Table 28 Packet Header

FIELD	VALUE
PktSize	
DeliveryFlag	Valid values include: ■ '17' – Only one packet in Refresh update Uncompressed

FIELD	VALUE
	■ ‘18 – Start of Refresh Update Uncompressed ■ ‘19’ – Part of a Refresh sequence Uncompressed ■ ‘20’ – End of Refresh Update Uncompressed
NumberMsgs	
SeqNum	
SendTime	
SendTimeNS	

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

Table 29 Refresh Header

FIELD NAME	OFFSET	SIZE (BYTES)	VALUE
MsgSize	0	2	16 Bytes
MsgType	2	2	35’ – Refresh Header Message
CurrentRefreshPkt	4	2	1
TotalRefreshPkts	6	2	3
LastSeqNum	8	4	5000
LastSymbolSeqNum	12	4	6000

In the case of the Integrated Feed, the refresh header will be followed by the following message types:

1. Symbol Index Mapping Message (Msg Type 3)
2. Imbalance Message (Msg Type 105), if there is an imbalance message
3. Security Status Message (Msg Type 34), if there is a trading status notification
4. Trade Session Change (Msg Type 33), the most current trading session
5. Add Order Refresh (Msg Type 106)

3. PRODUCTION CONFIGURATION

3.1 INTRODUCTION

3.1.1 Data Content

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

3.1.2 Data Delivery

NYSE Euronext market data is available on the Mahwah LCN network.

3.1.3 XDP Production Hours

The following table describes the main daily event generating activity and indicative time on the XDP Integrated feed.

Table 30 Production Event Schedule

EVENT	NYSE/ NYSE MKT TIME (EST)	NYSE ARCA TIME (EST)	COMMENT
Sequence Number reset	~1:00am	~1:00am	
Symbol Mapping	~1:00am	~1:00am	
Pre-Open			
Open	9:30am	4:00am	Open time and close time are for the first financial products to open or close
Close	4:00pm	8:00pm	

3.2 MULTICAST/TCP SETUP

3.2.1 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 NYSE Arca 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.

3.2.2 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, that is, they are synchronized with each other. Each message from each feed contains the same packet sequence number.

3.2.3 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

3.2.4 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 Euronext will provide each client with a default of one Source ID for Production.

Each Source ID may only be logged on to a server once at a given time. Please contact NYSE Euronext Service Desk to setup a Source ID. The Source ID assigned will be nine characters, null terminated.

3.2.5 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.

3.2.6 Retransmission/Refresh Request Limitations

The following recommendations apply to Production.

Table 31 Retransmission/Refresh Request 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 ARCA Service Desk to get a unique Source ID.	N/A	Request will not be processed.
Limitation of Retransmission Message Requests	Only retransmission requests for 1000 messages or less will be honored.	1000	Request will not be processed.
Limitation of Generic Request Age in Seq Nums	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.	x	Request will not be honored.
Limitation of Index Mapping Requests	Up to x Symbol Index Mapping requests will be honored.	x	Request will not be honored.

3.2.7 Production Channel Configuration

See http://www.nyxdata.com/ipaddresses.

4. TEST DATA FEED CONFIGURATION

4.1 INTRODUCTION

4.1.1 Data Content

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

4.1.2 Data Delivery

NYSE Euronext market data is on the Mahwah LCN network

4.2 XDP TEST HOURS

The following table is the overview of the main daily event generating activity and indicative time on the XDP Integrated Feed. Testing hours are normal hours ~1:00 AM - ~ 8:15 PM. All data will come from the UTP Cert matching engine and order entry.

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, that is, 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 Production and CERT 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 five seconds to stay connected to the server.

4.3.5 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 Euronext will provide each client with a default of one Source ID for Production.

Each Source ID may only be logged on to a server once at a given time. Please contact the NYSE Euronext Service Desk to setup a Source ID. The Source ID assigned will be nine characters, null terminated.

4.3.6 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.

4.3.7 Test Retransmission/Refresh Request Limitations

The following recommendations apply to production and test.

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 Retransmission Message Requests	Only retransmission requests for 1000 messages or less will be honored.	1000	Request will not be processed.
Limitation of Generic Request Age in Seq Nums	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.	N/A	Request will not be honored.
Limitation of Index Mapping Requests	Up to 500 Symbol Index Mapping requests will be honored.	500	Request will not be honored.

4.3.8 Test Channel Configuration

See http://www.nyxdata.com/ipaddresses.