



# **XDP COMMON CLIENT SPECIFICATION**

**NYSE AMERICAN XDP FEEDS**

**NYSE ARCA INTEGRATED FEED**

**NYSE XDP FEEDS**

**PRODUCTION AUGUST 21, 2017**

**PRODUCTION OCTOBER 2, 2017**

**Version**

**2.1c**

**Date**

**August 11, 2017**

© Copyright 2017 Intercontinental Exchange, Inc. ALL RIGHTS RESERVED. INTERCONTINENTAL EXCHANGE, INC. AND ITS AFFILIATES WHICH INCLUDE THE NEW YORK STOCK EXCHANGE, ("ICE" AND "NYSE") MAKE NO WARRANTY WHATSOEVER AS TO THE PRODUCT DESCRIBED IN THESE MATERIALS EXPRESS OR IMPLIED, AND THE PRODUCT IS PROVIDED ON AN "AS IS" BASIS. ICE AND NYSE EXPRESSLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NEITHER ICE, NYSE NOR THEIR RESPECTIVE DIRECTORS, MANAGERS, OFFICERS, AFFILIATES, SUBSIDIARIES, SHAREHOLDERS, EMPLOYEES OR AGENTS MAKE ANY WARRANTY WITH RESPECT TO, AND NO SUCH PARTY SHALL HAVE ANY LIABILITY FOR (i) THE ACCURACY, TIMELINESS, COMPLETENESS, RELIABILITY, PERFORMANCE OR CONTINUED AVAILABILITY OF PRODUCT, OR (ii) DELAYS, OMISSIONS OR INTERRUPTIONS THEREIN. ICE AND NYSE DO NOT, AND SHALL HAVE NO DUTY OR OBLIGATION TO, VERIFY, MONITOR, CONTROL OR REVIEW ANY INFORMATION IN RELATION TO THE PRODUCT.

Global OTC is an Alternative Trading System ("ATS") registered with the U.S. Securities and Exchange Commission ("SEC") and operated by Archipelago Trading Services, Inc. ("ATSI"), a broker-dealer registered with the SEC and a member of the Financial Industry Regulatory Authority ("FINRA"). Although ATSI is a wholly-owned subsidiary of NYSE Group, Inc., Global OTC is not a stock exchange or self-regulatory organization. The OTC equity securities traded on Global OTC are not U.S. exchange listed securities.

## Preface

---

### DOCUMENT HISTORY

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

Version	Date	Change Description
<b>2.0</b>	01/08/2015	Re-branded for ICE Structure and explanatory text revised for clarity and simplicity NYSE Security Status msg's Transaction ID field changed to Reserved Source Time Reference msg's SymbolIndex field changed to System ID IBF 2.0
<b>2.0a</b>	02/16/2015	Noted shorter length of Symbol Index Mapping (msg type 3) and Security Status (msg type 34) for Arca Integrated Feed only Minor tweaks to language
<b>2.0b</b>	02/18/2015	Corrections in section 3.6.2 regarding Order ID's Corrected Packet Header DeliveryFlags for Seq Num Reset messages (4.3 and 9.2)
<b>2.0c</b>	03/10/2015	Corrected Security Status msg length to 46 Added detail about multicast priming in Section 9 Added detail about refresh formats in 5.1.3 Added detail about IP Table filtering in 8.4
<b>2.0d</b>	04/07/2015	Clarified descriptions and availability of last 8 fields in the Security Status msg
<b>2.0e</b>	05/12/2015	Modified Security Status msg, Security Status field to reflect Arca adoption of NYSE SSR values Clarified the description of Time field in Security status message and System ID field in Order Acknowledgment message
<b>2.0f</b>	06/30/2015	Updated the Exchange Code field values in the Symbol Index Mapping Message to include Global OTC primary symbols Updated SSR Triggering Exchange ID field values in Security Status Message Updated Listed Market values in the Symbol Index Mapping File format to include the Listed Market for Global OTC primary symbols
<b>2.0g</b>	07/10/2015	Updated legal disclaimer for Global OTC on title page
<b>2.0h</b>	10/13/2015	Updated the location of Symbol Index Mapping File
<b>2.0i</b>	Mar 18, 2016	Corrected fields in the symbol index ftp file
<b>2.0j</b>	Mar 28, 2016	Clarified/corrected production hours in section 9.4
<b>2.0k</b>	June 16, 2016	Restored guidance on handling the Session Change msg for users of Arca Integrated and ArcaBook feeds. This was incorrectly removed from version 2.0.
<b>2.0L</b>	Aug 18, 2016	Clarified trailing pipe characters in Symbol Index Mapping File (section 10)
<b>2.0m</b>	Oct 4, 2016	Updated Security Status message to include V as the exchange ID for IEX. Updated section 2.2 to reflect change of heartbeat frequency from 60 secs to 1 sec.
<b>2.1</b>	Jan 24, 2017	Updated section 3.6 (Order IDs and Trade IDs) Removed Trading Session Change msg. Security Status now communicates this. Removed value 0x20 from Halt Condition field of the Security Status msg Added values E & L to Security Status and Market State fields
<b>2.1a</b>	Feb 2, 2017	Symbol Mapping & Security Status msgs: corrected msg sizes – same in all markets Corrected MPV & Unit of Trade – populated in all markets
<b>2.1b</b>	May 26, 2017	Section 10 – added ftp file path for NYSE American Symbol Mapping file 8.4.2 – Updated request quota information 5.2 Corrected packetization information for Message Unavailable messages 3.2 Updated Time Reference message usage 3.7 Clarified Symbol Index uniqueness and intraday usage Updated links to the NYSE Symbology Spec
<b>2.1c</b>	Aug 11, 2017	Updated links and contact information on page 3 Updated 3.6 Order ID's and Trade ID's Corrected 8.4.2 guidance for handling exceeded quotas

## 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 connectivity documents](#)
- [NYSE Symbology Specification](#)

## CONTACT INFORMATION

### Service Desk

- Telephone: +1 212 383 3640 (International)
- Telephone: 866 873 7422 (Toll free, US only)

## FURTHER INFORMATION

- For additional product information please visit, [our NYSE Real Time Market Data pages](#)
- For updated capacity figures, visit our capacity pages at: <http://www.nyxdata.com/capacity>
- For details of IP addresses, download our [IP Addresses spreadsheet](#)

# Contents

---

## Preface 2

Document History .....	2
REFERENCE MATERIAL .....	3
CONTACT INFORMATION .....	3
FURTHER INFORMATION .....	3

## Contents..... 4

<b>1. Introduction .....</b>	<b>5</b>
1.1 Receiving Real Time Market Data.....	5
1.2 Recovering from Errors .....	5
<b>2. Packets and Heartbeats.....</b>	<b>6</b>
2.1 Packet Header .....	6
2.2 Heartbeats .....	6
<b>3. Message Field Content .....</b>	<b>7</b>
3.1 Message Header.....	7
3.2 Date and Time Conventions.....	8
3.3 Sequence Numbers .....	9
3.4 Symbol Sequence Numbers .....	9
3.5 Prices .....	9
3.6 Order ID's and Trade ID's .....	9
3.7 Symbol Indexes .....	10
<b>4. Messages Sent by the Publisher .....</b>	<b>11</b>
4.1 Symbol Index Mapping Message (Msg Type 3) .....	11
4.2 Security Status Message (Msg Type 34) .....	13
4.3 Sequence Number Reset Message (Msg Type 1) .....	16
4.4 Source Time Reference Message (Msg Type 2).....	16
4.5 Symbol Clear Message (Msg Type 32).....	17
<b>5. Messages Sent by Refresh and Retrans Servers Only .....</b>	<b>18</b>
5.1 Refresh Header (Msg Type 35).....	18
5.2 Message Unavailable Message (Msg Type 31) .....	20
<b>6. Messages Sent by the Client to the Request Server .....</b>	<b>21</b>
6.1 Retransmission Request Message (Msg Type 10).....	21
6.2 Refresh Request Message (Msg Type 15).....	22
6.3 Symbol Index Mapping Request Message (Msg Type 13).....	23
6.4 Heartbeat Response Message (Msg Type 12).....	23
<b>7. Messages Sent by Request Server .....</b>	<b>24</b>
7.1 Request Response Message (Msg Type 11) .....	24
<b>8. Error Handling and the Request Server .....</b>	<b>25</b>
8.1 Handling Sequence Number Gaps .....	25
8.2 Recovering from Client Late Starts or Intraday Failures.....	25
8.3 Refreshing Symbol Information .....	26
8.4 Request Server .....	26
<b>9. Operational Information .....</b>	<b>28</b>
9.1 System Behavior on Start and Restart .....	28
9.2 XDP Publisher Failover .....	28
9.3 Disaster Recovery Site.....	28
9.4 XDP Production Hours .....	28
9.5 XDP Test Hours .....	28
<b>10. Symbol Index Mapping File .....</b>	<b>29</b>

## 1. Introduction

---

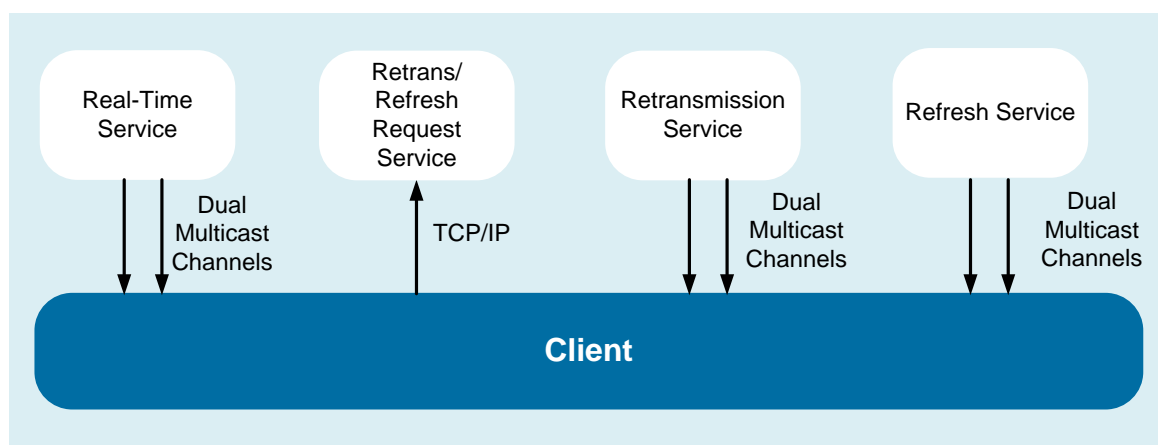
### 1.1 RECEIVING REAL TIME MARKET DATA

Real-time XDP data is published in the form of messages with fixed length fields. All fields are binary except a very small number that are in ASCII format. For efficient use of the network, the messages are bundled into application packets, and the packets are published via the multicast protocol.

For capacity reasons, packets are routed over a number of predefined data sets called channels. Each channel is duplicated and published to two distinct multicast groups for redundancy. The two redundant multicast groups per channel (called lines) are referred to as line A and line B. The union of the data in all channels that make up a product is called a feed.

The IP addresses and port numbers of the production and test channels for each XDP feed can be found at [www.nyxdata.com/ipaddresses](http://www.nyxdata.com/ipaddresses). A client application receives a product by subscribing to some or all of the channels that make up the feed.

### 1.2 RECOVERING FROM ERRORS



- In case of dropped multicast packets, the client can connect to a Request Server via TCP/IP to request retransmissions of missed messages.
- In case of client late start or intraday failure, the client can connect to the Request Server and request snapshot refreshes of the state of the market.
- At system startup, each channel publishes referential data about all symbols published on the channel. If a client process misses this initial spin of symbol data, he can connect to the Request Server and request a refresh of some or all of the missed data.

In response to these requests, retransmission and refresh data is published by the exchange over dedicated multicast channels which correspond one-to-one with the real-time channels.

See [Error Handling and the Request Server](#) for complete information.

## 2. Packets and Heartbeats

### 2.1 PACKET HEADER

All packets sent on any XDP feed have an XDP Packet Header followed by one or more messages (with the exception of Heartbeat packets which do not contain any messages).

The maximum length of a packet is 1400 bytes, so no message can be longer than 1400 – 16 bytes (max packet size - the length of the Packet Header).

#### Packet Header Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>PktSize</b>	0	2	Binary	The size of the packet in bytes, including this 16-byte packet header
<b>DeliveryFlag</b>	2	1	Binary	A flag that indicates whether this is an original, retransmitted, or 'replayed' message. Valid values include: <ul style="list-style-type: none"> <li>1 – Heartbeat</li> <li>10 – XDP Failover (see <a href="#">XDP Publisher Failover</a>)</li> <li>11 – Original Message</li> <li>12 – Sequence Number Reset Message</li> <li>13 – Only one packet in retransmission sequence</li> <li>15 – Part of a retransmission sequence</li> <li>17 – Only one packet in Refresh sequence</li> <li>18 – Start of Refresh sequence</li> <li>19 – Part of a Refresh sequence</li> <li>20 – End of Refresh sequence</li> <li>21 – Message Unavailable</li> </ul>
<b>NumberMsgs</b>	3	1	Binary	The number of messages in this packet
<b>SeqNum</b>	4	4	Binary	The message sequence number of the first message in this packet
<b>SendTime</b>	8	4	Binary	The time when this packet was published to the multicast channel, in seconds since Jan 1, 1970 00:00:00 UTC.
<b>SendTimeNS</b>	12	4	Binary	The nanosecond offset from the Send Time

### 2.2 HEARTBEATS

To assist the client in confirming connection health, application heartbeats are sent once a minute by the Request Server, and once a second by the real-time publishing servers (data, refresh and retransmissions channels).

A heartbeat consists of a packet containing a Packet Header and no messages. The Packet Header's Delivery Flag is set to 1 and Number Msgs is 0. Since a heartbeat packet contains no messages, a heartbeat does not increment the next expected sequence number. See [Sequence Numbers](#).

Heartbeats sent by the Request Server must be acknowledged by the client. See [Request Server](#).

### 3. Message Field Content

Messages are contiguous data structures consisting of fixed-length fields. No names or 'tags' appear in the message.

- Message fields align on 1 byte boundaries, so there are no filler fields for alignment purposes
- Binary fields are published in Little-Endian ordering
- All ASCII string fields are left aligned and null padded
- Segmentation of messages across packets is not supported, so a message will never straddle a packet boundary.
- The length of a message as actually published may differ from the length of the message structure defined in the client specifications. See [Msg Size Field](#) below for details.

#### 3.1 MESSAGE HEADER

The format of each message varies according to type, but each type starts with a standard 4-byte message header:

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	The size of this message in bytes
<b>MsgType</b>	2	2	Binary	The type of this message

##### 3.1.1 Msg Size Field

In order to handle future releases of XDP feeds smoothly, clients should never hard code msg sizes in feed handlers. Instead, the feed handler should use the Msg Size field to determine where the next message in a packet begins.

This allows

- Support of XDP format variations among markets
- Client flexibility when revised message structures go live in production

In example 1 below, a message type is defined in the specification to have different lengths in different markets. The trailing field is not published in the Arca market. An Arca-coded client can process NYSE data correctly (but of course cannot use the trailing Volume field without field-specific coding).

#### Example 1: Message type with format variations across markets

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>Msg Size</b>	0	2	Binary	Size of the message. NYSE – 24 bytes NYSE American – 24 bytes NYSE Arca - 20 bytes
<b>Msg Type</b>	2	2	Binary	The type of this message: 998 – Example 1 msg type
<b>SourceTimeNS</b>	4	4	Binary	
<b>SymbolIndex</b>	8	4	Binary	
<b>OrderID</b>	12	4	Binary	
<b>Price</b>	16	4	Binary	
<b>Volume</b>	28	4	Binary	Not published in Arca market

Look at the Msg Size field to know where the next message starts.

Market-specific content

The variable message size can also insulate client code from future field additions that you may not need.

In example 2, an existing message type is 16 bytes long.

### Example 2: Release N

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>Msg Size</b>	0	2	Binary	Size of the message: 16 bytes
<b>Msg Type</b>	2	2	Binary	The type of this message: 999 – Price message example
<b>SourceTimeNS</b>	4	4	Binary	
<b>SymbolIndex</b>	8	4	Binary	
<b>Price</b>	12	4	Binary	

Look at the Msg Size field to know where the next message starts.

In a future release, a four-byte volume field will be added, increasing the Msg Size to 20 bytes.

If the client wishes to delay upgrading his feed handler for the new content, no coding is needed at the time of the release. Proper coding of the MsgSize field up front allows the client to handle the unforeseen 20-byte format. On his own schedule, the client can upgrade his feed handler to process the new field.

### Example 2: Release N+1: a new field is added

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>Msg Size</b>	0	2	Binary	Size of the message: 20 bytes
<b>Msg Type</b>	2	2	Binary	The type of this message: 999 – Price message example
<b>SourceTimeNS</b>	4	4	Binary	
<b>SymbolIndex</b>	8	4	Binary	
<b>Price</b>	12	4	Binary	
<b>Volume</b>	16	4	Binary	New field

Look at the Msg Size field to know where the next message starts.

Unmodified clients can handle longer message structure (but can't benefit from new content)

## 3.2 DATE AND TIME CONVENTIONS

Dates and times are in UTC (Universal Time, Coordinated), and are expressed in nanoseconds since the Unix Epoch (Jan 1, 1970 00:00:00). A complete timestamp consists of two 4-byte fields: seconds since the Unix Epoch, and nanoseconds within the current second, as in a Unix timespec structure.

The XDP Packet Header contains SendTime and SendTimeNS fields to show the time that the packet was published to the wire by the XDP Publisher.

Most XDP messages additionally contain a timestamp called Source Time to show the time of the Matching Engine event that caused the publication of this message.

Many of the higher-volume XDP feeds such as Integrated and BBO explicitly publish only the nanoseconds portion of the Source Time in each message. The seconds portion is explicitly published in a [Source Time Reference Message \(Msg Type 2\)](#) once a second.

**Source Time Reference messages are published per Matching Engine partition (per TXN, which is equivalent to the Integrated Feed channel number).**



### 3.3 SEQUENCE NUMBERS

Each message in a given channel is assigned a unique sequence number. Sequence numbers increase monotonically per channel, and can be used to detect publication gaps.

To optimize publication efficiency, the sequence number is not explicitly published in each message. Instead, the Packet Header contains the sequence number of the first message in the packet, along with the number of messages in the packet. Using these fields, the client can easily associate the correct sequence number with each message.

The sequence number combined with the channel ID form a message ID which is unique across the feed.

### 3.4 SYMBOL SEQUENCE NUMBERS

In addition to the sequence number, many message types explicitly include a field called Symbol Sequence Number, which identifies the message's position in the sequence of all messages published by the feed for a given symbol.

Clients who are tracking only a small number of symbols may opt to ignore sequence numbers and track only Symbol Sequence Numbers for each symbol of interest. If such a client ever experiences a Symbol Sequence Number gap, he can request a refresh for that symbol.

### 3.5 PRICES

All price fields are published as unsigned binary integers. To interpret a price correctly, the client must use the published price value as a numerator along with the Price Scale Code in the symbol's [Symbol Index Mapping Message \(Msg Type 3\)](#) as follows:

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

For example, a price of \$27.56 is represented as a published price field of 2756 and a PriceScaleCode of 2.

### 3.6 ORDER ID'S AND TRADE ID'S

The Order ID and the Trade ID in order-based feeds such as NYSE Integrated Feed are binary integers that uniquely identify an order or an execution. Order and Trade IDs are valid for the trading day only.

#### 3.6.1 NYSE American and Arca (Pillar matching engine, Integrated Feed v2.1)

Trade IDs are 4 bytes long and correspond to the lowest 4 bytes of the 8-byte Deal ID field in the gateway Order Ack. The Trade ID is unique per ME symbol partition (System ID in the Symbol Index Mapping message), so unique per symbol. Prepend 3 fields to the Trade ID as discussed below to make a unique match across markets to the gateway Order Ack message (the Deal ID field in Pillar Gateway, or the ArcaExID in the legacy Gateway) .

Order IDs are 8 bytes long and correlate uniquely across markets to the 8 byte OrderID in the gateway Execution Report.

#### 3.6.2 NYSE Tape A (UTP matching engine, NYSE legacy gateways, Integrated Feed v2.1)

Trade IDs are 4 bytes long and correspond to the DBExecID field in the gateway Order Ack. The Trade ID is unique per ME symbol partition (System ID in the Symbol Index Mapping message), so unique per symbol. Prepend 3 fields to both the Trade ID and the Order Ack's DBExecID as discussed below to make a unique match across markets.

Order IDs are 8 bytes long, but the upper 4 bytes are not populated. The lower 4 bytes correspond to the 4 byte MEOrderID in the gateway Execution Report and are unique per matching engine instance. Prepend 3 fields to both the Order ID and the Execution Report's MEOrderID as discussed below to make a unique match across markets.

#### 3.6.3 NYSE Tapes B and C (Pillar matching engine, NYSE legacy gateways, Integrated Feed v2.1)

Remarks on Trade IDs are the same as they are for NYSE Tape A above.

Order IDs are 8 bytes long. Their lower 4 bytes correspond to the 4 byte MEOrderID in the gateway Execution Report. Prepend 3 fields to the Execution Report's MEOrderID as discussed below to make a unique match across markets.

### Matching 4-byte with 8-byte ID fields in the market data vs. the order entry messaging

By combining a 4-byte Order ID or Trade ID with the Market ID and System ID from the Symbol Mapping message as shown below, you can obtain the corresponding 8-byte ID. The table assumes the client byte ordering is Little Endian. If the client byte ordering is Big Endian, the byte order is reversed.

XDP FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
	0	1	Binary	0
<b>System ID</b>	1	1	Binary	Unique ID for a single matching engine instance (Pillar Symbol Partition or UTP TU). This value is found in the Symbol Index Mapping message's ID field
<b>Market ID</b>	2	2	Binary	ID of the Originating market in the Symbol Index Mapping
<b>OrderID or TradeID</b>	4	4	Binary	Contents of 4-byte field being disambiguated

### Field sizes and correlation procedures for making unique matches across markets

	GATEWAY		XDP V2.1
American and Arca	<b>Order ID</b>	<b>8 bytes</b>	<b>8 bytes</b>
	<b>Trade ID</b>	<b>8 bytes</b>	4 bytes, prepend 3 fields
NYSE Tape A	<b>Order ID</b>	4 bytes, prepend 3 fields	8 bytes, prepend 3 fields
	<b>Trade ID</b>	4 bytes, prepend 3 fields	4 bytes, prepend 3 fields
NYSE Tapes B & C	<b>Order ID</b>	4 bytes, prepend 3 fields	<b>8 bytes</b>
	<b>Trade ID</b>	4 bytes, prepend 3 fields	4 bytes, prepend 3 fields

## 3.7 SYMBOL INDEXES

In all XDP feeds, symbol-specific referential data is published in a [Symbol Index Mapping Message \(Msg Type 3\)](#) at system startup. Symbol Index Mapping messages appear in each channel only for the symbols that appear in that channel.

Any client who misses this initial spin can request a refresh of Symbol Indexes by sending a [Symbol Index Mapping Request Message \(Msg Type 13\)](#) to the Request Server. The requested Symbol Index Mapping messages will be re-published over the Refresh channels.

The Symbol Index Mapping message includes the ASCII symbol in NYSE format along with a unique ID called a Symbol Index. Other symbol-specific messages such as Trade and BBO messages contain only the Symbol Index and no other referential data.

Symbol Indexes are the same for each symbol every day and the same across all Pillar-powered NYSE equity markets. Symbol Indexes for obsolete instruments are not re-used.

If more than one Symbol Index Mapping message is received for the same symbol within a trading day, the correspondence between the Symbol and the Symbol Index will not change, but other field values might. In this case, the latest field values override any earlier values, but do not apply retroactively.

## 4. Messages Sent by the Publisher

### 4.1 SYMBOL INDEX MAPPING MESSAGE (MSG TYPE 3)

This message is published over the real-time data channels at system startup or in the context of a refresh sequence after a Matching Engine or XDP Publisher failover. It provides referential data for a single specified symbol.

See [Symbol Indexes](#) for more information.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>Msg Size</b>	0	2	Binary	Size of the message: 44 bytes
<b>Msg Type</b>	2	2	Binary	The type of this message: 3 – Symbol Index Mapping Message
<b>SymbolIndex</b>	4	4	Binary	The unique ID of this symbol for all products within this market. This ID cannot be used to cross reference a security between markets.
<b>Symbol</b>	8	11	ASCII	Null-terminated ASCII symbol in <a href="#">NYSE Symbology</a> .
<b>Reserved</b>	19	1	Binary	This field is reserved for future use
<b>Market ID</b>	20	2	Binary	ID of the Originating Market: <ul style="list-style-type: none"> <li>1 - NYSE Cash</li> <li>3 – NYSE Arca Cash</li> <li>4 – NYSE Arca Options</li> <li>5 – NYSE Bonds</li> <li>6 – Global OTC</li> <li>8 – NYSE Amex Options</li> <li>9 - NYSE American Cash</li> </ul>
<b>System ID</b>	22	1	Binary	ID of the Originating matching engine server.
<b>Exchange Code</b>	23	1	ASCII	Exchange where the symbol is listed: <ul style="list-style-type: none"> <li>N – NYSE</li> <li>P – NYSE Arca</li> <li>Q – NASDAQ</li> <li>A – NYSE American</li> <li>B Global OTC Primary Symbols</li> <li>U – OTCBB symbols for Global OTC</li> <li>V – Other OTC symbols for Global OTC</li> <li>Z – BATS</li> </ul>
<b>PriceScaleCode</b>	24	1	Binary	Specifies placement of the decimal point in price fields for this security. See <a href="#">Prices</a> .
<b>Security Type</b>	25	1	ASCII	Type of Security used Arca and American exchanges: <ul style="list-style-type: none"> <li>A – ADR</li> <li>C - COMMON STOCK</li> <li>D – DEBENTURES</li> <li>E – ETF</li> <li>F – FOREIGN</li> <li>H – AMERICAN DEPOSITARY SHARES</li> <li>I – UNITS</li> <li>L – INDEX LINKED NOTES</li> <li>M - MISC/LIQUID TRUST</li> <li>O – ORDINARY SHARES</li> <li>P - PREFERRED STOCK</li> <li>R – RIGHTS</li> </ul>

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				<ul style="list-style-type: none"> <li>S - SHARES OF BENEFICIARY INTEREST</li> <li>T – TEST</li> <li>U – UNITS</li> <li>W – WARRANT</li> </ul> <p>Type of Security used by NYSE:</p> <ul style="list-style-type: none"> <li>A – COMMON STOCK</li> <li>B – PREFERRED STOCK</li> <li>C – WARRANT</li> <li>D – RIGHT</li> <li>E – CORPORATE BOND</li> <li>F – TREASURY BOND</li> <li>G – STRUCTURED PRODUCT</li> <li>H – ADR COMMON</li> <li>I – ADR PREFERRED</li> <li>J-ADR WARRANTS</li> <li>K – ADR RIGHTS</li> <li>L – ADR CORPORATE BOND</li> <li>M – NY REGISTERED SHARE</li> <li>N – GLOBAL REGISTERED SHARE</li> <li>O – INDEX</li> <li>P – FUND</li> <li>Q – BASKET</li> <li>R – UNIT</li> <li>S – LIQUIDATING TRUST</li> <li>U - UNKOWN</li> </ul>
<b>Lot Size</b>	26	2	Binary	Round lot size in shares.
<b>PrevClosePrice</b>	28	4	Binary	The previous day's closing price for this security.
<b>PrevCloseVolume</b>	32	4	Binary	The previous day's closing volume for the security.
<b>Price Resolution</b>	36	1	Binary	<ul style="list-style-type: none"> <li>0 - All Penny</li> <li>1 - Penny/Nickel</li> <li>5 - Nickel/Dime</li> </ul>
<b>Round Lot</b>	37	1	ASCII	<p>Round Lots Accepted:</p> <ul style="list-style-type: none"> <li>Y – Yes</li> <li>N – No</li> </ul>
<b>MPV</b>	38	2	Binary	The minimum increment for a trade price, in 100ths of a cent. Typically 1, or \$0.0001, but for some Tick Pilot stocks, can be 500, or \$0.05.
<b>Unit of Trade</b>	40	2	Binary	This field specifies the security Unit of Trade in shares. Valid values are 1, 10, 50 and 100
<b>Reserved</b>	42	2	Binary	Reserved for future use. Disregard any content.

## 4.2 SECURITY STATUS MESSAGE (MSG TYPE 34)

This message informs clients of changes in the status of a specific security, such as Trading Halts, Short Sale Restriction state changes, etc.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 46 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 34 – Security Status Message
<b>SourceTime</b>	4	4	Binary	The time when this msg was generated in the order book, in seconds since Jan 1, 1970 00:00:00 UTC.
<b>SourceTimeNS</b>	8	4	Binary	The nanosecond offset from the SourceTime
<b>SymbolIndex</b>	12	4	Binary	The unique ID of the symbol in the Symbol Index msg
<b>SymbolSeqNum</b>	16	4	Binary	The unique ID of this message in the sequence of messages published for this specific symbol.
<b>Security Status</b>	20	1	ASCII	<p>The new status that this security is transitioning to.</p> <p>The following are Halt Status Codes:</p> <ul style="list-style-type: none"> <li>3 - Opening Delay (NYSE/American only)</li> <li>4 - Trading Halt</li> <li>5 - Resume</li> <li>6 - No open/no resume (NYSE/American only)</li> </ul> <p>The following are Short Sale Restriction Codes (published for all symbols traded on this exchange):</p> <ul style="list-style-type: none"> <li>A – Short Sale Restriction Activated (Day 1)</li> <li>C – Short Sale Restriction Continued (Day 2)</li> <li>D - Short Sale Restriction Deactivated</li> </ul> <p>Market Session values :</p> <ul style="list-style-type: none"> <li>P – Pre-opening</li> <li>E – Early session (Arca and American only)</li> <li>O – Core session</li> <li>L – Late session (Arca and American only)</li> <li>X – Closed</li> </ul> <p>If this security is not halted at the time of a session change, the Halt Condition field = ~. If this security is halted on a session change, Halt Condition is non-~, and the security remains halted into the new session.</p> <p>The following values are the Price Indication values:</p> <ul style="list-style-type: none"> <li>T – T - Time</li> <li>I – Price Indication</li> <li>G – Pre-Opening Price Indication</li> <li>R – Rule 15 Indication.</li> </ul>
<b>Halt Condition</b>	21	1	ASCII	<ul style="list-style-type: none"> <li>~ - Security not delayed/halted</li> <li>D - News dissemination</li> <li>I - Order imbalance</li> <li>P - News pending</li> <li>M – LULD pause</li> <li>S - Related security (not used)</li> <li>X - Equipment changeover</li> <li>Z - No open/No resume</li> </ul>

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				Market Wide Circuit Breakers: <ul style="list-style-type: none"> <li>1 - Market Wide Circuit Breaker Halt Level 1</li> <li>2 - Market Wide Circuit Breaker Halt Level 2</li> <li>3 - Market Wide Circuit Breaker Halt Level 3</li> </ul>
<b>Reserved</b>	22	4	Binary	Future use. Any field content should be ignored.
<b>Price 1</b>	26	4	Binary	Default value is 0. <ul style="list-style-type: none"> <li>If securityStatus = A and this security is listed on this exchange, then this field is the SSR Triggering Trade Price</li> <li>If securityStatus = G, I, or R, then this field is the Indication Low Price.</li> </ul>
<b>Price 2</b>	30	4	Binary	Default value is 0 <ul style="list-style-type: none"> <li>If securityStatus = G, I, or R, then this field is the Indication High Price.</li> </ul>
<b>SSR Triggering Exchange ID</b>	34	1	ASCII	This field is only populated when securityStatus = A and this security is listed on this exchange. Otherwise it is defaulted to 0x20. <p>Valid values are:</p> <ul style="list-style-type: none"> <li>N – NYSE</li> <li>P – NYSE Arca</li> <li>Q – NASDAQ</li> <li>A – NYSE American</li> <li>B – NASDAQ OMX BX</li> <li>C – NSX</li> <li>D – FINRA</li> <li>I – ISE</li> <li>J – EDGA</li> <li>K – EDGX</li> <li>M – CHX</li> <li>S – CTS</li> <li>T – NASDAQ OMX</li> <li>V – IEX</li> <li>W – CBSX</li> <li>X – NASDAQ OMX PSX</li> <li>Y – BATS Y</li> <li>Z – BATS</li> </ul>
<b>SSR Triggering Volume</b>	35	4	Binary	Default value is 0. <p>This field is only populated when securityStatus = A and this security is listed on this exchange</p>
<b>Time</b>	39	4	Binary	Default value is 0. <p>Format : HHMMSSmmm (mmm = milliseconds)</p> <ul style="list-style-type: none"> <li>If securityStatus = A and this security is listed on this exchange, then this field is the SSR Trigger Time</li> <li>If securityStatus = T, then this field is the T-Time (mmm always = 000)</li> </ul>
<b>SSRState</b>	43	1	ASCII	The current SSR state, which this msg updates if the Security Status field contains an SSR Code. Valid values:

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
				<ul style="list-style-type: none"> <li>~ – No Short Sale Restriction in Effect</li> <li>E – Short Sale Restriction in Effect</li> </ul>
<b>MarketState</b>	44	1	ASCII	<p>The current Market State, which this msg updates if the Security Status field contains a Market State Code. Valid values:</p> <ul style="list-style-type: none"> <li>P – Pre-opening</li> <li>E – Early session (Arca and American only)</li> <li>O – Core session</li> <li>L – Late session (Arca and American only)</li> <li>X -- Closed</li> </ul>
<b>SessionState</b>	45	1	ASCII	Unused. Defaulted to 0x00.

### 4.3 SEQUENCE NUMBER RESET MESSAGE (MSG TYPE 1)

This message is sent to reset the Message Sequence Number at start of day, or in response to failures.

This message always appears in its own dedicated packet with a Sequence Number of 1 (the new, reset number). The packet Delivery Flag is normally 12, as on system startup, but during failover events it is set to 10.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 14 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 1 – Sequence Number Reset message
<b>SourceTime</b>	4	4	Binary	The time when this msg was generated in the order book, in seconds since Jan 1, 1970 00:00:00 UTC.
<b>SourceTimeNS</b>	8	4	Binary	The nanosecond offset from the SourceTime
<b>ProductID</b>	12	1	Binary	The unique ID for this NYSE feed listed in the feed's client specification.
<b>ChannelID</b>	13	1	Binary	The ID of the multicast channel over which the packet was sent.

### 4.4 SOURCE TIME REFERENCE MESSAGE (MSG TYPE 2)

For high-volume feeds such as XDP Integrated, this message is sent at the start of every second during periods of active data publication. Unlike some control messages, Source Time Reference messages can come in packets containing market data messages.

The client can concatenate the SourceTime field with the SourceTimeNS field in subsequent market data messages to get full 8-byte Matching Engine event timestamps. The contents of the ID field can be linked via the [Symbol Index Mapping Message \(Msg Type 3\)](#) to the applicable data messages.

See [Date and Time Conventions](#) for more information.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 16 bytes
<b>MsgType</b>	2	2	Binary	The type of message: 2 – Source Time Reference Message
<b>ID</b>	4	4	Binary	<b>For NYSE, Arca and American Integrated feeds:</b> ID of the originating Matching Engine partition to which this message applies. This usage will become standard across all products in future releases.  <b>For all other feeds:</b> Symbol Index of the symbol to which this message applies.
<b>SymbolSeqNum</b>	8	4	Binary	<b>For NYSE, Arca and American Integrated feeds:</b> Reserved for future use. Ignore any content. This usage will become standard across all products in future releases.  <b>For all other feeds:</b> The unique ID of this message in the sequence of messages published for this specific symbol.
<b>SourceTime</b>	12	4	Binary	The time when this msg was generated in the order book, in seconds since Jan 1, 1970 00:00:00 UTC.



#### 4.5 SYMBOL CLEAR MESSAGE (MSG TYPE 32)

In case of a failure and recovery of a Matching Engine or an XDP Publisher, the publisher may send a full state refresh for every symbol affected. This kind of unrequested refresh is preceded by a Symbol Clear message. The client should react to receipt of a Symbol Clear message by clearing all state information for the specified symbol in anticipation of receiving a full state refresh.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 20 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 32 – Symbol Clear
<b>SourceTime</b>	4	4	Binary	The time when this msg was generated in the order book, in seconds since Jan 1, 1970 00:00:00 UTC.
<b>SourceTimeNS</b>	8	4	Binary	The nanosecond offset from the SourceTime
<b>SymbolIndex</b>	12	4	Binary	The unique ID of the symbol in the Symbol Index msg
<b>NextSourceSeqNum</b>	16	4	Binary	The sequence number in the next message for this symbol

## 5. Messages Sent by Refresh and Retrans Servers Only

### 5.1 REFRESH HEADER (MSG TYPE 35)

The first message in each packet of refresh messages published over the Refresh multicast channels is of this type.

Valid values for the DeliveryFlag in the PacketHeader are:

- 17 – Only one packet in Refresh sequence
- 18 – Start of Refresh sequence
- 19 – Part of a Refresh sequence
- 20 – End of Refresh sequence
- 

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 16 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 35 – Refresh Header Message
<b>CurrentRefreshPkt</b>	4	2	Binary	The current refresh packet in the update
<b>TotalRefreshPkts</b>	6	2	Binary	The total number of refresh packets you should expect in the update
<b>LastSeqNum</b>	8	4	Binary	The last sequence number sent on the channel for any symbol. The refresh is the state of the order book as of this sequence number.
<b>LastSymbolSeqNum</b>	12	4	Binary	The last symbol sequence number sent for this symbol. The refresh is the symbol state of this symbol as of this symbol sequence number.

#### 5.1.1 Shortened Refresh Header

The first message in the first packet for a given symbol is a full 16-byte Refresh Header message.

Every other packet for the same symbol contains an 8-byte Refresh Header. The LastSeqNum and the LastSymbolSeqNum fields are removed so as not to send duplicate information in every packet.

#### 5.1.2 Refresh Example

Assuming this refresh of a single symbol requires three packets:

The first, second and third Packet structures look as follows:

PACKET HDR delivery = first	FULL REFRESH HDR	MESSAGE 1	MESSAGE 2	...	MESSAGE N
PACKET HDR delivery = part	SHORT REFRESH HDR	MESSAGE 1	MESSAGE 2	...	MESSAGE N
PACKET HDR delivery = last	SHORT REFRESH HDR	MESSAGE 1	MESSAGE 2	...	MESSAGE N

For a depth of book feed such as XDP Integrated or XDP ArcaBook, the sequence of refresh messages per symbol consists of the following message types:

1. Symbol Index Mapping Message (Msg Type 3)
2. Imbalance Message (Msg Type 105), if there is a current imbalance
3. Security Status Message (Msg Type 34)

4. Trade Session Change (Msg Type 33), indicates the current trading session (applies to Arca market only)
5. Add Order Refresh (Msg Type 106), repeated as needed to specify the book state for this symbol

### 5.1.3 Header Fields in the Refresh Channels

#### 5.1.3.1 Refresh response to a request for all Symbol Index Mapping messages

There are no Refresh Header messages

- First packet                      Delivery Flag =18 (START of refresh)
- Intermediate packets          Delivery Flag = 19 (PART of refresh)
- Last packet                      Delivery Flag = 20 (END of refresh)

#### 5.1.3.2 Refresh response to a request for a single Symbol Index Mapping message

There is no Refresh Header message.

- One packet is sent              Delivery Flag = 17 (ONE packet in the refresh)

#### 5.1.3.3 Refresh response to a request for a full refresh of all symbols

Each packet contains messages for a single symbol only.

- All packets for the first symbol      Delivery Flag =18 (START of refresh)
- All packets for intermediate symbols      Delivery Flag = 19 (PART of refresh)
- All packets for the last symbol          Delivery Flag = 20 (END of refresh)

The first message in each packet is a Refresh Header.

**For each symbol:**

- The currentRefreshPkt and totalRefreshPkts fields in the Refresh Header apply to this symbol only.
- The first packet contains a full Refresh Header (16 bytes). The LastSequenceNumber field contains the sequence number of the last message processed in this channel for any symbol. The LastSymbolSeqNum field contains the last Symbol Sequence Number processed for this symbol.
- All subsequent packets contain a short Refresh Header (8 bytes).

#### 5.1.3.4 Refresh response to a request for a full refresh of a single symbol

- If there are multiple packets in the response      Delivery Flags = 19 (PART of refresh)
- If there is only one packet in the response          Delivery Flag = 17 (ONE packet in the refresh sequence)

All packets begin with a Refresh Header message.

- The first packet contains a full Refresh Header (16 bytes).
- The first packet for a symbol contains a full Refresh Header (16 bytes). The LastSequenceNumber field contains the sequence number of the last message processed in this channel for any symbol. The LastSymbolSeqNum field contains the last Symbol Sequence Number processed for this symbol.
- All subsequent packets contain a short Refresh Header (8 bytes).

## 5.2 MESSAGE UNAVAILABLE MESSAGE (MSG TYPE 31)

This message will be sent over the Retransmission multicast channels to inform clients of unavailability of a range of messages (or part of a range) for which they may have requested a retransmission.

For any packet containing a Message Unavailable message, the Packet Header Delivery Flag will be set to 21.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 14 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 31 – Message Unavailable
<b>BeginSeqNum</b>	4	4	Binary	The beginning sequence number of the unavailable range of messages.
<b>EndSeqNum</b>	8	4	Binary	The ending sequence number of the unavailable range of messages.
<b>ProductID</b>	12	1	Binary	The unique ID of the feed for which the retransmission was requested (listed in the feed's client specification).
<b>ChannelID</b>	13	1	Binary	The ID of the multicast channel for which the retransmission was requested.

## 6. Messages Sent by the Client to the Request Server

---

### 6.1 RETRANSMISSION REQUEST MESSAGE (MSG TYPE 10)

Clients who have experienced a sequence number gap and need a retransmission of the missed messages should send a Retransmission Request message via TCP to the Request Controller. A Request Response message will be sent over the TCP connection back to the client, and if the request was valid, the requested message(s) will be re-published over the relevant Retransmission multicast channel.

The retransmitted message(s) will have the same message format and content as the original messages that were missed.

Retransmission Requests should be sent in a packet whose Packet Header Delivery Flag is set to 11, and which contains a valid sequence number.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 24 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 10 – Retransmission Request message
<b>BeginSeqNum</b>	4	4	Binary	The beginning sequence number of the range of messages to be retransmitted.
<b>EndSeqNum</b>	8	4	Binary	The end sequence number of the range of messages to be retransmitted.
<b>SourceID</b>	12	10	ASCII	The ID of the client requesting this retransmission . This field is up to 9 characters, null terminated.
<b>ProductID</b>	22	1	Binary	The unique ID of the feed for which a retransmission is requested (listed in the feed's client specification).
<b>ChannelID</b>	23	1	Binary	The ID of the multicast channel on which the gap occurred.

## 6.2 REFRESH REQUEST MESSAGE (MSG TYPE 15)

Clients who have experienced a failure and need a refresh of the state of one or all symbols in a specific channel should send a Retransmission Request message via TCP to the Request Controller. A Request Response message will be sent over the TCP connection back to the client, and if the request was valid, the requested message(s) will be published over the relevant Refresh multicast channel.

Retransmission Requests should be sent in a packet whose Packet Header Delivery Flag is set to 11, and which contains a valid sequence number.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 20 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 15 – Refresh Request Message
<b>SymbolIndex</b>	4	4	Binary	The ID (from the Symbol Index msg) of the symbol for which a refresh is requested. To request a refresh for all symbols in the channel, set this field to 0.
<b>SourceID</b>	8	10	ASCII	The ID of the client requesting this refresh. This field is up to 9 characters, null terminated.
<b>ProductID</b>	18	1	Binary	The unique ID of the feed for which the refresh is requested (listed in the feed's client specification).
<b>ChannelID</b>	19	1	Binary	The ID of the multicast channel for which the refresh is requested.

### 6.3 SYMBOL INDEX MAPPING REQUEST MESSAGE (MSG TYPE 13)

This message is sent by clients via TCP/IP requesting the Symbol Index Mapping messages for one or all symbols in a specified channel.

The Symbol Index Mapping Request messages should be sent in a packet whose Packet Header Delivery Flag is set to 11, and which contains a valid sequence number.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 21 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 13 – Symbol Index Mapping Request Message
<b>SymbolIndex</b>	4	4	Binary	The ID (from the Symbol Index msg) of the symbol for which a refresh is requested.  To request a refresh for all symbols in the specified channel, set this field to 0.
<b>SourceID</b>	8	10	ASCII	The ID of the client requesting this symbol refresh. This field is up to 9 characters, null terminated.
<b>ProductID</b>	18	1	Binary	The unique ID of the feed for which the refresh is requested (listed in the feed's client specification).
<b>ChannelID</b>	19	1	Binary	The ID of the multicast channel for which the refresh is requested.
<b>RetransmitMethod</b>	20	1	Binary	The delivery method for the requested symbol index mapping information. Valid values: <ul style="list-style-type: none"> <li>0 – deliver via UDP</li> </ul>

### 6.4 HEARTBEAT RESPONSE MESSAGE (MSG TYPE 12)

Clients who remain connected to the Retransmission Server intraday must respond to a Heartbeat with a Heartbeat Response message within 5 seconds. If no timely client response is received, the connection will be closed.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 14 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message: 12 – Heartbeat Response message
<b>SourceID</b>	4	10	ASCII	The ID of the connected client . This field is up to 9 characters, null terminated.

## 7. Messages Sent by Request Server

### 7.1 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.

FIELD NAME	OFFSET	SIZE	FORMAT	DESCRIPTION
<b>MsgSize</b>	0	2	Binary	Size of the message: 29 Bytes
<b>MsgType</b>	2	2	Binary	The type of this message:  11 – Request Response Message
<b>RequestSeqNum</b>	4	4	Binary	The sequence number of the request message sent by the client. This can be used by the client to couple this response with the original request message.
<b>BeginSeqNum</b>	8	4	Binary	For Retrans Request responses, the beginning sequence number of the requested retransmission range.  For responses to Refresh or Symbol Mapping Requests, 0.
<b>EndSeqNum</b>	12	4	Binary	For Retrans Request responses, the ending sequence number of the requested retransmission range.  For responses to Refresh or Symbol Mapping Requests, 0.
<b>SourceID</b>	16	10	ASCII	The ID of the client making the request. This field is up to 9 characters, null terminated.
<b>ProductID</b>	26	1	Binary	The unique ID of the feed for which the request was made (listed in the feed's client specification).
<b>ChannelID</b>	27	1	Binary	The ID of the multicast channel for which the request was made.
<b>Status</b>	28	1	ASCII	The reason why the request was rejected. Valid values: <ul style="list-style-type: none"> <li>0 – Message was accepted</li> <li>1 – Rejected due to an Invalid Source ID</li> <li>2 – Rejected due to invalid sequence range</li> <li>3 – Rejected due to maximum sequence range (see threshold limits)</li> <li>4 – Rejected due to maximum request in a day</li> <li>5 – Rejected due to maximum number of refresh requests in a day</li> <li>6 – Rejected. Request message SeqNum TTL (Time to live) is too old. Use refresh to recover current state if necessary.</li> <li>7 – Rejected due to an Invalid Channel ID</li> <li>8 – Rejected due to an Invalid Product ID</li> <li>9 – Rejected due to: 1) Invalid MsgType, or 2) Mismatch between MsgType and MsgSize</li> </ul>



## 8. Error Handling and the Request Server

---

### 8.1 HANDLING SEQUENCE NUMBER GAPS

Since multicast is an unreliable protocol, messages can be dropped. For this reason, clients are advised to process both lines in a channel. If a gap occurs on one line, the gap can be filled immediately from the other.

If a gap occurs on both lines simultaneously, the client can send a [Retransmission Request Message \(Msg Type 10\)](#) via TCP to the Request Server. The Retransmission Request contains a unique client ID called a Source ID, along with the Product and Channel IDs and the sequence number range of the missing messages.

On receipt of a Retransmission Request message, the Request Server will send back a [Request Response Message \(Msg Type 11\)](#). If any of the fields of the the Retransmission Request contained malformed or meaningless information, the request is rejected. If the request is accepted, the Retransmission Server will re-send the requested messages via multicast over the Retransmission channels.

If the request is rejected for exceeding a predefined system limit, the client will be prevented from making any further requests. See [Request Quotas](#). If further requests are required, please contact NYSE.

#### 8.1.1 Retransmission Format

Retransmitted messages have the same message format and content as the originally published messages (including the [Sequence Numbers](#), but they may be packetized differently for best efficiency.

Packets of retransmitted messages have special Delivery Flag values in the Packet Header:

- 13 – Only one packet in retransmission sequence
- 15 – Part of a retransmission sequence

### 8.2 RECOVERING FROM CLIENT LATE STARTS OR INTRADAY FAILURES

If a client process experiences a late start or an intraday failure, the client will usually want to receive snapshots of the current market state for each symbol before resuming processing of real-time data. To do this, the client requests a refresh from the Refresh Server.

Specifically, a late-starting or recovering client should

1. Subscribe to the Publisher multicast channels. Any messages received should be cached but not processed until all refresh information is processed.
2. Connect to the Request Server. This connection should be maintained all day.
3. Subscribe to the Refresh multicast channels
4. Send a [Refresh Request Message \(Msg Type 15\)](#) to the Request Server

The Refresh Request contains

1. A unique client ID called a Source ID
2. Product and Channel IDs
3. A Symbol Index, specifying a particular symbol to be refreshed or else if 0, specifying all symbols.

On receipt of a Refresh Request message, the Request Server will send back a [Request Response Message \(Msg Type 11\)](#). If any of the fields of the the Refresh Request contained malformed or meaningless information, the request is rejected. If the request is rejected for exceeding a predefined system limit, the client will be prevented from making any further requests. See [Request Quotas](#). If further requests are required, please contact NYSE.

If the request is accepted, the Refresh Server will send the snapshot message(s) over the specific Refresh channel. All these messages should be used to rebuild the current state of the order book. Once all refresh messages are processed, messages from the Publisher can now be processed. Note that any messages received whose sequence numbers are lower than the LastSequenceNumber indicated in the refresh sequence should be discarded.

#### 8.2.1 Refresh Format

Each refresh packet begins with a Packet Header, followed by a [Refresh Header \(Msg Type 35\)](#).

The Packet Header for a refresh packet has special Delivery Flag values:

- 17 – Only one packet in Refresh sequence
- 18 – Start of Refresh sequence

- 19 – Part of a Refresh sequence
- 20 – End of Refresh sequence

These Delivery Flag values are used 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.

The Refresh Header identifies the position of the current packet in this sequence of Refresh packets, along with the total number packets in this sequence. By use of the Delivery Flag and the packet sequence information in the Refresh Header, the client can know when the last packet of the refresh sequence has been received.

No dedicated retransmission service is available for the Refresh Server; if message loss is detected in a refresh channel, clients should submit another refresh request.

## 8.3 REFRESHING SYMBOL INFORMATION

At system startup, each channel publishes a [Symbol Index Mapping Message \(Msg Type 3\)](#) for each symbol published on this channel.

If a client process misses the initial spin of symbol information for whatever reason, he may wish to receive a refresh of some or all Symbol Index Mapping messages before resuming processing of real-time data. To do this, the client should follow the procedure described in [Recovering from Client Late Starts or Intraday Failures](#), but send a [Symbol Index Mapping Request Message \(Msg Type 13\)](#) to the Request Server instead of a Refresh Request Message.

### 8.3.1 Symbol Index Mapping Refresh Format

Requested Symbol Index Mapping messages are published by the Refresh Server with the same Packet Header Delivery Flags used for Refresh publications. Refresh Headers are not used in Symbol Index Mapping refreshes.

## 8.4 REQUEST SERVER

It is possible to connect to the Request Server only as needed, disconnecting after each request, but it is recommended that you remain connected to the Request Server for the entire trading day.

The Request Service is subject to IP Table filtering in order to safeguard against events similar to denial-of-service attacks. The filtering prevents any client from making further connections to the RCF after the client has connected a truly excessive number of times.

Once a client establishes a TCP/IP connection, the Request Server will send a heartbeat to the client approximately every 60 seconds. Clients must respond to with a Heartbeat Response message within 5 seconds, otherwise the Request Server will assume the client or the network has failed and close the connection.

### 8.4.1 Request Queuing

Clients may send several requests at the same time with the same Source ID. There is no need to wait for one request to be fulfilled before requesting another one.

Responses to all requests are published in the order in which they are received, although overlapping requests may be de-duplicated for efficiency.

### 8.4.2 Request Quotas

The table below summarizes the retransmission/refresh request limitations that are enforced by the Request Server. The numbers represent thresholds per channel.

Any client who has been blocked by exceeding these quotas can connect to the backup Request Server. This will reset all quotas. This only works once.

CAPABILITY	DESCRIPTION
<b>Prevention of invalid subscribers</b>	Incoming requests from subscribers that are not in the enabled subscriber's source ID list will not be honored.
<b>Limitation of number of packets per Retrans Request</b>	Retransmission requests for more than 1,000 messages will not be honored.
<b>Limitation of timeliness of Retrans Requests</b>	Retransmission requests for sequence numbers more than 1,000,000 lower than the current sequence number will not be honored.
<b>Limitation of number of Retrans Requests in a day</b>	Retransmission requests from a client who has already made 5,000 retransmission requests today will not be honored, and the client will be blocked from making retransmission requests for the remainder of the day.
<b>Limitation of number of Refresh Requests in a day</b>	Refresh requests from a client who has already made 5,000 refresh requests today will not be honored.

## 9. Operational Information

### 9.1 SYSTEM BEHAVIOR ON START AND RESTART

At system startup or at start of system recovery following a failure, XDP feeds send the following messages over each channel:

- Multicast priming from the primary Publisher's source IPs: a series of Heartbeat packets for several seconds with the Delivery Flag set to 1 and sequence number set to 1, the next expected sequence number.
- [Sequence Number Reset Message \(Msg Type 1\)](#), the sequence number is 1 and the packet DeliveryFlag is 12
- A full spin of [Symbol Index Mapping Messages \(Msg Type 3\)](#), for securities published on this channel

### 9.2 XDP PUBLISHER FAILOVER

When failing over to the backup XDP Publisher, the following refresh information is published.

Note: During the failover refresh, DeliveryFlag fields for all Packet Headers except Heartbeats are set to 10.

1. Multicast priming from the backup Publisher's source IPs: a series of Heartbeat packets for several seconds with the Delivery Flag set to 1 and sequence number set to 1, the next expected sequence number.
2. A [Sequence Number Reset Message \(Msg Type 1\)](#) is sent in its own packet
3. For each symbol, the following are published,
  - [Symbol Index Mapping Messages \(Msg Type 3\)](#)
  - [Symbol Clear Message \(Msg Type 32\)](#)
  - The last [Security Status Message \(Msg Type 34\)](#)
  - The last [Trading Session Change Message \(Msg Type 33\)](#) (applies to the Arca market only)
  - All required refresh messages
  - The last [Source Time Reference Message \(Msg Type 2\)](#)

Once all symbols have been refreshed, Packet Header DeliveryFlag fields return to the normal 11.

### 9.3 DISASTER RECOVERY SITE

All XDP feeds are published out of the NYSE Mahwah data center. In case of catastrophic failure in Mahwah, all affected systems including XDP feeds will be coldstarted at the Cermak Disaster Recovery site in Chicago. The Cermak configuration of channels and multicast groups is identical to Mahwah for all feeds, but of course the source IPs are different. The initial publication sequence is as described in [System Behavior on Start and Restart](#).

### 9.4 XDP PRODUCTION HOURS

EVENT	NYSE/ AMERICAN TIME (ET)	NYSE ARCATIME (ET)
Sequence Number Reset	12:30am	12:30am
Symbol Mapping	12:30am	12:30am
Early Open Auction		4:00am
Core Open Auction and Open	9:30am	9:30am
Closing Auction and Close	4:00pm	4:00pm
End of Late Session		8:00pm

### 9.5 XDP TEST HOURS

UAT testing hours are approximately 1:00 AM until 8:15 PM.

## 10. Symbol Index Mapping File

For customers that would prefer to download the symbol index mapping from an FTP server, a file containing 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.txt>

XML format: <ftp://ftp.nyxdata.com/NYSESymbolMapping/NYSESymbolMapping.xml>

### ■ NYSE American

Pipe-delimited: <ftp://ftp.nyxdata.com/AmericanSymbolMapping/AmericanSymbolMapping.txt>

XML format: <ftp://ftp.nyxdata.com/AmericanSymbolMapping/AmericanSymbolMapping.XML>

### ■ NYSE ARCA

Pipe-delimited: <ftp://ftp.nyxdata.com/ARCASymbolMapping/ARCASymbolMapping.txt>

XML format: <ftp://ftp.nyxdata.com/ARCASymbolMapping/ARCASymbolMapping.XML>

### Symbol Index Mapping File Format

FIELD NAME	FORMAT	DESCRIPTION
<b>Symbol</b>	ASCII	The full symbol in <a href="#">NYSE Symbolology</a> .
<b>CQS Symbol</b>	ASCII	The full symbol in CTS and UTP line format. See <a href="#">NYSE Symbolology</a> .
<b>SymbolIndex</b>	Numeric	The unique ID for this symbol. See <a href="#">Symbol Indexes</a> . This ID is unique for products within each market. It cannot be used to cross reference a security between markets.
<b>NYSE Market</b>	Character	The market within the NYSE Group where this symbol is traded: <ul style="list-style-type: none"> <li>■ N – NYSE</li> <li>■ P – NYSE Arca</li> <li>■ A – NYSE American</li> </ul>
<b>Listed Market</b>	Character	The market where this symbol is listed: <ul style="list-style-type: none"> <li>■ A – NYSE American</li> <li>■ B – Global OTC Primary Symbols</li> <li>■ N – NYSE</li> <li>■ P – NYSE Arca</li> <li>■ Q – NASDAQ</li> <li>■ T – NASDAQ stock traded on NYSE Arca</li> <li>■ U – OTCBB symbol for Global OTC</li> <li>■ V – Other OTC symbols for Global OTC</li> <li>■ Z - BATS</li> </ul>
<b>TickerDesignation</b>	Character	The SIP tape on which this symbol is published: <ul style="list-style-type: none"> <li>■ A – CTA Tape A</li> <li>■ B – CTA Tape B</li> <li>■ Q – CTA Tape C</li> </ul>
<b>UOT</b>	Numeric	The number of shares in a round lot.
<b>PriceScaleCode</b>	Numeric	A code used to place the decimal point in all price fields for this symbol. See <a href="#">section 3.5</a> for details on price handling.
<b>SystemID</b>	Numeric	The ID of the matching engine instance that handles this symbol.
<b>Bloomberg BSID</b>	Empty	Reserved field. No character between the delimiting pipe characters.
<b>Bloomberg Global ID</b>	Empty	Reserved field. No character between the delimiting pipe characters.

*NOTE: The pipe delimited (.txt) version of this file has an additional pipe character at the end of the Bloomberg Global ID field, so every record in the file has three pipe characters after the System ID field.*