Cboe Application Programming Interface

Cboe Streaming Market (CSM)

Opening Auction Feed Specifications

Version 1.0

June 13, 2018

Change Notices

The following change notices are provided to assist users in determining the impact of changes to their applications.

Date	Version	Description of Change
6/13/2018	1.0	Removed RUT as a symbol to be provided on the feed
6/1/2018	1.0	New document

Support and Questions Regarding This Document

Questions regarding this document can be directed to the Cboe Exchange at 312.786.7300 or via e-mail: api@cboe.com. The latest version of this document can be found on the Cboe web site at http://systems.cboe.com.

Table of Contents

Reference Tables

Examples of Data Transmissions

1 Introduction

Cboe Streaming Market (CSM) Opening Auction feed publishes market data over the Cboe Financial Network (CFN) using the message format defined in this document. Data is transmitted using the IP Multicast network protocol. To connect to the CFN network, refer to the CFN Network Specification document on the Cboe API website at https://systems.cboe.com/Auth/CFN.aspx.

1.1 System Overview

CSM Opening Auction feed distributes security definition, current market (top of book), and expected opening price (EOP) data. A **feed** is a set of one or more data channels and one security definition channel. Several CSM feeds are available, grouped by Cboe exchange and by product category or complexity.

A **channel** consists of two Multicast groups in a primary/secondary architecture where the data is duplicated on the two Multicast groups for redundancy.

Communication is one way only with no mechanism for retransmission, but there are mechanisms for recovery. Messages are encoded using a mix of ASCII characters and binary data in the format defined in this document. Cboe also distributes templates that describe the message structure of CSM opening auction feeds via the API website at https://systems.cboe.com/Auth/CFN.aspx. These templates may be used for decoding messages. The templates and message structures defined in this document will be static, and will not change over the course of the trading day, nor even in most software releases. Clients can expect sufficient advance notice about any changes to these templates or message structures.

1.2 CSM Opening Auction Feed Offered

A CSM Opening Auction feed is a set of related multicast groups over which current market, EOP and security definition information is transmitted for a particular Cboe exchange and product category.

The Cboe Streaming Market feeds offered are:

Exchange Feed And Product Category
Cboe Non-Strategy Options

1.2.1 For Customers of CSM Current Market: Security Definition Format is Identical

The *CSM Opening Auction* feed shares a common security definition format with the *CSM Current Market* feed, which is the top of book feed offered separately. Each type of feed transmits a copy of the security definitions to its own security definition channel so that the security definition multicast groups are logically grouped with the data channels' multicast groups for the feed and so that network multicast masking and routing is less difficult to manage. The two types of feeds are purchased separately and operate from different systems.

If you are an existing customer of the CSM Current Market feed and you also subscribe to this auction feed**,** you are required to read the security definition channel from both feeds. The format of the security definition message is identical on both feeds and will publish the same data. However, the opening auction feed will stop transmitting when the product is opened. The timing and ordering of

messages and the sequence numbers on the security definition channels will differ because the systems operate independently, but the content will be the same set of products on both.

Customers are free to leverage re-use of the same code to parse and process the security definition from either feed and they may choose to read only one copy of the security definitions. However, there is no penalty for reading both copies of the security definition should you choose to do so.

In the event of a future change to the format of the security definition message, the format will change on both CSM Current Market and CSM Opening Auction at the same time and will be rolled out to customers concurrently so the format is consistent on both types of feeds.

1.3 Hours of Operation

The system is expected to be available from 6:00 AM Central Time (CT) until all products are opened. Securities are expected to transition to Pre-open around 6:00 AM. Security Definition Messages, Current Market Refresh Messages and Current Market Update Messages will be published during this time.

2 Data Feed and Message Overview

2.1 Data Feed Overview

A feed consists of one or more **data** channels and a **security definition** channel. The number of data channels differs for various feeds based on capacity requirements. Low volume feeds may have only one data channel, while high volume feeds have multiple data channels. Each feed has a single security definition channel.

General characteristics of a feed include the following:

- Each feed contains market data only for products from a particular exchange and product category as defined in *CSM Opening Auction* Feed Offered.
- Each channel is duplicated and sent to 2 different multicast groups and ports over 2 networks in a primary / secondary configuration. Data sent to the primary and secondary multicast groups for each channel is identical.
- There are no retransmissions. If a recipient is late to join, or if packets are dropped, **one complete cycle** of Market Data Refresh messages must be processed to insure accuracy of all Current Market data.
- A sequence number is sent for each message. This can be used to identify missed messages over a particular channel. Each channel has its own sequence number, so for example, channel 1 of a feed may be at sequence numbers 1, 2, 3, etc. while channel 2 may be at sequence numbers of 24, 25, 26 etc. Tracking of sequence numbers must be done for each channel.
- Messages are placed into blocks (packets) for delivery which allows for multiple messages per block. The maximum block size is 1000 bytes.
- The message structures, field names and field values are based as much as possible on the FIX 5.0 SP2 standard. However, messages are encoded using a proprietary ASCII + binary format, and FIX tags are not transmitted in the data stream. The FIX format was used for the convenience of those familiar with the FIX standard, so messages are defined in terms of FIX field names and FIX tags. Some user-defined fields were necessary for those fields not in the FIX specification, and some modifications to standard FIX fields are implemented for efficiency reasons.

2.2 Message Overview

The following types of messages are transmitted over a feed:

2.2.1 Security Definitions

Security definitions describe an exchange's products by name and trading parameters and associates those products with a class key and security ID that is unique to each product. To conserve bandwidth, market data messages sent over the data channels are transmitted with only class key and security ID information, so the security definition message is used to relate the class key and security ID from those messages to detailed product information.

Security definition messages are sent for all products on a dedicated channel in a cycle that repeats approximately every two minutes. Security definitions are used to establish the initial set of products

traded on a feed, and to allow recipients to join the feed at any time and get initialized with all of the exchange's products it will receive over the data channels for that feed.

For new products added intraday, a security definition is sent to the data channel for that product preceding the first update for that product. Thereafter, newly added products are added to the next cycle of the security definition channel. There is no guarantee regarding the order in which a security definition is sent to the security definition channel versus the first update for a security sent to the data channel. For this reason, you may see an update for a product on the data channel before the first security definition is sent on the security definition channel. However, if the product is new and had not existed before the first update, you will see a security definition over the data channel preceding the first update, but this applies only for new products added intraday.

2.2.2 Market Data Refresh

Market Data Refresh messages contain a snapshot of the market data for a product, which includes current market, last sale price, size of last sale, total volume, previous close price, open price, high price, and low price. They are sent to the data channels of a feed.

Market Data Refresh messages are sent for all products that have market data in a cycle that repeats approximately every two minutes (similar to security definitions, but sent over data channels). The Market Data Refresh is used establish the current state of market data at start-up, and to recover the state of market data when messages are dropped.

Market Data Refresh messages contain two sequence numbers: The *msgSeqNumber* in the standard header is used to monitor for channel-level data loss. This sequence number will be set to 1 at the start of each trading day's session and continue to increase as long as no other errors in the system have occurred. Additionally, there is an *ApplSeqNum* in the refresh message body. This sequence number is used to determine when a refresh cycle begins.

When a refresh cycle starts, the *ApplSeqNum* sequence number associated with the refresh will begin again at 1 for each data channel. This can be used to determine that a new refresh cycle has just begun.

2.2.3 Current Market Update

Current Market Update messages contain the current market (top of book) and product state for a product. They are sent when the top of book changes or when the product state for a product changes.

2.2.4 EOP (Expected Opening Price)

EOP messages contains expected opening price and size information for a product. They are sent periodically when the market is in pre-opening or opening rotation state.

2.2.5 Heartbeats

Heartbeat / line integrity messages are transmitted every five seconds to every channel. These messages may be used to determine that a channel is working during times when market data is not transmitted on the feed (such as pre-market or post-market times).

2.3 Message Routing

Messages are routed to specific channels of a feed based on the type of message and its content.

2.3.1 Security Definitions

- Security definitions are sent to a dedicated security definition channel for each feed, with one exception, (see below).
- Security definition messages for all products are sent to the security definition channel in a repeating cycle of approximately 2 minutes duration. When one cycle completes the next begins immediately. A security definition for a particular product should therefore repeat approximately every 2 minutes.
- Security definitions for *existing* products are sent only to the security definition channel for the feed.
- When a *new* product is added intraday, a security definition will be sent once to the *data* channel for the newly added security, just before the first update for that security is sent to the data channel.

  This is done to allow recipients to stop reading the security definition channel after they initially synchronize with a complete cycle and to insure that the security definition for a new product is seen before the first update for the product.
- After a new product is added intraday and its initial security definition is sent once to the data channel for the product, the product is added to the security definition repeating cycle. Subsequent security definitions for the product will appear only on the security definition channel.
- A field in the security definition, *TargetLocationID* indicates over which data channel the market data for a product will be transmitted. This is discussed in more detail in the definition section of this document.

2.3.2 Market Data Refresh, EOP and Current Market Update Messages

- Messages are routed to a single data channel within a feed by **product class key**. A product class **is a numeric identifier that is related to, but not the same as the underlying security** for a product. One channel is chosen for a particular product class key which will not change intra-day.

  For example, all IBM products (options for options feeds, equities for equities feeds, etc.) will be sent to one channel. Corporate actions for IBM such as IBM1 share the same product class, so its products will be sent to the same channel.

  By contrast, SPX (regular) and SPXW (weekly) options share a common underlying (the S&P 500), but are different product classes and may be on different data channels.
- Across days (I.E. overnight), the data channel chosen for a product class may change due to load-balancing considerations or system improvements or enhancements to the distribution method.
- All market data messages related to a security are transmitted over the same channel to preserve correct sequencing.
- Data for symbols SPX and SPXW will initially be provided on the feed. This list of symbols can be modified in the future without notice.

2.3.3 Heartbeats

Heartbeat / line integrity messages are transmitted every five seconds to every channel including data and security definition channels.

3 Message Templates, Field Data Types and Data Encoding

3.1 Message Templates

Messages for the CSM feeds are described in this document in tabular text format and as an XML template. Templates define the content and characteristics of the messages to be encoded or decoded.

XML templates that describe the structure of messages are available to recipients on the Cboe web site at https://systems.cboe.com/Auth/CFN.aspx. Firms are encouraged to write software capable of using the XML templates to decode data from a CSM feed.

XML templates are used to specify the structure, data types, field names, and FIX tags of a message:

1 - Example of an XML based template

`<template name="MDIncRefresh" id = "0">`	Start of new template
`<string name="MessageType" id="35" byteLength="1" value="X" />`	Defines a String Data Type Field, The id which represents the fix Tag is not transferred on the wire.
`<uInt32 name="MsgSeqNum" id="34" />`	
`<sequence name="MDEntries">`	Defines the start of a repeating group
`<length name="NoMDEntries" id ="268"/>`	Length of repeating group
`<string name="Symbol" id="55"/>`	
`<uInt32 name="Quantity" id="53 />`	
`</sequence>`	End repeating group
`</template>`	End template

3.2 Template IDs

Each message structure is defined with a unique template ID. A template ID is a binary integer value stored as the first byte of every message that identifies the structure of the message.

Template IDs are assigned from a common pool for CSM Current Market, CSM Opening Auction and the CSM Level 2 Book Depth feeds, so there may be gaps in the numbering when either specification is updated. Numbers are assigned sequentially as they are needed for new message structures and old template ids are retired when new versions of the feed are launched.

The template ids for CSM Auction feeds are as follows:

2 - Templates and their IDs

Template Name	Template ID
Current Market Update	12
Security Definition	13
EOP	15
Heartbeat	16
Market Data Refresh	20

3.3 Field Data Types and Data Encoding

Fields defined in messages for CSM feeds will have one of the following data types and methods of encoding:

3.3.1 STRING Field

Strings are ASCII character arrays or single-byte characters. There are two types of string fields which are encoded differently:

Single Byte String

If the *byteLength* attribute of a string field is defined as "1", for example:
<string name="MessageType" id="35" byteLength="1"/>
it is a single byte string, which is encoded with a single ASCII character.

Character Array String

If the *byteLength* attribute is not defined for the field, for example:
<string name="Symbol" id="55"/>
The string is variable length and is encoded with an unsigned binary byte indicating the length of string, followed by the string's characters:

Length of String (1 byte)	String Characters

For example, the string "IBM" would be encoded as:
Binary value 3, then the characters "IBM".

3.3.2 INTEGER and LENGTH Fields

Integer and Length fields are big endian binary encodings of numeric values. The *byteLength* attribute in a template field definition can act as a modifier to restrict the number of bytes used to encode the integer value. Unsigned integers are encoded as zero or positive-only values. The top-most bit is part

of the magnitude of the value. Signed integers are encoded as two's-complement binary values with the top-most bit as the sign bit. Length fields are unsigned integer values used to indicate the length of a Sequence field.

There are several types of integer or length fields:

Integer Field Type	**Byte Length**	**Encoding**	**Example**
uInt32	1	8 bit unsigned integer	<uInt32 name="MDPriceLevel" id="1023" byteLength="1"/>
uInt32	4 or omitted	32 bit unsigned integer	<uInt32 name="MDEntrySize" id="271" byteLength="4"/>
uInt64	8 or omitted	64 bit unsigned integer	<uInt64 name="SendingTime" id="52"/>
int32	1	8 bit signed integer	<int32 name="MDPriceLevel" id="1023" byteLength="1"/>
int32	4 or omitted	32 bit signed integer	<int32 name="MDEntrySize" id="271"/>
int64	8 or omitted	64 bit signed integer	<int64 name="SendingTime" id="52" byteLength="8">
length	1	8 bit unsigned integer	<length name="NoMDEntries" id="268" byteLength="1"/>

The table below shows the min and max values for different integer data types.

Type	**Min**	**Max**
uInt32 with byteLength="1"	0	255
uInt32	0	4,294,967,295
uInt64	0	18,446,744,073,709,551,615
int32 with byteLength="1"	-128	127
int32	-2,147,483,648	2,147,483,647
int64	- 9,223,372,036,854,775,808	9,223,372,036,854,775,807
length with byteLength="1"	0	255

3.3.3 DECIMAL Field

A decimal field is used to represent a floating point number as exponent and mantissa. The exponent is a signed 8 bit integer used to express precision and the mantissa is a signed 32 bit integer used to express the value. The numerical value is obtained by multiplying the mantissa with the base-10 power of the exponent expressed as: number = mantissa * 10^{exp}. The exponent and mantissa is decoded as a single, composite field.

Decimal fields are 5 bytes in length. The first byte is the exponent and the remaining 4 bytes are the mantissa. For example, the number 0.90 is encoded as FE0000005A.
FE (exponent) == -2, 0000005A (mantissa) == 90, value == 90 * 10^{-2} == 90 * 0.01 == 0.90

3.3.3.1 NO PRICE Decimal Value

In certain messages there may be decimal price fields that are always present in the message, but for which the value of the price is unknown or not applicable. To represent a price that is unknown or not applicable, a special decimal price value is used. That value is:

Exponent: -9
Mantissa: -2147483648 (the minimum 32 bit signed integer value)
The hex value of the Exponent + Mantissa is: F780000000
The price value decodes to the decimal value: -2.147483648

3.3.4 SEQUENCE Field

A sequence is a repeating group of fields. The group of fields contained in a sequence can be a simple type as described above, or another nested sequence. A length field encoded as an unsigned int immediately precedes the fields contained in the sequence. The length field is defined in a template with a special attribute of "<length", and it can be modified with *byteLength* attribute. If byteLength="1", the encoded length field is a single 8-bit unsigned byte. All sequences transmitted to CSM feeds use a single 8-bit unsigned length and can be no longer than 255 entries.

Sequences are encoded as follows:

Length field	Group#1 Field #1	Group#1 Field#2	…	Group#1 Field#N	Group#2 Field#1	Group#2 Field#2	…	Group#2 Field#N	…

Note that a sequence may contain a nested sequence, as in the following template example:

```
<sequence   name="MDEntries">
    <length     name="NoMDEntries"          id="268"    byteLength="1"/>
    <string     name="MDEntryType"          id="269"    byteLength="1"/>
    <decimal    name="MDEntryPx"            id="270"    byteLength="5"/>
    <sequence   name="MDVolumeEntries">
        <length     name="NoMDVolumeEntries"    id="21000"  byteLength="1"/>
        <uInt32     name="MDVolumeType"         id="21001"  byteLength="1"/>
        <uInt32     name="MDEntrySize"          id="271"    byteLength="4"/>
    </sequence>
</sequence>
```

Here is an example of the above sequence with 2 MDEntries elements each with different lengths of nested MDVolumeEntries.

Field	**Length in Bytes**	**Value**	**Comments**
NoMDEntries	1	2	Length of MDEntries Sequence
MDEntryType	1	'0'	Bid entry
MDEntryPx	5	FE0000000A	FE == -2 exponent, 0000000A == 10 mantissa, value == 0.10
NoMDVolume Entries	1	2	Bid entry has 2 volume entries
MDVolumeType	1	1	Customer limit volume type
MDEntrySize	4	00000004	Customer limit volume == 4
MDVolumeType	1	0	Total Limit volume type
MDEntrySize	4	00000014	Total Limit volume == 20

MDEntryType	1	‘1’	Ask entry
MDEntryPx	4	FE00000010	FE == -2 exponent, 00000010 == 16 mantissa, value == 0.16
NoMDVolume Entries	1	1	Ask entry has 1 volume entry.
MDVolumeType	1	0	Total limit volume type (no customer volume for Ask)
MDEntrySize	4	0000000A	Total Limit Volume == 10

4 Packet and Message Header Format

All messages are sent in Multicast packets. Each packet consists of a packet header and one or more messages.

3 - Packet Format

Packet (a.k.a. Block)					
Packet Header					Contents
Version	**Length**	**Sending Time**	**Number of messages**	**First Msg Seq #**	**Messages...**

4.1 Packet Header

Each packet has a packet header that appears once at the beginning of the packet. The packet header has the following structure:

4 - Packet Header

Field Name	Type	Length (Bytes)	Comments
Version	uInt32	1	The version associated with the contents and format of this header. Currently, this will be a constant value of 1.
Length	uInt32	2	Length of the packet including this length field and the version. Note that this is a 2 byte length.
Sending Time	uInt64	8	The time that this packet was sent. It applies to all messages in this packet.
Number of messages	uInt32	1	The number of messages in this packet.
First Msg Seq #	uInt32	4	The sequence number on the first message in this packet.

The version of a packet indicates the format of the packet. This may be incremented in future releases to indicate a change in the format of the packet. Initially, it is set to the number 1.

The Packet Length is encoded as a 2 byte (16 bit) unsigned integer that includes the length of the version, the 2 byte Packet Length itself, and the remainder of the packet.

The Sending Time is the time that the CSM application published the packet on the feed. The sending time is the millisecond timestamp from midnight, January 1, 1970 UTC.

The "First Msg Seq #" is the sequence number of the first message of this packet, and the "Number of Messages" indicates the total number of messages contained in the packet.

For verification of data at the channel level, one could compute the expected "first msg seq #" of the next packet by adding the number of messages to the current packet's "first msg seq #".

4.2 Message Header

A packet contains multiple messages. Each message is preceded by a message header common to all messages.

5 - Message Format

Message						
Message Header				Contents		
		Template defined fields				
Length	**Template ID**	**Msg Type**	**Msg Seq #**	**Field #1**	**…**	**Field #N**

6 - Message Header

Field ID	Field Name	Type	Length (Bytes)	Comments
	Message Length	uInt32	2	The length of this message including the 2 bytes for this length field.
	Template ID	uInt32	1	The Template ID is for decoding the message. See table: *2 - Templates and their IDs*
35	MessageType	String	1	See table: *7 –Message Types*
34	MsgSeqNum	uInt32	4	Sequence Number

The Message Length is encoded as a 2 byte (16 bit) unsigned integer that includes itself and the remainder of the message including all Message Header fields.

The Template ID defines the specific Structure of the message.

The Message Type defines the market data message type compliant to the FIX standard.

The message sequence number is a consecutively increasing number from the previous message. The first message in a packet will start with 1 number greater than the last message in the previous packet

7 –Message Types

d	Security Definition
W	Market Data Refresh
X	Current Market Update
0	Heartbeat

4.2.1 Message Sequence Numbers

Every packet has a "first" sequence number in the header. This is the number associated with the first message in the packet. Each subsequent message in the packet has a sequence number that is one greater than the previous message. The next packet will have a starting sequence number that is one more than the last message in the previous packet except for the start of the trading day's session and in the event of a Cboe system failure. When the session is started the sequence number will be reset to

1. During a Cboe system failure the sequence number can reset to a lower value than was previously seen prior to the failure. In either case, the sequence number will again increase by one for each message.

Each channel of a feed has its own sequence number associated with it starting with sequence number 1. Verification of message sequence numbering must be done for each individual channel.

Firms must ensure that the sequence numbers maintain continuity. Any deviation from an expected sequence number must be considered as an error condition. Firms are required to take appropriate recovery action any time that an unexpected sequence number is detected.

4.2.2 Recovery from Unexpected Message Sequence Numbers

Each message sent on a channel causes its *MsgSeqNumber* to increment by one. To detect missing data at the channel-level, compare each incoming *MsgSeqNumber* with the last received *MsgSeqNumber* + 1. If the incoming *MsgSeqNumber* is not equal to the (last received *MsgSeqNumber* + 1), data is missing from the channel.
Regardless of whether missing data is detected or not, the *MsgSeqNumber* of the incoming message should be stored associated with the channel so subsequent missing data can be detected.

When missing data is detected at the channel-level, all market data for products that were received over that channel should be treated as "suspect", meaning their market data may be incorrect. At the time missing data is detected on a channel, there is no way to know which product's data is missing, therefore, all products received from that channel must be treated as though the market data for those products may be incorrect.

Products marked as suspect or possibly-incorrect should remain in that state until a Market Data Refresh for that product is received. When a Market Data Refresh is received, use it to update the suspect product's market data and mark the product's market data as no longer suspect. Note, however, that it will take approximately two minutes for every product to be updated with a Market Data Refresh. You can use the *ApplSeqNum* of the Market Data Refresh to detect when the refresh cycle has completed.

If desired, it is possible to implement recovery of Current Market data. You will need to track the Current Market data suspect separately, however. When missing data is detected on a channel, mark the Current Market data suspect for all products received over that channel. If a Current Market Update arrives, you can update the Current Market and mark it non-suspect, because Current Market Update messages contain the full top of book state.

5 Messages

5.1 Security Definition Message – Template ID 13

Security definitions describe an exchange's products by name and trading parameters and associates those products with a product security ID.

Security definition messages for all products are sent to the security definition channel in a repeating cycle of approximately 2 minutes duration. When one cycle completes the next begins immediately

When a *new* product is added intraday, a security definition will be sent once to the *data* channel for the newly added security, just before the first update for that security is sent to the data channel.

At startup, to build the initial set of products, a complete cycle of the security definition channel must be read, while concurrently reading the data channels also watching for security definitions. To detect a cycle, watch for the same securityID to appear twice on the security definition channel.

To insure that you see security definitions for all products, you must process security definitions from the data channels and from the security definition channel. If a product is added, a security definition is sent to the data channel immediately, but that newly added product may not appear in the current security definition channel's cycle. It is for this reason that you must also read the data channel security definitions when building the initial set of products from the security definition channel.

After a complete security definition cycle is read, it is not necessary to continue reading the Security Definition channel unless a message on one of the data channels is dropped. The data that was dropped could have been a new security, so another security definition cycle should be read to re-establish the complete set of products. As long as there is no missing data on any data channel, however, it is not necessary to read the security definition channel after the first complete cycle.

The Security Definition message fields are as follows:

8 - Security Definition Message Structure

Field ID	Field Name	Type	Length (Bytes)	Comments
	Standard Header			See table: *7 –Message Types*, MessageType = "d"
167	SecurityType	string		See table: *10 - Security Types*
207	SecurityExchange	single byte string	1	See table: *11 - Security Exchanges*
55	Symbol	string		Symbol of the class
143	TargetLocationID	string		See Target Location ID
21004	ClassKey	uInt32	4	Class key
48	SecurityID	uInt32	4	Product key
541	MaturityDate	uInt64	8	Expiration date: Format is "YYYYMMDD" This field is required for options and futures
423	PriceType	uInt32	1	Specifies how to interpret the value in the StrikePrice field. See table: 12 - Security Price Types.
202	StrikePrice	decimal	5	First byte is the exponent, last 4 bytes are the mantissa This field is required for options
201	PutOrCall	uInt32	1	This field is required for options. See table: 13 - Security Put / Call
21005	MinimumStrike PriceFraction	decimal	5	The multiple by which the strike price can be
21006	MaxStrikePrice	decimal	5	The maximum allowable strike price.
21007	PremiumBreak Point	decimal	5	The premium price where above and below fractions take effect.
21008	MinimumAbove PremiumFraction	decimal	5	The multiple that premium can be when above the break point.
21009	MinimumBelow PremiumFraction	decimal	5	The multiple that premium can be when below the break point.
21010	ExerciseStyle	uInt32	1	See table: *14 - Exercise Styles*
996	CurrencyCode	string		Not used
311	UnderlyingSymbol	string		The underlying symbol
310	UnderlyingType	string		See table: *10 - Security Types*
231	ContractSize	uInt32	4	The number of contracts per unit of size

9 - Security Definition Template

```
<template   name="MDSecurityDefinition"     id="13">
    <string     name="MessageType"                  id="35"     byteLength="1"
                                                                value="d" />
    <uInt32     name="MsgSeqNum"                    id="34"     byteLength="4"/>
    <string     name="SecurityType"                 id="167"/>
    <string     name="SecurityExchange"             id="207"    byteLength="1"/>
```

```
    <string     name="Symbol"                          id="55"/>
    <string     name="TargetLocationID"                id="143"/>
    <uInt32     name="ClassKey"                        id="21004"  byteLength="4"/>
    <uInt32     name="SecurityID"                      id="48"/>
    <uInt64     name="MaturityDate"                    id="541"    byteLength="8"/>
    <uInt32     name="PriceType"                       id="423"    byteLength="1"/>
    <decimal    name="StrikePrice"                     id="202"    byteLength="5"/>
    <uInt32     name="PutOrCall"                       id="201"    byteLength="1"/>
    <decimal    name="MinimumStrikePriceFraction"      id="21005"  byteLength="5"/>
    <decimal    name="MaxStrikePrice"                  id="21006"  byteLength="5"/>
    <decimal    name="PremiumBreakPoint"               id="21007"  byteLength="5"/>
    <decimal    name="MinimumAbovePremiumFraction"     id="21008"  byteLength="5"/>
    <decimal    name="MinimumBelowPremiumFraction"     id="21009"  byteLength="5"/>
    <uInt32     name="ExerciseStyle"                   id="21010"  byteLength="1"/>
    <string     name="CurrencyCode"                    id="996"/>
    <string     name="UnderlyingSymbol"                id="311" />
    <string     name="UnderlyingType"                  id="310"/>
    <uInt32     name="ContractSize"                    id="231"    byteLength="4"/>
    </template>
```

10 - Security Types

OPT	Options

11 - Security Exchanges

C	Cboe

12 - Security Price Types

1	Percentage
3	Fixed Amount

13 - Security Put / Call

0	PUT
1	CALL

14 - Exercise Styles

0	American
1	European

5.1.1 Target Location ID

Target Location ID is an ASCII-character encoded string containing a numerical index that indicates over which data channel the market data for a given product will be delivered. It is a zero-based index of the data channel number for a feed. This can be used if desired to figure out over which data channel a product will be transmitted.

The relationship between the TargetLocationID and the data channel index can be found in *Appendix A – Multicast Group and Port Information*

The TargetLocationID is unique only within a feed and not across feeds. There is no cross-feed reference.

5.2 Market Data Refresh Message – Template ID 20

The Market Data Refresh message contains a snapshot / refresh of the Current Market information for one product.

Market Data Refresh messages are sent continuously to the data channel associated with a product in a repeating cycle of approximately two minutes duration. Market Data Refresh messages are co-mingled with Current Market Update and other messages.

Market Data Refresh messages are used to establish the initial state of market data for a security and to recover market data if messages are dropped. Under *normal* processing, once a complete cycle of Market Data Refresh messages are processed, Market Data Refresh messages can be ignored unless a gap is detected in the data. If a gap is detected, a complete cycle must be processed to synchronize with the feed and re-establish the correct market data for all products.

For more information regarding recovery from missing data, see section 4.2.2 - *Recovery from Unexpected Message Sequence Numbers.*

The *ApplSeqNum* field is a per-data-channel sequence number that resets to 1 for each repeating refresh cycle. This can be used to detect when a refresh cycle for each data channel is complete.

Market data is contained in MDEntries with an MDEntryType indicating the kind of market data. The Market Data Refresh message may include any of the following MDEntryTypes (See table *18 – MD Entry Type*).

- Bid
- Ask
- Trade (Last Sale)
- Open
- High
- Low

The *MDEntrySz* field is relevant only for an MDEntry whose MDEntryType == Bid, Ask, or Trade. For all other MDEntryTypes it will be set to zero. For MDEntryType == Trade, the MDEntrySz refers to the size of the latest trade.

The *MDVolumeType* field is relevant only for an MDEntry whose MDEntryType == Bid or Ask. The MDVolumeType indicates the type of volume contained in the MDEntrySize field. For all other MDEntryTypes it will be set to zero.

To process a MarketDataRefresh, all market data should be cleared and re-built entirely from the data present in the MDEntries contained in the Market Data Refresh message.

The absence of an MDEntry for a particular MDEntryType indicates that the data for that MDEntryType is empty (does not exist). For example:

- For products that do not have a bid market (no-price), an MDEntryType of Bid ('0') will NOT be present in the MDEntries sequence.
- For products that do not have an ask market (no-price), an MDEntryType of Ask ('1') will NOT be present in the MDEntries sequence.
- For products that are not yet open or that do not have a LastSale price (meaning they have not traded), there will be NO MDEntryTypes sent for Trade ('3') Open ('4'), High ('7'), nor Low ('8').

MDEntryTypes of Bid and Ask may each appear zero to 4 times, once for each type of volume, if volume is present for a particular volume type and MDEntryType. The absence of an MDEntryType Bid or Ask for a particular volume type indicates there is no volume for that volume type.

MDEntryTypes Trade, Open, High, Low will be present in the Market Data Refresh only when a product has traded and there is a valid Trade (Last Sale) price, Open, High, and Low price. Their absence implies the Trade, Open, High, and Low are zero.

15 - Market Data Refresh Message Structure

Field ID	Field Name	Type	Length (Bytes)	Comments
	Standard Header			See table: *7 –Message Types,* MessageType = "W"
21004	ClassKey	uInt32	4	Class key
48	SecurityID	uInt32	4	Product ID
326	SecurityTrading Status	uInt32	1	See table: *17 - Security Trading Status*
423	PriceType	uInt32	1	Specifies how to interpret the value in the MDEntryPx field. See table: *12 - Security Price Types*
1181	ApplSeqNum	uInt32	4	Sequence number for Refresh message, will be set per line and reset to 1 (one) when refresh is completed
140	PrevClosePx	decimal	5	Previous day's close price for security. **Note**: This may contain the *NO PRICE Decimal Value* described in section *3.3.3.1*
1020	TradeVolume	uInt32	4	Total trade volume for security
	MDEntries	Sequence		

The `MDEntries` sequence field has the following sub fields (For detailed sequence field format information, refer to the Fields Data Types section in this document):

Field ID	Field Name	Type	Length (Bytes)	Comments
268	NoMDEntries	length	1	Number of MDEntries in this message. Will not exceed 255
The Following Fields Repeat NoMDEntries times				
269	MDEntryType	single byte string	1	Entry Type. See table: *18 – MD Entry Type*
270	MDEntryPx	decimal	5	Price associated with MDEntryType: First byte is exponent, last 4 bytes is mantissa.
271	MDEntrySize	uInt32	4	Quote Quantity
21001	MDVolumeType	uInt32	1	See table: *19 - Volume Types*

16 - Market Data Refresh Template

```
<template   name="MarketDataRefresh"        id="20">
    <string     name="MessageType"              id="35"     byteLength="1"
                                                            value="W" />
    <uInt32     name="MsgSeqNum"                id="34"     byteLength="4"/>
    <uInt32     name="ClassKey"                 id="21004"  byteLength="4"/>
    <uInt32     name="SecurityId"               id="48"     byteLength="4"/>
    <uInt32     name="SecurityTradingStatus"    id="326"    byteLength="1"/>
    <uInt32     name="PriceType"                id="423"    byteLength="1"/>
    <uInt32     name="ApplSeqNum"               id="1181"   byteLength="4"/>
    <decimal    name="PrevClosePx"              id="140"    byteLength="5"/>
    <uInt32     name="TradeVolume"              id="1020"   byteLength="4"/>
    <sequence   name="MDEntries">
        <length     name="NoMDEntries"          id="268"    byteLength="1"/>
        <string     name="MDEntryType"          id="269"    byteLength="1"/>
        <decimal    name="MDEntryPx"            id="270"    byteLength="5"/>
        <uInt32     name="MDEntrySize"          id="271"    byteLength="4"/>
        <uInt32     name="MDVolumeType"         id="21001"  byteLength="1"/>
    </sequence>
</template>
```

17 - Security Trading Status

21	Pre Open
22	Market in Opening Rotation

18 – MD Entry Types

0	Bid
1	Ask
2	Trade (LastSale)
3	Index Value (Used only in Index Value messages)
4	Opening Price
6	Settlement value (Used only in Settlement messages)
7	High (Trading Session High Price)
8	Low (Trading Session Low Price)

19 - Volume Types

0	Total Limit
1	Customer Limit
2	Total Contingency (All or None)
3	Customer Contingency (All or None)

5.3 Current Market Update Message – Template ID 12

The Current Market Update message contains real-time top of book market updates for a security.

Top of book market data is contained in MDEntries with an MDEntryType indicating the kind of market data. The Market Data Refresh message may include any of the following MDEntryTypes (See table *18 – MD Entry Type*).

- Bid
- Ask

An MDVolumeType in each MDEntry indicates the type of volume that is associated with a Bid or Ask price See table: *19 - Volume Types*

A Current Market Update may contain zero to 4 Bid MDEntries and zero to 4 Ask MDEntries, one for each of the 4 volume types if there is volume for that volume type and MDEntryType. The absence of an MDEntry and volume type indicates no volume for that volume type exists for that MDEntryType. There may be zero MDEntries for Bid or zero MDEntries for Ask, which means a no-bid or no-ask market respectively.

IMPORTANT: For each series, the last quote disseminated prior to the open represents the best market prior to OPG orders being cancelled. It is important to understand that series are eligible for inclusion in the special opening quotation ("SOQ") only if there is remaining buy interest (i.e., "non-zero" bid) at the conclusion of the opening rotation. In cases where remaining buy interest consists of OPG buy orders, the first OPRA quote may show a '0' bid, yet the series may be included in the settlement calculation due to the existence of remaining buy OPG interest at the conclusion of the opening rotation. All OPG orders are cancelled after the rotation is completed but prior to the first quote dissemination to OPRA.

20 – Current Market Update Message Structure

Field ID	Field Name	Type	Length (Bytes)	Comments
	Standard Header			See table: *7 –Message Types*, MessageType = "X"
21004	ClassKey	uInt32	4	Class key
48	SecurityID	uInt32	4	Product ID
326	SecurityTrading Status	uInt32	1	See table: *17 - Security Trading Status*.
423	PriceType	uInt32	1	Specifies how to interpret the value in the MDEntryPx field. See table: 12 - Security Price Types
	MDEntries	Sequence		

The `MDEntries` sequence field has the following sub fields (For detailed sequence field format information, refer to the Fields Data Types section in this document):

Field ID	Field Name	Type	Length (Bytes)	Comments
268	NoMDEntries	length	1	Number of MDEntries in this message. Will not exceed 255
The Following Fields Repeat NoMDEntries times				
269	MDEntryType	single byte string	1	Entry Type. See table: *18 – MD Entry Type*
270	MDEntryPx	decimal	5	Quote Price: First byte is the exponent, last 4 bytes are the mantissa
271	MDEntrySize	uInt32	4	Quote Quantity
21001	MDVolumeType	uInt32	1	See table: *19 - Volume Types*

21 - Current Market Update Template

```
<template   name="CurrentMarketUpdate"      id="12">
    <string     name="MessageType"              id="35"     byteLength="1"
                                                            value="X" />
    <uInt32     name="MsgSeqNum"                id="34"     byteLength="4"/>
    <uInt32     name="ClassKey"                 id="21004"  byteLength="4"/>
    <uInt32     name="SecurityId"               id="48"     byteLength="4"/>
    <uInt32     name="SecurityTradingStatus"    id="326"    byteLength="1"/>
    <uInt32     name="PriceType"                id="423"    byteLength="1"/>
    <sequence   name="MDEntries">
        <length     name="NoMDEntries"          id="268"    byteLength="1"/>
        <string     name="MDEntryType"          id="269"    byteLength="1"/>
        <decimal    name="MDEntryPx"            id="270"    byteLength="5"/>
        <uInt32     name="MDEntrySize"          id="271"    byteLength="4"/>
        <uInt32     name="MDVolumeType"         id="21001"  byteLength="1"/>
    </sequence>
</template>
```

5.4 EOP Message (Expected Opening Price and Size) – Template ID 15

The EOP message contains expected opening price and size information for a security. EOP messages will be published while a product is in the pre-open and opening rotation states. It will cease to be published while a product is in any other state such as open. This message will repeat as often as the trading engine creates it; every five seconds.

22 - EOP Message Structure

Field ID	Field Name	Type	Length (Bytes)	Comments
	Standard Header			See table: *7 –Message Types*, MessageType = "X"
21004	ClassKey	uInt32	4	Class key
48	SecurityID	uInt32	4	Product ID
270	EOP	decimal	5	The expected opening price
271	EOS	uInt32	4	The expected opening size
21002	Type	uInt32	1	See table: *24 - EOP Types*
21003	LegalMarket	uInt32	1	See table: *25 - Legal Markets*

23 - EOP Template

```
<template   name="EOP"                      id="15">
    <string     name="MessageType"              id="35"         byteLength="1"
                                                                value="X" />
    <uInt32     name="MsgSeqNum"                id="34"         byteLength="4"/>
    <uInt32     name="ClassKey"                 id="21004"      byteLength="4"/>
    <uInt32     name="SecurityId"               id="48"         byteLength="4"/>
    <decimal    name="EOP"                      id="270"        byteLength="5"/>
    <uInt32     name="EOS"                      id="271"        byteLength="4"/>
    <uInt32     name="Type"                     id="21002"      byteLength="1"/>
    <uInt32     name="LegalMarket"              id="21003"      byteLength="1"/>
</template>
```

24 - EOP Types

Value	Description
0	Undefined – not used.
1	Opening Price
2	Need More Sellers and Size
3	Need More Buyers and Size
4	No Opening Trades
5	Multiple Opening Prices
6	Need Quote To Open
7	Price Not In Quote Range
8	Need DPM Quote To Open
9	DPM Quote Invalid
10	Price Not In BOTR Range

25 - Legal Markets

0	Not a Legal Market
1	Legal Market

5.5 Heartbeat Message (Line Integrity Message) – Template ID 16

This message contains only a standard header. The heartbeat will repeat at a regular interval.

26 - Heartbeat Message Structure

Field ID	Field Name	Type	Length (Bytes)	Comments
	Standard Header			See table: *7 –Message Types*, MessageType = "0" (zero)

27 - Heartbeat Template

```
<template   name="Heartbeat"                id="16">
  <string       name="MessageType"              id="35"         byteLength="1"
                                                                value="0" />
  <uInt32       name="MsgSeqNum"                id="34"         byteLength="4"/>
</template>
```

6 Appendix A – Multicast Group and Port Information

Refer to the CFN Network Specifications document for complete multicast group and port information.

Cboe OAF Primary Groups

Group	Description	Target Location ID	A or B	Port	RP	Source Networks
233.103.126.88/30	Cboe OAF Prod A Groups		A			
233.103.126.88	Cboe Market Data	0	A	64860	170.137.128.253	170.137.144.0/26
233.103.126.89	Cboe Market Data	1	A	64861	170.137.128.253	170.137.144.0/26
233.103.126.90	*Reserved for future use*		A	64862	170.137.128.253	170.137.144.0/26
233.103.126.91	Cboe Securities Definition		A	64863	170.137.128.253	170.137.144.0/26

Cboe OAF Backup Groups

Group	Description	Target Location ID	A or B	Port	RP	Source Networks
233.103.126.216/30	Cboe OAF Prod B Groups		B			
233.103.126.216	Cboe Market Data	0	B	64864	170.137.128.254	170.137.144.64/26
233.103.126.217	Cboe Market Data	1	B	64865	170.137.128.254	170.137.144.64/26
233.103.126.218	*Reserved for future use*		B	64866	170.137.128.254	170.137.144.64/26
233.103.126.219	Cboe Securities Definition		B	64867	170.137.128.254	170.137.144.64/26

Cboe OAF Disaster Recovery Groups

Group	Description	Target Location ID	A or B	Port	RP	Source Networks
233.65.120.168/30	Cboe OAF Disaster Recovery A Groups		A			
233.65.120.168	Cboe Market Data	0	A	64860	170.137.255.124	170.137.1.128/26
233.65.120.169	Cboe Market Data	1	A	64861	170.137.255.124	170.137.1.128/26
233.65.120.170	*Reserved for future use*		A	64862	170.137.255.124	170.137.1.128/26
233.65.120.171	Cboe Securities Definition		A	64863	170.137.255.124	170.137.1.128/26

7 Appendix B – Examples

7.1 Understanding the Hex Data Diagrams

Through this section you will see hexadecimal printouts of data that is from a channel. Each of the examples will be in a similar format.

```
Offset    Hexadecimal data 4 bytes per group   ASCII representation
00000000  CD580000 00030000 0134C8E5 A8992B6C  |.X.......4....+l|
00000010  031FFE00 00006600 00006401 00        |......f...d..   |
```

The first column of numbers is the zero-based byte offset of the first hex byte on that line, expressed as a hexadecimal offset. The next 4 columns of data are hexadecimal values for the bytes, with 4 bytes per group. Spaces are for formatting purposes only. Characters between pipes ("|") are ascii representations of the hex data. Periods indicate non-printable values.

The first 16 bytes of each hex dump is the packet header.

7.2 Packet Header Example

1 - Packet Header Hex Dump

```
00000000  01038B00 000135A6 B387070B 00000000  |......5.........|
```

2 - Packet Header Decoded

SendTime(02/22 14:14:37.831)
BlkVersion(1) BlkSize(907) MsgsInBlock(11) FirstMsgSeq(0)

7.3 Security Definition Message

In these examples it is assumed that this and all other security definition messages have been received and processed.

3 - Security Definition Hex Dump

```
00000000  01034E00 0001375B C7509B0A 0024419A  |..N...7[.P...$A.|
00000010  00530D64 0024419A 034F5054 43044144  |.S.d.$A..OPTC.AD|
00000020  42450134 1C1A88EA 207643D3 00000000  |BE.4.... vC.....|
00000030  013305BC 03FD0000 BF6800FC 000004E2  |.3.......h......|
00000040  FE000F42 36FE0000 012CFE00 000005FE  |...B6....,......|
00000050  00000001 00000441 44424502 43530000  |.......ADBE.CS..|
00000060  00640000 530D6400 24419B03 4F505443  |.d.             |
```

4 - Security Definition Decoded

```
Message#(1) MsgSize(83)
TemplateId(13)   AppMsg: MDSecurityDefinition Type: d
  FieldType        FieldName                          Id     Value
  sbSTRING:        MessageType                        35     d
  UINT32:          MsgSeqNum                          34     2376090
  STRING:          SecurityType                       167    OPT
  sbSTRING:        SecurityExchange                   207    C
  STRING:          Symbol                             55     ADBE
  STRING:          TargetLocationID                   143    4
  UINT32:          ClassKey                           21004  471501034
  UINT32:          SecurityID                         48     544621523
  UINT64:          MaturityDate                       541    20121020
  UINT8:           PriceType                          423    3
  uDECIMAL:        StrikePrice                        202    49.000
  UINT8:           PutOrCall                          201    0
  uDECIMAL:        MinimumStrikePriceFraction         21005  0.1250
  uDECIMAL:        MaxStrikePrice                     21006  9999.90
  uDECIMAL:        PremiumBreakPoint                  21007  3.00
  uDECIMAL:        MinimumAbovePremiumFraction        21008  0.05
  uDECIMAL:        MinimumBelowPremiumFraction        21009  0.01
  UINT8:           ExerciseStyle                      21010  0
  STRING:          CurrencyCode                       996
  STRING:          UnderlyingSymbol                   311    ADBE
  STRING:          UnderlyingType                     310    CS
  UINT32:          ContractSize                       231    100
  SEQUENCE:        NoLegs                             555    Len=0
```

7.4 Heartbeat Message

5 - Heartbeat Hex Dump

```
00000000  01001800 000135A7 00C6C901 00000F95  |......5.........|
00000010  00081030 00000F95                    |...0....        |
```

6 - Heartbeat Decoded

```
Message#(1) MsgSize(8)
TemplateId(16)   AppMsg: Heartbeat Type: 0
  FieldType        FieldName                          Id     Value
  sbSTRING:        MessageType                        35     0
  UINT32:          MsgSeqNum                          34     3989
```

7.5 Creating a test market

Quote entered for product A Feb-18-12 17.00 CALL .80-1.20 20x20.

7 - Current Market Update Hex Dump

```
00000000  01003900 000135A6 EF5DB201 000007AB  |..9...5..]......|
00000010  00290C58 000007AB 04200003 45B88E5E  |.).X..... ..E..^|
00000020  11030230 FE000000 50000000 140031FE  |...0....P.....1.|
00000030  00000078 00000014 00                 |...x.....       |
```

8 - Current Market Update Decoded

```
Message#(1) MsgSize(41)
TemplateId(12)   AppMsg: CurrentMarketUpdate Type: X
  FieldType        FieldName                        Id     Value
  sbSTRING:        MessageType                      35     X
  UINT32:          MsgSeqNum                        34     1963
  UINT32:          ClassKey                         21004  69206019
  UINT32:          SecurityId                       48     1169722974
  UINT8:           SecurityTradingStatus            326    17
  UINT8:           PriceType                        423    3
  SEQUENCE:        NoMDEntries                      268    Len=2
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    0
      uDECIMAL:        MDEntryPx                    270    0.80
      UINT32:          MDEntrySize                  271    20
      UINT8:           MDVolumeType                 21001  0
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    1
      uDECIMAL:        MDEntryPx                    270    1.20
      UINT32:          MDEntrySize                  271    20
      UINT8:           MDVolumeType                 21001  0
```

9 - Current Market Update Formatted

```
15:19:59.410 (    1963)  12-X-MktUpd    mcast(9) 1169722974
A.20120218.C 17.0 #Ent: 2
    0 Update:   Typ: Bid  Prc: 0.8 Vol:     20 VolType: TLMT
    1 Update:   Typ: Ask  Prc: 1.2 Vol:     20 VolType: TLMT
```

7.6 Market Data Refresh Message

This is a Market Data Refresh message.

10 - Current Market Refresh Hex Dump

```
00000000  01004600 0001638D 73431F01 00000036  |..F...c.sC.....6|
00000010  00361457 00000036 042006E3 7A8744DC  |.6.W...6. ..z.D.|
00000020  15030000 0002F780 00000000 00000002  |................|
00000030  30FE0000 00780000 00640030 FE000000  |0....x...d.0....|
00000040  78000000 6401                        |x...d.          |
```

11 - Current Market Refresh Decoded

```
Message#(1) MsgSize(54)
TemplateId(20) TemplateName(MktDataRefresh)   AppMsg: MarketDataRefresh
Type: W
  FieldType        FieldName                        Id     Value
  sbSTRING:        MessageType                      35     W
  UINT32:          MsgSeqNum                        34     54
  UINT32:          ClassKey                         21004  69207779
  UINT32:          SecurityId                       48     2055685340
  UINT8:           SecurityTradingStatus            326    21
  UINT8:           PriceType                        423    3
  UINT32:          ApplSeqNum                       1181   2
  uDECIMAL:        PrevClosePx                      140    -2.147483648
  UINT32:          TradeVolume                      1020   0
  SEQUENCE:        NoMDEntries                      268    Len=2
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    0
      uDECIMAL:        MDEntryPx                    270    1.20
      UINT32:          MDEntrySize                  271    100
      UINT8:           MDVolumeType                 21001  0
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    0
      uDECIMAL:        MDEntryPx                    270    1.20
      UINT32:          MDEntrySize                  271    100
      UINT8:           MDVolumeType                 21001  1
```

12 - Current Market Refresh Formatted

```
18-05-23 09:42:11.359 (       54)  20-W-MDRfrsh   mcast(9) 2055685340
AZN.20180119.P 15.0 #(2) PCpx:        NP Tvol:      0
    0 Refresh:  Typ: Bid  Prc:      1.20 Vol:    100 VolType: TLMT
    1 Refresh:  Typ: Bid  Prc:      1.20 Vol:    100 VolType: CLMT
```

7.7 Updating the market

Quote changed to .90-1.10 30x50.

13 - Current Market Update Hex Dump

```
00000000  01003900 000135A7 00DF0B01 000009FE  |..9...5.........|
00000010  00290C58 000009FE 04200003 45B88E5E  |.).X..... ..E..^|
00000020  11030230 FE000000 5A000000 1E0031FE  |...0....Z.....1.|
00000030  0000006E 00000032 00                |...n...2.       |
```

14 - Current Market Update Decoded

```
Message#(1) MsgSize(41)
TemplateId(12)   AppMsg: CurrentMarketUpdate Type: X
  FieldType        FieldName                        Id     Value
  sbSTRING:        MessageType                      35     X
  UINT32:          MsgSeqNum                        34     2558
  UINT32:          ClassKey                         21004  69206019
  UINT32:          SecurityId                       48     1169722974
  UINT8:           SecurityTradingStatus            326    17
  UINT8:           PriceType                        423    3
  SEQUENCE:        NoMDEntries                      268    Len=2
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    0
      uDECIMAL:        MDEntryPx                    270    0.90
      UINT32:          MDEntrySize                  271    30
      UINT8:           MDVolumeType                 21001  0
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    1
      uDECIMAL:        MDEntryPx                    270    1.10
      UINT32:          MDEntrySize                  271    50
      UINT8:           MDVolumeType                 21001  0
```

15 - Current Market Update Formatted

```
15:39:30.244 (    2558)  12-X-MktUpd    mcast(9) 1169722974
A.20120218.C 17.0 #Ent: 2
    0 Update:   Typ: Bid  Prc: 0.9 Vol:     30 VolType: TLMT
    1 Update:   Typ: Ask  Prc: 1.1 Vol:     50 VolType: TLMT
```

7.8 One-sided Market

Enter an order: sell 30@0.90.

16 - One-sided Market Hex Dump

```
00000000  01002E00 000135AA A98F0401 000003E5  |......5.........|
00000010  001E0C58 000003E5 04200003 45B88E5E  |...X..... ..E..^|
00000020  11030131 FE000000 5A000000 1E00      |...1....Z.....  |
```

17 - One-sided Market Decoded

```
Message#(1) MsgSize(30)
TemplateId(12)   AppMsg: CurrentMarketUpdate Type: X
  FieldType        FieldName                        Id     Value
  sbSTRING:        MessageType                      35     X
  UINT32:          MsgSeqNum                        34     997
  UINT32:          ClassKey                         21004  69206019
  UINT32:          SecurityId                       48     1169722974
  UINT8:           SecurityTradingStatus            326    17
  UINT8:           PriceType                        423    3
  SEQUENCE:        NoMDEntries                      268    Len=1
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                  269    1
      uDECIMAL:        MDEntryPx                    270    0.90
      UINT32:          MDEntrySize                  271    30
      UINT8:           MDVolumeType                 21001  0
```

18 - One-sided Market Formatted

```
08:42:13.380 (     997)  12-X-MktUpd    mcast(9) 1169722974
A.20120218.C 17.0 #Ent: 1
    0 Update:   Typ: Ask  Prc: 0.9 Vol:     30 VolType: TLMT
```

7.9 Contingent Customer Order

Enter an order: CALL (1169722980) sell 30@0.90 customer AON with a quote of 90-1.10 15x15.

19 - AON Order Hex Dump

```
00000000  01004F00 000135AB 17538401 00000973  |..O...5..S.....s|
00000010  003F0C58 00000973 04200003 45B88E5E  |.?.X...s. ..E..^|
00000020  11030430 FE000000 5A000000 0F0031FE  |...0....Z.....1.|
00000030  0000005A 0000001E 0231FE00 00005A00  |...Z.....1....Z.|
00000040  00001E03 31FE0000 006E0000 000F00    |....1....n..... |
```

20 - AON Order Decoded

```
Message#(1) MsgSize(63)
TemplateId(12)   AppMsg: CurrentMarketUpdate Type: X
  FieldType        FieldName                          Id     Value
  sbSTRING:        MessageType                        35     X
  UINT32:          MsgSeqNum                          34     2419
  UINT32:          ClassKey                           21004  69206019
  UINT32:          SecurityId                         48     1169722974
  UINT8:           SecurityTradingStatus              326    17
  UINT8:           PriceType                          423    3
  SEQUENCE:        NoMDEntries                        268    Len=4
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                    269    0
      uDECIMAL:        MDEntryPx                      270    0.90
      UINT32:          MDEntrySize                    271    15
      UINT8:           MDVolumeType                   21001  0
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                    269    1
      uDECIMAL:        MDEntryPx                      270    0.90
      UINT32:          MDEntrySize                    271    30
      UINT8:           MDVolumeType                   21001  2
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                    269    1
      uDECIMAL:        MDEntryPx                      270    0.90
      UINT32:          MDEntrySize                    271    30
      UINT8:           MDVolumeType                   21001  3
    MSG_GROUP NumElems: 4
      sbSTRING:        MDEntryType                    269    1
      uDECIMAL:        MDEntryPx                      270    1.10
      UINT32:          MDEntrySize                    271    15
      UINT8:           MDVolumeType                   21001  0
```

21 - AON Order Formatted

```
10:42:07.108 (    2419)  12-X-MktUpd    mcast(9) 1169722974
A.20120218.C 17.0 #Ent: 4
    0 Update:   Typ: Bid  Prc: 0.9 Vol:     15 VolType: TLMT
    1 Update:   Typ: Ask  Prc: 0.9 Vol:     30 VolType: AON
    2 Update:   Typ: Ask  Prc: 0.9 Vol:     30 VolType: CAON
    3 Update:   Typ: Ask  Prc: 1.1 Vol:     15 VolType: TLMT
```

7.10 All Markets are removed

After all quotes and orders are canceled an empty market is transmitted.

22 - Cancelled Market Hex Dump

```
00000000  01002300 000135AB 26E76501 00000B7A  |..#...5.&.e....z|
00000010  00130C58 00000B7A 04200003 45B88E5E  |...X...z. ..E..^|
00000020  110300                              |...             |
```

23 - Cancelled Market Decoded

```
TemplateId(12)   AppMsg: CurrentMarketUpdate Type: X
  FieldType        FieldName                      Id     Value
  sbSTRING:        MessageType                    35     X
  UINT32:          MsgSeqNum                      34     2938
  UINT32:          ClassKey                       21004  69206019
  UINT32:          SecurityId                     48     1169722974
  UINT8:           SecurityTradingStatus          326    17
  UINT8:           PriceType                      423    3
  SEQUENCE:        NoMDEntries                    268    Len=0
```

24 - Cancelled Market Formatted

```
10:59:08.005 (    2938)  12-X-MktUpd    mcast(9) 1169722974
A.20120218.C 17.0 #Ent: 0
```