MEMOIR Depth Feed

V1.3

04/05/21

This specification is being provided to you strictly for informational purposes solely for the purpose of developing or operating systems that interact with the Members Exchange, or MEMX. All proprietary rights and interest in this specification and the information contained herein shall be vested in MEMX and all other rights including, but without limitation, patent, registered design, copyright, trademark, service mark, connected with this publication shall also be vested in MEMX. No part of this specification may be redistributed or reproduced in any form or by any means or used to make any derivative work (such as translation, transformation, or adaptation) without written permission from MEMX. MEMX reserves the right to withdraw, modify, or replace the specification at any time, without notice. No obligation is made by MEMX regarding the level, scope, or timing of MEMX's implementation of the functions or features discussed in this specification.

THE SPECIFICATION IS PROVIDED "AS IS", "WITH ALL FAULTS" AND MEMX MAKES NO WARRANTIES AND DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, OR STATUTORY RELATED TO THE SPECIFICATIONS. MEMX IS NOT LIABLE FOR ANY INCOMPLETENESS OR INACCURACIES IN THE SPECIFICATIONS. MEMX IS NOT LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES RELATING TO THE SPECIFICATIONS OR THEIR USE.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

Table of Contents

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

< THIS PAGE INTENTIONALLY LEFT BLANK>

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

1 Overview

MEMOIR Depth Feed is a real-time depth of book feed directly from the MEMX Exchange.

The MEMOIR Depth Feed provides an event based binary messaging protocol that supports the following functionality:

- **Trading Session Information** - Provides the state of the MEMX trading session.
- **Order Data** - Provides the complete order book containing all displayed orders on MEMX.
- **Non Displayed Trades** - Provides trade information for executions on non-displayed orders.
- **Instrument Directory** - Provides a mapping of all supported MEMX security IDs to all their basic attributes.
- **Trading Actions** - Supplies trading status messages to inform participants of events for an instrument.

The MEMOIR Depth Feed is unidirectional and cannot be used to enter orders.

Application messages are implemented using a binary protocol based on SBE (Simple Binary Encoding).

2 Architecture

The MEMOIR Depth Feed is a sequenced message stream using fixed width binary messages.

The business level messaging is transported via the following protocols for framing and recovery:

- MEMX-UDP protocol - UDP Multicast based session transport for the real-time delivery of messages.
- MEMX-TCP protocol - TCP based gap fill replay service.

At the start of the day the a session ID will be provided, and messages begin at sequence number 1. An Instrument Directory message spin will be sent out to map all the traded symbols to a symbol locate code that will be used throughout the session. The locate code will not change during the course of the session, however intraday there may be subsequent updates to the same or new instruments.

In order to build the book properly all orders will be sent keyed upon an OrderID. Events on the order (Add, Replace, Reduce, Executed) will alter the order on the book. It is the responsibility of the client to organize their representation of the order book based upon time, side, size and price.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

3 Encoding

The MEMOIR Depth Feed uses the FIX Trading Community's Simple Binary Encoding (SBE) to specify message encoding. More information about SBE can be found at the FIX SBE XML Primer.

The MEMOIR Depth Feed is always encoded in Big Endian byte order.

3.1 Data Types

All encoding and decoding for SBE is centered around a set of basic primitive types.

For more information on primitive type encoding, see the SBE specification.

Type	Length (bytes)	Description	Value Range	Null Value	Null Value (Hex)
CHAR	1	ASCII Character	`0` (`NUL`) to `127` (`DEL`)	`0`	`0x00`
INT8	1	Signed Integer	`-127` to `127`	`-128`	`0x80`
INT16	2	Signed Integer	`-32767` to `32767`	`-32768`	`0x8000`
INT32	4	Signed Integer	`-2^31 + 1` to `2^31 - 1`	`-2^31` (`-2147483648`)	`0x80000000`
INT64	8	Signed Integer	`-2^63 + 1` to `2^63 - 1`	`-2^63`	`0x8000000000000000`
UINT8	1	Unsigned Integer	`0` to `254`	`255`	`0xFF`
UINT16	2	Unsigned Integer	`0` to `65534`	`65535`	`0xFFFF`
UINT32	4	Unsigned Integer	`0` to `2^32 - 2`	`2^32 - 1` (`4294967295`)	`0xFFFFFFFF`
UINT64	8	Unsigned Integer	`0` to `2^64 - 2`	`2^64 - 1`	`0xFFFFFFFFFFFFFFFF`

The MEMOIR specification does not use the SBE floating point data types.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

3.1.1 PriceType

Prices are encoded as a fixed-point scaled decimal, consisting of a signed long (int64) mantissa and a constant exponent of -6.

Type Name	Length	Type	Description
Mantissa	8	INT64	The fixed-point decimal representation of the price.
Exponent	N/A	INT8	MEMOIR uses a constant exponent of -6 for all prices.

For example, a mantissa of 1234567 with the constant exponent of -6 represents the decimal number 1.23456 and would appear encoded on the wire as the hex value `000000000012D687`.

3.1.2 String

Strings are fixed length ASCII based character sequences. Lengths are defined in the schema. Variable length fields are not supported.

3.1.3 BooleanType

Boolean values are not defined specifically in SBE. The schema defines a BooleanType which represents a numeric value with True = `1` and False = `0`.

3.1.4 UTC Timestamp Nanos

Fields with the `UTCTimestampNanos` type represents a timestamp in Coordinated Universal Time (UTC) which begins at the UNIX epoch (January 1, 1970 at 00:00:00 UTC).

UTCTimestampNanos has the time unit of nanoseconds and is encoded as follows:

Type Name	Length	Type	Description
Time	8	UINT64	UTC timestamp since unix epoch with nanosecond precision.
Unit	N/A	UINT8	Unit of time. UTCTimestampNanos are represented in nanoseconds. This field is constant (value=`9`) and as such will not be transferred on the wire.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

3.2 Header

SBE includes a header for each message. The SBE header is followed by the SBE body for the message.

The common SBE message header contains the fields that allows the decoder to identify what codec should be used as the template for a message.

1. **blockLength**: The number of bytes in the message body (does not include the header bytes).
2. **templateID**: The identifier for the template type of the message that is to follow.
3. **schemaID**: The identifier for the schema the message belongs to.
4. **version**: The version of the schema allowing for extension. MEMX packs two pieces of information into the (UINT16) version field, a major version and a minor update version. So, for example, a version of 258 = 0x0102 indicates major version 1, minor update 2.

The MEMOIR SBE header appears on the wire as:

Field	Offset	Length	Type	Description
BlockLength	0	2	UINT16	The number of bytes in the message body (does not include the header bytes). Note that MEMOIR messages do not use repeating groups or variable-length fields.
TemplateID	2	1	UINT8	Identifier of the message template (ie. the message type).
SchemaID	3	1	UINT8	The identifier of a message schema. SchemaId=2 for MEMOIR Depth.
Version	4	2	UINT16	The version number of the message schema that was used to encode a message. MEMX packs two pieces of information into the (UINT16) version field, a major version and a minor update version. So, for example, a version of 258 = 0x0102 indicates major version 1, minor update 2.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

4 Message Field Types

All messages are composed of fields. Each field has a type. This section defines the field types, their underlying on the wire type, acceptable values and a description of the type.

4.1 SideType

SideType describes the side the order is on: either bid (buy) or offer (sell).

SideType is a 1-byte CHAR value.

Value	Name
B	Buy
S	Sell/SellShort/SellShortExempt

Messages that contain a field of SideType include: Order Added

4.2 SecurityTradingStatusType

SecurityTradingStatusType represents the current security trading state of a symbol on the MEMX Exchange.

SecurityTradingStatusType is a 1-byte CHAR value.

Value	Name
H	Halted
P	Paused
Q	Quoting
T	Trading

Messages that contain a field of SecurityTradingStatusType include: Security Trading Status

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

4.3 SecurityTradingStatusReasonType

SecurityTradingStatusReasonType represents the reason for this security status.

SecurityTradingStatusReasonType is a 1-byte CHAR value.

Value	Name	Description
X	None	The security trading status does not apply (e.g the security is trading normally).
R	Regulatory	The security trading status originated from a regulator or the primary listing exchange for the security and was received via the SIP.
A	Administrative	The security trading status change originated from MEMX

Messages that contain a field of SecurityTradingStatusReasonType include: Security Trading Status

4.4 TradingSessionType

TradingSessionType represents the current trading session of the MEMX market.

TradingSessionType is a 1-byte CHAR value.

Value	Name	Description
1	Opening	Pre-Market session
2	Trading	Market session
3	PostTrading	Post-Market session
4	Closed	Market closed

Messages that contain a field of TradingSessionType include: Trading Session Status.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5 Messages

This section defines the messages that make up the protocol. For each message, it lists the fields in the message, as well as each field's position and length in the message, its underlying type, and a description of its purpose.

5.1 Instrument Directory

At the start of the trading day, an instrument directory message will be sent for all tradable securities on the MEMX exchange. In order to reduce the message sizes and improve performance, a SecurityID will be used in place of a ticker symbol on subsequent messages. SecurityIDs will start at 1 and increment. Do not rely on SecurityIDs being the same in subsequent trading days. MEMX uses CMS as its trading symbology, which requires a root and suffix combination.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=1, BlockLength=36
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	A unique code identifying the instrument for any messages in this session.
Symbol	16	6	String	The CMS root for the tradable instrument.
SymbolSfx	22	6	String	The CMS suffix for the tradable instrument.
RoundLot	28	4	UINT32	The number of shares in a round lot for the instrument.
Reserved	32	1	UINT8	Reserved for future use
IsTestSymbol	33	1	BooleanType	Determines if this security is a test instrument.
MPV	34	8	PriceType	The minimum price variation for an instrument. (the smallest price increment of the stock)

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.2 Reg SHO Restriction

In 2010, the Securities and Exchange Commission adopted Rule 201 of Regulation SHO. Rule 201 restricts the price at which short sales may occur when a stock has experienced significant downward price pressure. Rule 201 is designed to prevent short selling, including potentially manipulative or abusive short selling, from driving down further the price of a security that has already experienced a significant intra-day price decline, and to facilitate the ability of long sellers to sell first upon such a decline.

This message is used to indicate when a specific security is subject to the short sale price restriction. If this message is sent and the ShortSaleRestriction flag is set to true, Rule 201 is now in effect for that security ID.

NOTE: This message may be sent immediately after the related `Instrument Directory` message is sent for the security if the restriction is already in effect at the start of the MEMOIR data session.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=2, BlockLength=11
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
ShortSaleRestriction	16	1	BooleanType	Identifies if this security is subject to the Regulation SHO short sales restrictions.

5.3 Security Trading Status

This message will inform the client of the current trading status of a security on the MEMX Exchange. The trading status of a security on the MEMX exchange may change due to the following reasons:

- The primary market for a security has the authority to halt or pause a security via the SIP based upon upcoming news dissemination, or for regulatory purposes such as LULD threshold violations.
- MEMX may internally halt trading for a security for administrative or operational reasons.

Security Trading Status messages are sent out after Instrument Directory messages have been sent.

This message can and will be sent out throughout the trading session to indicate realtime changes in the security state.

If a `SecurityTradingStatus` message is not received for a security, it should be assumed that the status is `Halted`.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=3, BlockLength=12
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
SecurityTradingStatus	16	1	SecurityTradingStatusType	The current trading state.
SecurityTrading StatusReason	17	1	SecurityTrading StatusReasonType	The source of the trading status change.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.4 Trading Session Status

The MEMX trading system has entered a new trading session.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=5, BlockLength=9
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
TradingSession	14	1	TradingSessionType	The trading session which was entered.

5.5 Order Added

A visible order has been added to the MEMX book.

The associated OrderID on this message will be used as the key that may be used to track this order's lifecycle in subsequent messages.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=10, BlockLength=31
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
OrderID	16	8	UINT64	The trading session unique reference number assigned to the order.
Side	24	1	SideType	The side of the order being added.
Quantity	25	4	UINT32	The total quantity being added.
Price	29	8	PriceType	The displayed price of the new order.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.6 Order Deleted

A visible order was removed from the MEMX book. This can be due to a cancellation requested by the customer, or due to an administrative operation.

This OrderID will no longer be tracked, and should be removed from the book.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=11, BlockLength=18
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
OrderID	16	8	UINT64	The trading session unique reference number assigned to the order in the Order Added message.

5.7 Order Reduced

A visible order was reduced in quantity while maintaining its matching priority in the book. This can be due to a replace or partial cancel request received from the customer.

The quantity field must be subtracted from the existing order quantity, with the remainder remaining on the book. If the value resulting from the subtraction reduce the order quantity to zero the order should be removed from the book.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=12, BlockLength=22
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
OrderID	16	8	UINT64	The trading session unique reference number assigned to the order in the Order Added message.
Quantity	24	4	UINT32	The total quantity being reduced. This value will always be greater than 0.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.8 Order Executed

A visible order on the book was executed.

If the quantity executed reduces the remaining quantity of the order to 0, the order should be removed from the book. Otherwise, the remaining order quantity after the execution remains on the order book with the same time-priority.

The executed price given by this message may differ from the original displayed price of the order, due to system price improvement or price sliding.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=13, BlockLength=38
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
OrderID	16	8	UINT64	The trading session unique reference number assigned to the order in the Order Added message.
TradeID	24	8	UINT64	A trading session unique match number of the trade.
Quantity	32	4	UINT32	The total quantity executed.
Price	36	8	PriceType	The executed price of the order, it may not be the price that it was entered at due various trading methodologies.

5.9 Trade

A non-displayed order on the book was executed.

This message does not modify the state of the displayed order book, but does impact cumulative volume and average price calculations.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=14, BlockLength=30
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
TradeID	16	8	UINT64	A trading session unique match number of the trade.
Quantity	24	4	UINT32	The total quantity traded.
Price	28	8	PriceType	The price of the matched execution.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.10 Broken Trade

A MEMX operational or regulatory action has resulted in an execution being busted. Trades cannot be reinstated once they have been reported broken.

Broken trades should be removed from daily volume and average price calculations.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=15, BlockLength=30
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
TradeID	16	8	UINT64	The trading session unique match number assigned to the original trade by the Trade or Order Executed message.
OriginalQuantity	24	4	UINT32	The original quantity of the trade being broken.
OriginalPrice	28	8	PriceType	The original price of the trade being broken.

5.11 Corrected Trade

A MEMX operational or regulatory action has resulted in an execution being corrected, without busting the entire trade.

Corrected trades should result in the original volume and price being removed from daily volume and average price calculations and replaced by the new values given in this message.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=16, BlockLength=42
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.
TradeID	16	8	UINT64	The trading session unique match number assigned to the original trade by the Trade or Order Executed message.
OriginalQuantity	24	4	UINT32	The original quantity of the trade being corrected.
OriginalPrice	28	8	PriceType	The original price of the trade being corrected.
CorrectedQuantity	36	4	UINT32	The corrected quantity of the trade being corrected.
CorrectedPrice	40	8	PriceType	The new price of the trade being corrected.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

5.12 Clear Book

A MEMX operational or regulatory action has resulted in the book state being reset.

Consumers processing this message should remove all existing orders from the book for the security ID specified.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=18, BlockLength=10
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
SecurityID	14	2	UINT16	An instrument code uniquely identifying the security from the Instrument Directory.

5.13 Snapshot Complete

End of the snapshot message, all messages have been sent. Used only in Snapshot recovery (see Message/State Recovery Methods below)

Upon receipt of this message, clients should disconnect from the Depth of Book Snapshot feed and follow the recovery procedure to reconcile their state against the incremental Depth of Book feed.

Field	Offset	Length	Type	Meaning
SBE Header	0	6	N/A	SBE Header with SchemaId=2, TemplateId=100, BlockLength=16
Timestamp	6	8	UTCTimestampNanos	The timestamp when the event occurred.
AsOfSequenceNumber	14	8	UINT64	The sequence number on the real-time multicast channel that this snapshot is based upon. All messages buffered from the real-time feed that are less than or equal to this number should be discarded. All messages buffered after this number should be applied. (with appropriate gaps filled normally).

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

6 Message/State Recovery Methods

As MEMOIR Depth of Book is disseminated primarily over the MEMX-UDP transport protocol, consumers of the feed may periodically miss messages due to the unreliable nature of the UDP protocol. Two independent Multicast groups will be provided for standard A/B arbitration and quick recovery of messages dropped due to packet loss on a single channel. However, there are cases, such as a catastrophic system issue on the client's end, or the loss of the same message from both channels, where the client may need to request data to be re-sent from MEMX. For these situations, two modes of recovery are available for users. The first is a "gap fill" mechanism, used to recover small numbers of messages, typically for small episodes of packet loss that affect the same messages on both Multicast channels. The second is a "snapshot" mechanism, where the current business state of the feed is sent, followed by a "live" sequence number for the client to begin processing the real-time MEMX-UDP channels.

6.1 Gap Fill

A client detects missing messages over the MEMX-UDP transport for MEMOIR Depth of Book by using the sequence number in the MEMX-UDP datagram header to determine if a gap in sequence numbers has occurred. In order to recover these messages, the client may use a connection to a Gap Fill Server via the MEMX-TCP Replay mode to request the missing messages.

NOTE: The client may initiate and maintain a connection to the MEMX-TCP Gap Fill Server early (to verify connectivity) and stay connected (while periodically sending heartbeats) until the functionality is needed.

The process by which a client recovers missing messages via the Gap Fill Server is as follows:

1. Issue a MEMX-TCP `ReplayRequest` to the MEMOIR Depth of Book Gap Fill Server, containing the `SessionID` received over the MEMX-UDP channel(s), the `NextSequenceNumber` of the first missing message, and the `Count` of messages required (including the sequence provided)
2. While the above request is being processed, the client should buffer any received MEMX-UDP multicast messages and not apply them until the missing sequence numbers have been recovered.
3. After the receipt of the MEMX-TCP Replay Request, the Gap Fill Server will send the following:
    a. A MEMX-TCP `ReplayBegin` message, with `NextSequenceNumber` set to the requested Sequence number, and a `PendingMessageCount` set to the number of messages to follow in the replay. (This may be less than the requested count - see NOTE below)
    b. The requested number of **MEMOIR Depth Of Book** messages (or less given the constraints listed above on request size), starting at the requested sequence number
    c. A MEMX-TCP `ReplayComplete` message
4. After the `ReplayComplete` message is received, the client is free to send another request (if this request did not satisfy the entire gap)
5. Once the client has received all of the missing messages from one or more replay requests to the MEMX-TCP Replay server, the client may then process the messages buffered from the MEMX-UDP channel, discarding duplicate sequence numbers and applying future sequence numbers in order.

NOTE: the replay provided may consist of fewer messages than requested due to some constraints on the size of the response. The number of messages in the replay response will be the minimum of:

- the requested count
- the configured maximum messages per request for the service
- the number of messages remaining from the requested start sequence to the highest published sequence

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

6.2 Snapshot

If a catastrophic failure occurs on the client side that requires a full state reset on the client, and the client does not wish to manually replay the entire day via multiple MEMX-TCP Replay requests, the client may opt to use the MEMX-TCP snapshot mechanism to recover state. Once the snapshot has completed, the client should use normal Gap Fill processing (described above) to recover any further gaps in data that may occur during that trading session, as Snapshots can be quite large compared to a gap of one or two datagrams.

NOTE: The client may initiate and maintain a connection to the MEMX-TCP snapshot server early (to verify connectivity) and stay connected (while periodically sending heartbeats) until the functionality is needed.

The process by which a client recovers via snapshot is as follows:

1. Join the real-time MEMX-UDP multicast group(s) for the data feed, and begin buffering all received messages.
2. Issue a MEMX-TCP `ReplayAllRequest` to the MEMOIR Depth of Book snapshot server, containing the `SessionID` received over the MEMX-UDP channel(s)
3. After the receipt of the MEMX-TCP ReplayAll request, the Snapshot Server will send the following:
    a. A MEMX-TCP `ReplayBegin` message, with `NextSequenceNumber` set to 1 and `PendingMessageCount` set to the number of MEMOIR Depth of Book Business Level messages contained in the snapshot
    b. MEMOIR Depth of Book Business Level InstrumentDirectory messages containing a symbol description and associated SecurityID to be used throughout the session.
    c. MEMOIR Depth of Book Business Level `InstrumentDirectory` messages containing the current Reg-SHO Restriction State for each symbol
    d. MEMOIR Depth of Book Business Level `SecurityTradingStatus` messages containing the current trading status for each symbol
    e. MEMOIR Depth of Book Business Level `TradingSessionStatus` message containing the current trading session status.
    f. MEMOIR Depth of Book Business Level `AddOrder` messages for each symbol
    g. A single MEMOIR Depth of Book Business Level `SnapshotComplete` event containing an `AsOfSequenceNumber` set to the sequence number on the MEMX-UDP multicast channel that this snapshot was taken (clients should resume processing at this number + 1 on the multicast feed after applying the snapshot)
    h. A MEMX-TCP `ReplayComplete` message
4. Discard all buffered events from the MEMX-UDP real-time feed at or before the `AsOfSequenceNumber` in the received Snapshot Complete event.
5. Apply all buffered events from the MEMX-UDP real-time feed after the `AsOfSequenceNumber` in the received Snapshot Complete event, in sequence order.
6. Continue to process the MEMX-UDP real-time multicast feed normally from this point onward, using the **Gap Fill** mechanism above to recover dropped messages.

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

7 Message Examples

Note: the code fragments shown are representative of how to set field values in the SBE messages. Your interface may be be different.

7.1 Instrument Directory Message

```
instrumentDirectoryEncoder.timestamp().time(getTimeInNanos());
instrumentDirectoryEncoder.securityID(0xABCD);
instrumentDirectoryEncoder.symbol("AAPL");
instrumentDirectoryEncoder.roundLot(100);
instrumentDirectoryEncoder.reserved(InstrumentDirectoryEncoder.reservedNullValue());
instrumentDirectoryEncoder.isTestSymbol(BooleanType.False);
instrumentDirectoryEncoder.mPV().mantissa(10000);

Hex code:
00000000: 00240102 00010005 e175163d da53abcd   |.$.......u.=.S..|
00000010: 4141504c 00000000 00000000 00000064   |AAPL...........d|
00000020: ff000000 00000000 2710                |........'.|
```

7.2 Reg Sho Restriction Message

```
regShoRestrictionEncoder.timestamp().time(getTimeInNanos());
regShoRestrictionEncoder.securityID(0xABCD);
regShoRestrictionEncoder.shortSaleRestriction(BooleanType.True);

Hex code:
00000000: 000b0202 00010005 e17518f6 e552abcd   |.........u...R..|
00000010: 01                                    |.|
```

7.3 Security Trading Status Message

```
securityTradingStatusEncoder.timestamp().time(getTimeInNanos());
securityTradingStatusEncoder.securityID(0xABCD);
securityTradingStatusEncoder.securityTradingStatus(SecurityTradingStatusType.Quoting);
securityTradingStatusEncoder.securityTradingStatusReason(SecurityTradingStatusReasonType.
Regulatory);

Hex code:
00000000: 000c0302 00010005 e1751905 3967abcd   |.........u..9g..|
00000010: 5152                                  |QR|
```

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

7.4 Order Added Message

```
orderAddedEncoder.timestamp().time(getTimeInNanos());
orderAddedEncoder.securityID(0xABCD);
orderAddedEncoder.orderId(0x1122334455667788L);
orderAddedEncoder.side(SideType.Buy);
orderAddedEncoder.quantity(1500);
orderAddedEncoder.price().mantissa(123450000);

Hex code:
00000000: 001f0a02 00010005 e1751925 b0d3abcd   |.........u.%....|
00000010: 11223344 55667788 42000005 dc000000   |."3DUfw.B.......|
00000020: 00075bb2 90                           |..[..|
```

7.5 Order Deleted Message

```
orderDeletedEncoder.timestamp().time(getTimeInNanos());
orderDeletedEncoder.securityID(0xABCD);
orderDeletedEncoder.orderId(0x1122334455667788L);

Hex code:
00000000: 00120b02 00010005 e175193b 96d1abcd   |.........u.;....|
00000010: 11223344 55667788                     |."3DUfw.|
```

7.6 Order Reduced Message

```
orderReducedEncoder.timestamp().time(getTimeInNanos());
orderReducedEncoder.securityID(0xABCD);
orderReducedEncoder.orderId(0x1122334455667788L);
orderReducedEncoder.quantity(2200);

Hex code:
00000000: 00160c02 00010005 e175194d 01f8abcd   |.........u.M....|
00000010: 11223344 55667788 00000898            |."3DUfw.....|
```

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

7.7 Order Executed Message

```
orderExecutedEncoder.timestamp().time(getTimeInNanos());
orderExecutedEncoder.securityID(0xABCD);
orderExecutedEncoder.orderId(0x1122334455667788L);
orderExecutedEncoder.tradeId(0xFFEEDDCCBBAA9988L);
orderExecutedEncoder.quantity(2100);
orderExecutedEncoder.price().mantissa(123450000);

Hex code:
00000000: 00260d02 00010005 e175195c acbaabcd   |.&.......u.\....|
00000010: 11223344 55667788 ffeeddcc bbaa9988   |."3DUfw.........|
00000020: 00000834 00000000 075bb290            |...4.....[..|
```

7.8 Trade Message

```
tradeEncoder.timestamp().time(getTimeInNanos());
tradeEncoder.securityID(0xABCD);
tradeEncoder.tradeId(0X112233);
tradeEncoder.quantity(200);
tradeEncoder.price().mantissa(123450000);

Hex code:
00000000: 001e0e02 00010005 e175196c 177eabcd   |.........u.l.~..|
00000010: 00000000 00112233 000000c8 00000000   |......"3........|
00000020: 075bb290                              |.[..|
```

7.9 Broken Trade Message

```
brokenTradeEncoder.timestamp().time(getTimeInNanos());
brokenTradeEncoder.securityID(0xABCD);
brokenTradeEncoder.tradeId(0x11223344L);
brokenTradeEncoder.originalQuantity(400);
brokenTradeEncoder.originalPrice().mantissa(123450000);

Hex code:
00000000: 001e0f02 00010005 e175197a 4cdaabcd   |.........u.zL...|
00000010: 00000000 11223344 00000190 00000000   |......"3D.......|
00000020: 075bb290                              |.[..|
```

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.

7.10 Corrected Trade Message

```
correctedTradeEncoder.timestamp().time(getTimeInNanos());
correctedTradeEncoder.securityID(0xABCD);
correctedTradeEncoder.tradeId(0X112233);
correctedTradeEncoder.originalQuantity(200);
correctedTradeEncoder.originalPrice().mantissa(123450000);
correctedTradeEncoder.correctedQuantity(300);
correctedTradeEncoder.correctedPrice().mantissa(123470000);

Hex code:
00000000: 002a1002 00010005 e1751989 545cabcd   |.*.......u..T\..|
00000010: 00000000 00112233 000000c8 00000000   |......"3........|
00000020: 075bb290 0000012c 00000000 075c00b0   |.[.....,.....\..|
```

7.11 Clear Book Message

```
clearBookEncoder.timestamp().time(getTimeInNanos());
clearBookEncoder.securityID(0xABCD);

Hex code:
00000000: 000a1202 00010005 e1751996 dc46abcd   |.........u...F..|
```

7.12 Snapshot Complete Message

```
snapshotCompleteEncoder.timestamp().time(getTimeInNanos());
snapshotCompleteEncoder.asOfSequenceNumber(0x11223344L);

Hex code:
00000000: 00106402 00010005 e17519aa 085a0000   |..d......u...Z..|
00000010: 00001122 3344                         |..."3D|
```

COPYRIGHT MEMX LLC 2021. ALL RIGHTS RESERVED.