Xetra® Release 11.0

Enhanced Broadcast Solution - Interface Specification
Final Version

Table of Contents

1. Introduction

Xetra disseminates market information to members through a high performance, functionally enhanced broadcast interface known as the Xetra Enhanced Broadcast Solution. The Enhanced Broadcast Solution interface is a low latency solution with a highly granular dissemination model and closely resembles the FIX 5.1 (Financial Information eXchange) protocol. Members are able to effectively use the Enhanced Broadcast Solution interface to develop applications that fit their requirements.

This document is intended for system designers and programmers who wish to develop/adapt their client application to interact with the services offered by the Enhanced Broadcast Solution interface.

General Introduction

- Chapter 2 "Infrastructure Requirements" highlights the basic infrastructure requirements for accessing the services offered by the Enhanced Broadcast Solution interface.
- Chapter 3 "Overview" provides an overview of the Enhanced Broadcast Solution interface explaining some commonly used terms.

Enhanced Broadcast Solution Interface Structure

- Chapter 4 "Broadcast Streams" introduces the Enhanced Broadcast Solution interface streams and describes the interface data dissemination schedule during a Xetra business day.

Message Structures

- Chapter 5 "Structure of Messages" explains the Enhanced Broadcast Solution interface message structures in detail and describes the classification of the disseminated messages.

Appendices provide additional information on the Enhanced Broadcast Solution interface.

- Chapter 6 "Appendix - Glossary of Terms" provides an alphabetical listing of some commonly used terms.
- Chapter 7 "Appendix - Data Field Dictionary" contains the explanation of data fields, their formats and allowed value ranges.
- Chapter 8 "Appendix - Interface Limits" provides a list of system limits for the Enhanced Broadcast Solution interface.
- Chapter 9 "Appendix - How to Use" covers the description of the operation of an example client system using the Enhanced Broadcast Solution interface.
- Chapter 10 "Appendix - Message Encoding" describes the formats used for the encoding of data fields.
- Chapter 11 "Appendix - FIX mapping" presents the method for mapping messages provided by the Enhanced Broadcast Solution interface to FIX.
- Chapter 12 "Appendix - FAST Message Templates" includes a hard-copy of the FAST templates used for the Enhanced Broadcast Solution interface messages.

2. Infrastructure Requirements

2.1 Hardware and Network Infrastructure

The Enhanced Broadcast Solution interface disseminates market information over a multicast network set up by Xetra. A router capable of handling the IP multicast is required as the communication equipment for accessing the Enhanced Broadcast Solution interface. The multicast management protocol is the IGMPv2. Utilizing IGMPv3 it has to be considered that the IGMPv2 compatibility mode is enabled.

Members subscribing to the information can contact the Xetra Network Support to obtain further details on configuring their network equipment.

2.2 Software Infrastructure

Members need to have a standard FAST template based decoder in order to be able to use the Enhanced Broadcast Solution interface. Alternatively members may implement the FAST decoder by themselves.

Xetra will not provide any client software for accessing the services offered. Member systems for Enhanced Broadcast Solution can be based on any platform capable of receiving multicasts. Members will have the opportunity to develop and use specific applications that fit their requirements.

3. Overview

The Xetra Enhanced Broadcast Solution is a flexible, transparent, UDP based interface which disseminates market information to members over a multicast network. The information disseminated by the Enhanced Broadcast Solution interface includes orderbook, statistical and system status information.

The messaging protocol used by the Enhanced Broadcast Solution interface will closely resemble the Financial Information eXchange (FIX) protocol version 5.1 and will have message formats customized to fit the Xetra business requirements. The Enhanced Broadcast Solution interface will conform to the FIX Adapted for Streaming (FAST) protocol version 1.1 principles for efficient bandwidth utilization.

The interface provides members with the information in form of broadcast streams. From these broadcast streams, members will be able to receive the information that meets their requirements. The interface is designed so that members will only receive information from those streams they have joined.

The Enhanced Broadcast Solution interface will be supported in parallel to the existing public market data dissemination over VALUES.

3.1 Definition of Commonly Used Terms

Following are definitions of terms frequently used in the context of the Xetra Enhanced Broadcast Solution.

- Enhanced Broadcast Solution. The new broadcast concept bases on Multicast Infrastructure.
- Member/Exchange Participant. The term "Member" refers to an organization that is authorized by Deutsche Börse AG, Eurex Bonds or a Partner Exchange to receive the services offered through the Enhanced Broadcast Solution interface.
- Securities and instruments. "Security" and "Instrument" refer to an individual tradable component in Xetra.
- Broadcast. Broadcasts contain information about events affecting the market. The information comprised in these broadcasts is limited to public information.
- Broadcast groups. Certain functional instrument groups are aggregated to a single broadcast group. The instruments of this instrument group are sent over the same multicast addresses.
- Channels. Xetra offers multiple market depths for every instrument. Every channel offers a particular order book depth range that members can join.
- Streams. One or more tradable instruments on Xetra may be grouped together to form a logical group. Data pertaining to all instruments of such a group will be disseminated over one set of multicast addresses and constitute a stream. The system has five kinds of streams, the reference stream, the three available streams for market data (snapshot, delta/Incremental and All Trade Price) and the system state changes stream.
- Delta or Incremental Broadcasts. They are provided for a market-affecting event (such as order entry or a trade), only changes to the market are broadcast through this stream.

- Snapshot broadcasts. A snapshot broadcast contains all current market information for a given instrument, regardless of recent movements and will be sent on a different stream in addition to the “real time” delta broadcasts.

  Snapshot broadcasts will be sent less frequently than the delta broadcasts. Members should join this stream just long enough to receive one snapshot enabling them to establish a baseline market picture. After establishing the baseline market picture, members should leave the snapshot broadcasts stream and rely on delta broadcasts to maintain it. Staying joined to the snapshot stream consumes larger bandwidth but gives no additional information. The frequency of dissemination is calculated from the network load condition in real time.

  The expected member application behaviour is explained in chapter 9.Appendix - How to Use.

- All Trade Price broadcasts. A Xetra All Trade Price broadcast contains the Xetra All Trade Price information. All Trade Price messages will be disseminated over the All Trade Price stream.

- State changes broadcast. These broadcasts inform the receiver of any exchange or system state change.

- Joining. Joining a stream means electronically connecting to the information source. Once joined to a source, all information that is disseminated by the source will be received involuntarily and will continue until the stream is quitted.

- Financial Information eXchange (FIX) protocol. The FIX Protocol is a messaging standard developed specifically for the real-time electronic exchange of securities transactions. FIX is a public domain specification owned and maintained by FIX Protocol Ltd.

- FIX Adapted for STreamingSM (FAST Protocol SM). FAST is a standard developed by Data Representation and Transport Subgroup of FPL's Market Data Optimization Working Group. FAST uses proven data compression techniques that leverages knowledge about data content and data formats.

- Recoverable Information. Recoverability of information in the context of this document is availability of the most recent value of a field on the snapshot stream in form of order book snapshots. The Xetra Enhanced Broadcast Solution does not provide retransmission/query services.

- Unrecoverable Information. For information sent as market event the most recent value is available through the interface in form of delta/incremental messages since the snapshot messages are always sent a little after. If the packet carrying the delta information does not reach the member system, the information should be considered as lost. In most cases though, the member will be able to detect the loss of such information as such information is sequenced and then, recover it using a snapshot.

- Order Book (ODB). A digital book of orders for a tradable unit (instrument) available for matching and maintained by Xetra.

For a more detailed list of definitions, see table in section 6.Appendix - Glossary of Terms.

3.2 Disseminated Market Information

Reference Information

Instrument reference data information messages provide a description of all the available instruments for which the Enhanced Broadcast Solution interface will broadcast price updates. Instrument reference data information will be available from a pre-defined multicast address and ports and contains information on the granularity of the data offered for each instrument and the corresponding multicast addresses and ports for the channels.

Maintenance reference data messages provide the multicast address of the state changes stream.

Order Book and Statistical Information

The current state of the order book is distributed using the order book information messages. Members have to build and maintain their own picture of the order book from these messages.

- Snapshot. Snapshots contain complete orderbook information up to the depth indicated in the reference information. The snapshot message should be used only for the creation of the market picture at the beginning of a trading day and for its recovery in case of a data loss.

  Snapshots provide information about the actual instrument status, the details of the last trade and the days' statistical information for the instrument.

- Delta or Incremental. Deltas should be interpreted as commands issued by the exchange. A member has to alter/update their copy of the order book for each instrument based on the delta messages received.

  Under normal conditions members will be able to maintain their copy of the order book by joining the delta broadcast stream and applying the received messages.

More information about maintaining the global market picture can be found in chapter 9.Appendix - How to Use.

Trade Information

Trade Price Information. The Xetra All Trade Price broadcast stream contains the Xetra trade price information excluding OTC trade prices. All Trade Price messages will be disseminated over the All Trade Price stream.

Status Information

System and Exchange state change. This stream will inform the members of a system or exchange state change.

State change messages will not be sent per instrument. The instrument status changes throughout the day are received via the snapshot and the delta stream.

3.2.1 Exceptions

Listed below are some rules or exceptions pertaining to the data content published by the Enhanced Broadcast Solution interface.

- Order book rules for auctions: Order book information disseminated during auctions depends on the instrument set-up. Please find more details in the corresponding Xetra market model documents.

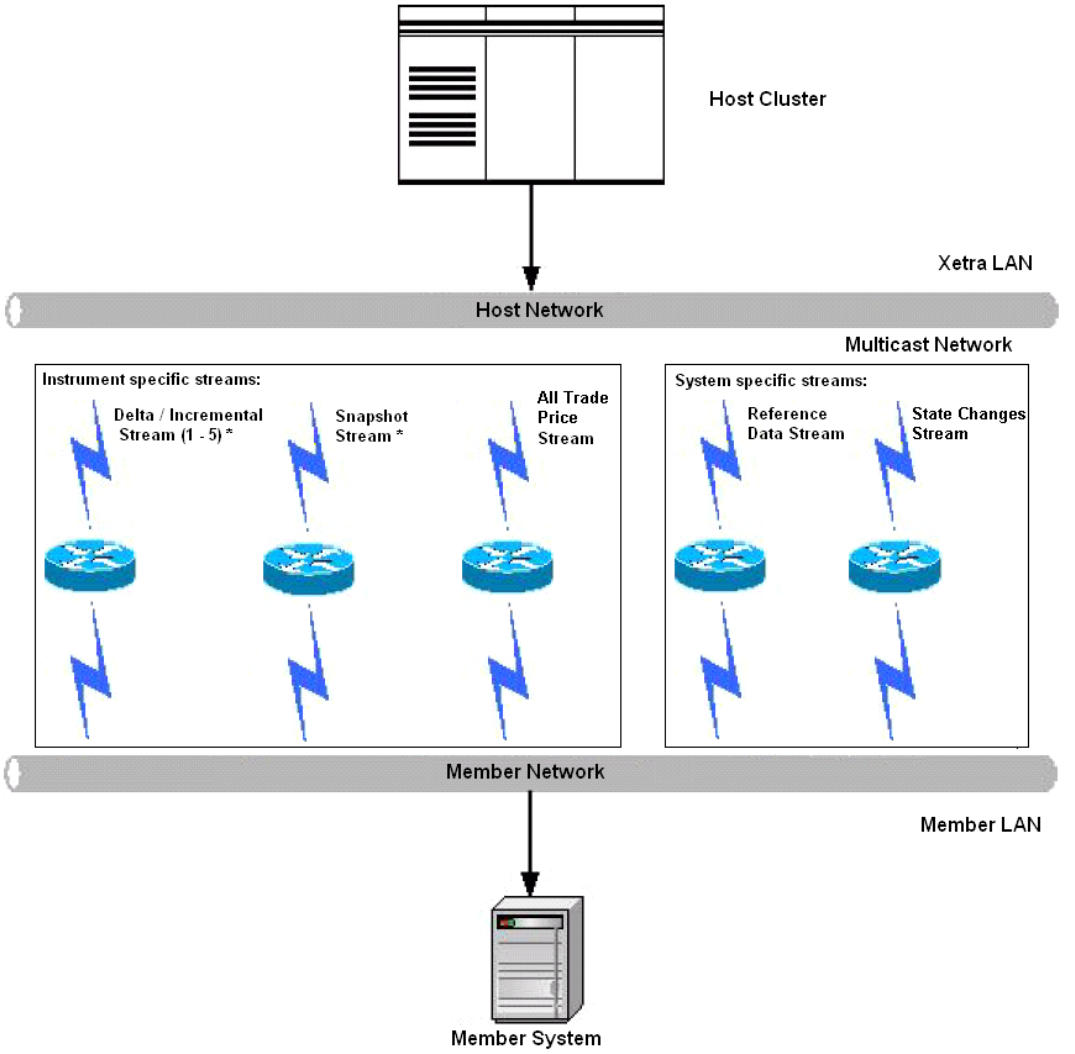
- Gap indication: The Enhanced Broadcast Solution interface will normally disseminate un-netted price information. However, in exceptional situations, e.g., due to heavy processing load some of the price level changes may not be sent. For these situations a gap indicator will be sent in the next message.

4. Broadcast Streams

The Xetra Enhanced Broadcast Solution interface will disseminate information in five kinds of streams in a live-live concept meaning that the data will be disseminated in separate streams over two services called service A and service B.

These five streams are:

- Reference Data. This stream is the first one to be connected to since it disseminates the addresses for all the other streams.
- Snapshot
- Delta / Incremental
- All Trade Prices
- State changes



Due to the inherently unreliable nature of the UDP protocol, data packets may be lost. Members are advised to join both services (A and B) to reduce the probability of data loss. Every stream will consist of one or more channels and will follow certain rules for dissemination of data.

4.1 Reference Data Stream

Member applications have to join the reference data stream at the start of each business day. To make the bandwidth available for other streams this stream should be unsubscribed once all the reference data has been received. Each market source has its own reference data stream.

The reference data stream disseminates two types of information, instrument reference data and maintenance reference data. Both provide reference information for configuration data. This data is disseminated the whole day cyclically. The reference data is deemed as static for a business day.

The instrument reference information provided by this stream contains the multicast channel information (i.e. multicast addresses and ports) of the snapshot, delta and All Trade Price streams. The maintenance reference data provided by this stream contains the multicast channel information (i.e. multicast addresses and ports) of the state changes stream. Members must, therefore, join these streams and receive all necessary information before being able to join the state changes stream.

4.2 Snapshot Stream

Members should join this stream to receive complete order book snapshots. The information provided by this stream needs to be used for creating the base order book structure. After snapshots for all instruments have been received, members should quit this stream. This stream can also be used for recovery purposes in case of a data loss.

Order book snapshots carry consolidated sequence numbers for all the incremental/delta channels, providing the receiver with sufficient information to position the snapshots exactly amongst the delta messages.

Snapshot information will be available for all instruments throughout the business day but should be joined only for starting, unless there is a data loss.

4.3 Delta/Incremental Stream

Members should remain connected to this stream throughout the trading day for receiving delta messages. All incoming delta messages should be applied to the copy of the order book maintained by the member applications in order to have the latest information.

4.4 All Trade Price Stream

The Xetra All Trade Price stream is the Xetra public broadcast data stream for dissemination of information about every single trade.

Members should remain connected to this stream throughout the trading day for receiving All Trades Prices messages.

4.5 State Changes Stream

The state change messages contain the system and exchanges states. The current system and exchange states are sent cyclically and event-driven. An event occurs as soon as an exchange or system change happens.

Members should remain connected to this stream throughout the trading day for receiving the system and exchange State Change messages.

4.6 Interface Schedule

The Enhanced Broadcast Solution interface will be based on a “push only” architecture and the data will be disseminated in streams consisting of one or more multicast channels. Member applications will have to join a particular multicast address and port combination for connecting to a stream. Whenever the interface sends data over a stream, all client applications joined to it will receive these data packets.

Please note that the information regarding streams and channels which member applications have to join for receiving market information, will be provided by the reference data stream.

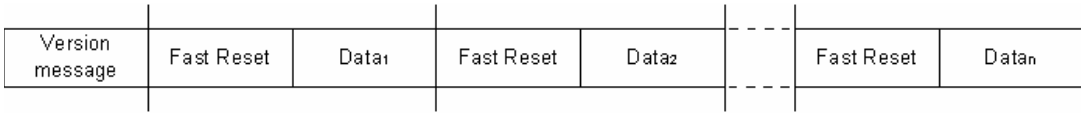
A mapping between the trading phases and the functional timeline of the Enhanced Broadcast Solution Interface is shown below:

Xetra Exchange Period	Xetra Enhanced Broadcast Solution	Member System
Start of business day	Start Xetra Enhanced Broadcast Solution	
	Issue start of service message. Start the instrument reference data and maintenance reference data stream.	Start of Trading day. Join the reference data stream and store the received instrument and maintenance reference data.
	Start snapshot stream	Join the snapshot stream and build the base order book for all instruments.
	Start Delta/Incremental stream	Join the Delta/Incremental channel[1]. Leave the snapshot stream and apply incoming deltas to the order book. Capture the non-incremental messages.
	Start All Trade Price stream	Join the All Trade price stream to receive the trades
	Start State Changes stream	Join the State Changes stream to receive the system and exchange state changes
Trading Phases: Pretrade Pre-open	Continue broadcasting the information...	Receive the disseminated information.
Post end-of-day-processing	Issue end of service message. Service is brought down	Leave the streams. End of trading day

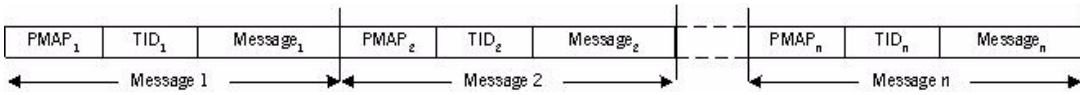
[1] If any gaps are detected in the sequence numbers on the delta stream, member applications can either connect to the snapshot stream and receive the latest order book snapshot or wait for the next delta message for this instrument to rebuild their copy of the order book. Member applications must be prepared for the fact that multicast does not guarantee the correct sequence of messages.

5. Structure of Messages

The Xetra Enhanced Broadcast Solution disseminates data in UDP datagrams. Every UDP datagram sent through the Enhanced Broadcast Solution will have the structure described in the diagram below:



Data consists of following blocks:



The UDP protocol adds at least a 28 byte header to every packet (20 byte IP header plus 8 byte UDP protocol header).

Every UDP datagram will be complete, i.e. there will be absolutely no dependency across datagrams. This is required to compensate for the unreliable nature of the UDP protocol.

All messages, all field data types and encoding formats disseminated by the Enhanced Broadcast Solution interface conform to the FAST standard 1.1.

Regarding FAST:

- Enhanced Broadcast Solution uses global dictionary scope for FAST operators. All the operators share the same dictionary regardless of the template and application type.
- The FAST reset message is inserted at the start of every datagram in order to explicitly reset all the dictionaries.

Presence Map (PMAP)

The presence map is a bit combination indicating the presence or absence of a field in the message body. The allocation of a bit for a field in the presence map is governed by the FAST field encoding rules[2].

If the presence map bit is set, it indicates that the field is sent within the message and if not, it indicates that the field is omitted.

If a certain field does not need a presence map bit, the table value will be "N.A." (non-applicable).

When dealing with repeating groups, the following should be taken into account:

- Repeating groups will have no presence map. The number of entries will always be sent, either with the actual number of entries or set to zero if there is no data in the repeating group to be sent.
- The FAST operator will be "none" for fields inside repeating groups that are not “delta”, since any other value will imply the need of a presence map.

[2] see section 10.5 of the FAST specification document at http://www.fixprotocol.org/documents/3066/FAST%20Specification%201%20x%201.pdf

Template Identifier (TID)

The template identifier is an integer field to inform which template should be used for decoding the message.

The following table lists the message types and their corresponding template identifiers. The template ids in the list are not sequenced, there are a couple of gaps due to the already existing templates in Eurex and to FAST rules as to template numbering[3].

Message	Template ID
Version Information Message	1
Beacon Message	2
Instrument Reference Data	3
Maintenance Reference Data	4
Snapshot Message	6
Delta/Incremental Message	7
All Trade Price Message	9
State Changes Message	10
FAST Reset Message[4]	120
Start of Service Message	128
End of Service Message	129
Start of Instrument Reference Data Message	130
End of Instrument Reference Data Message	131
Start of Maintenance Reference Data Message	132
End of Maintenance Reference Data Message	133

[4]Standard FAST template:
<template name="Reset" scp:reset="yes" id="120"><typeRef name="Reset"/></template>

Message (Body)

Messages are logically divided into service messages, that do not contain any market information and market data messages.

Some special comments that should not be forgotten:

[3]To learn more about the template identifiers rules, read appendix 2.2 of the following FAST document: http://www.fixprotocol.org/documents/3795/FAST%20Session%20Control%20Protocol%201.1.doc

- Timestamps contain the time since midnight in Central European Time (CET/CEST) and will be formatted in milliseconds, unless otherwise specified. Member systems can use these timestamps and source identifiers along with the sequence numbers (wherever sent) to order messages.
- The Xetra Enhanced Broadcast Solution maintains the sequence numbers per instrument per source.
- Price and quantity fields are sent as scaled numbers with a mantissa of 8 bytes and an exponent of 4 bytes.

5.1 Structure of Service Messages

Service messages do not carry any market information. These messages are sent for the purpose of synchronization or to indicate the status of the service.

5.1.1 Version Information Message

The Xetra Enhanced Broadcast Solution version information message will be sent in every UDP datagram. Every UDP datagram will be tagged with the service version number to allow multiple services to run in parallel over the same set of systems and network.

VERSION INFORMATION MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
versNo	Enhanced Broadcast Solution service version number	N.A.	Mandatory	none	Version number type (7.34)
srcId	Source identifier	N.A.	Mandatory	none	Source ID type (7.24)
seqNum	Sequence number of the datagram	N.A.	Mandatory	none	Sequence number type (7.22)

Every data disseminating source stamps outgoing datagrams with a sequence number, field seqNum. This sequence number along with the source identifier (field srcId) can be used to detect duplicates or a loss of a datagram.

Please note that this datagram sequence number cannot be used to detect data loss on the application level. The receiver application has to use the sequence numbers inside the messages for this purpose.

The field versNo is a bit encoded field and carries details to identify the service to which the datagram belongs. The bit significance is as follows:

Bit Name	Field Size in Bits	Bit Description
reserved	1	unused
exchangeCode	5	Environment Number: ‘X’ - Xetra. This field is ITA2 encoded. ‘X’ will be sent as 0x1D
swVersion	10	Exchange software release number. For example, value 110 Xetra release 11.0
subVersion	4	Identifies different services within a Xetra release. Initial value will be 0.
areaType	5	Area type: ‘P’ - Production ‘S’ - (Advanced) Simulation. This field is ITA2 encoded. ‘P’ and ‘S’ would be sent as 0x016 and 0x05 respectively.
envNo	7	Environment Number: 51 – Production 52 – Simulation 53 – Advanced Simulation 55 – Irish Stock Exchange Production 57 – Xetra International Market Production 68 – Vienna Production 69 – Vienna Simulation

5.1.2 Functional Beacon Message

Functional beacon messages indicate the availability of exchange services. They contain the exchange timestamp and the last sequence number sent for the channel and source. These messages are sent for the delta stream.

BEACON MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)

BEACON MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
seqNum	Sequence number	N.A.	Mandatory	none	Sequence number type (7.22)
isix	Instrument identifier	N.A.	Mandatory	none	Isix type (7.15)

5.1.3 Technical Beacon Message

Technical beacon message is sent out periodically on every multicast address. A cycle duration does not exceed 120 seconds. The technical beacon message consists of the Version Information Message (TID=1).

The presence of the technical beacon message does not assure that the prices are being sent using the defined host, the intention is to keep the PIM sparse routing trees alive.

The technical beacon message should only be used as an indication that the respective Xetra broadcast host is available.

5.1.4 Start/End of Services

The start and end of service messages are sent by the reference data stream and can be used by member systems to synchronize their services with that of the interface.

START/END OF SERVICE MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	none	Timestamp type (7.31)
busDate	Business date	N.A.	Mandatory	none	Business date type (7.3)

Start of service messages are generally only sent once during a business day. For technical reasons, they may occasionally occur more than once per day, and client applications must be able to handle this by simply ignoring such messages.

5.1.5 Start/End of Instrument/Maintenance Reference Data Information

The start/end of instrument and maintenance reference information will be disseminated by the instrument and maintenance reference data stream to indicate the start and end of the reference information.

START/END OF ALL REFERENCE MESSAGES STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	none	Timestamp type (7.31)
busDate	Business date	N.A.	Mandatory	none	Business date type (7.3)
noOfMsgs	Number of messages	N.A.	Mandatory	none	Number of values type (7.18)

5.2 Structure of Data Messages

5.2.1 Reference Information

The reference data stream disseminates instrument and maintenance data reference information.

There is no need to send the whole tick table since it is being sent with VALUES or ETS, only the smallest tick size will be sent. It indicates the maximum number of decimal places for each instrument.

The field instMnem is only filled, if an instrument has the Instrument Mnemonic. In other cases the field will contain a single space.

5.2.1.1 Instrument Reference Data

Will be disseminated over the static reference interface.

INSTRUMENT REFERENCE DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)

INSTRUMENT REFERENCE DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
seqNum	Sequence number	3	Mandatory	increment	Sequence number type (7.22)
isix	Instrument identifier	N.A.	Mandatory	delta	Isix type (7.15)
isin	Instrument ISIN code	N.A.	Mandatory	delta	ISIN code type (7.14)
instMnem	Instrument mnemonic	N.A.	Mandatory	none	Mnemonic type (7.17)
exchId	Exchange identifier	4	Mandatory	copy	Exchange ID type (7.8)
instGrp	Instrument group	5	Mandatory	copy	Instrument group type (7.11)
instTypCod	Instrument type	6	Mandatory	copy	Instrument Type (7.13)
currCode	Currency type code	7	Mandatory	copy	Currency code type (7.5)
ticSiz	Tick size	N.A.	Mandatory	delta	General price type (7.10)
setId	Set identifier	8	Mandatory	copy	Set Id type (7.23)
noOf-Streams	Number of streams	N.A.	Mandatory	none	Number of values type (7.18)
>stream-Type	Type of stream	N.A.	Mandatory	none	Stream type (7.28)
>stream-Service	Service class of the stream	N.A.	Mandatory	none	Stream service type (7.27)
>inetAddr	Multicast address of the stream	N.A.	Mandatory	delta	Address type (7.1)
>port	Port number	N.A.	Mandatory	delta	Port type (7.19)
>mktDepth	Stream depth	N.A.	Optional	none	Stream depth type (7.26)
>mdBook-Type	Book type	N.A.	Optional	none	Book type (7.2)

For more information on the FAST implementation, refer to the XML template in section 12.

5.2.1.2 Maintenance Reference Data

Maintenance reference data messages provide the multicast address of the state changes stream.

Will be disseminated over the static reference interface.

MAINTENANCE REFERENCE DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)
seqNum	Sequence number	3	Mandatory	increment	Sequence number type (7.22)
exchId	Exchange identifier	4	Mandatory	copy	Exchange ID type (7.8)
noOf-Streams	Number of streams=2	N.A.	Mandatory	none	Number of values type (7.18)
>stream-Service	Service class of the stream	N.A.	Mandatory	none	Stream service type (7.25)
>inetAddr	Multicast address of the stream	N.A.	Mandatory	delta	Address type (7.1)
>port	Port number	N.A.	Mandatory	delta	Port type (7.19)

For more information on the FAST implementation, refer to the XML template in section 12.

5.2.2 Order Book Information

5.2.2.1 Order Book Snapshot

The snapshot messages will be disseminated over the snapshot stream.

The following principles are valid for the design of the snapshot messages.

- Snapshot messages include top-of-book and complete price-level information (depth ≥ 1).
- Orders are aggregated per price level and not distributed individually.

- Snapshot messages provide complete information about **only one** instrument up to a specific level.
- Orderbook information is combined with statistical and trade information in a single message.
- Price levels are provided explicitly (by a number) and do not need to be inferred through the price itself.
- During the pre-trading and post-trading phases when no price levels exist for an instrument, an empty book will be disseminated.

The snapshot data has five different groups. The number of entries of the repeating group will always be filled, if there is no information for that group, the number will be set to 0.

- Potential Auction Price. "PrcQty" group. Identical in structure to the delta message "PrcQty" group. When a potential auction price is generated, this group will be sent with the corresponding price and quantity.
- Statistics. "Prc" group. Identical in structure to the delta message "Prc" group. This repeating group will be filled whenever there is a new daily high, daily low, opening price, closing price, valuation price.
- Surplus. "Qty" group. Identical in structure to the delta message "Qty" group. This repeating group will be filled when there is a surplus, either bid or ask side. Will be filled during Order Book Balancing of an auction.
- Trade information. "All Trade Price" group. This group will be filled when there is any information on last trade price, crossed trade price or last auction price, last midpoint trade, last best price.
- Orderbook Data. "Depth" group. When orderbook entries are sent, the field *noEntries-Depth* will be set to the number of entries being sent, *entryType* will be equal to "BID_PRC" or "ASK_PRC" and all the fields in the repeating group will be filled. Market Orders have *entryType* = ASK_PRC_MARKET / BID_PRC_MARKET and a price = 0.
- When mapping Enhanced Broadcast Solution fields to FIX, these five parts can be combined to one repeating group (in order to be FIX compliant). This single repeating group will then have a varying number of fields depending on whether the data is some statistical information, an indication of a surplus, price information, an orderbook entry or some trade information. The distinction into five parts has been chosen for clarity and efficiency.

The total number of snapshot data entries in a message is the sum of the number of entries in each of the following five repeating groups, price-quantity (noEntriesPrcQty), price (noEntriesPrc), quantity (noEntriesQty), All Trade Price (noEntriesAtp) and orderbook (noEntries-Depth).

Every snapshot contains the sequence numbers for the last "consolidated" delta message for this instrument in order to be able to position it among the delta messages. Members are expected to remain connected to the delta channels while receiving the snapshot for an instrument and buffer all the deltas until the complete snapshot is received. After the snapshot is received, these deltas have to be applied to the snapshot to obtain the most recent picture of the market.

Each snapshot message has a reference to the last All Trade Price sequence number for that instrument represented by the field *lastTpSeqNum*. This field is set to 0 for instruments in which there have been no trades.

For more information on the FAST implementation, refer to the XML template in section 12.

SNAPSHOT DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)
isix	Instrument identifier	N.A.	Mandatory	delta	Isix type (7.15)
noOfSeqNum	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>consolSeqNum	Sequence number	N.A.	Mandatory	delta	Sequence number type (7.22)
lastTpSeqNum	Sequence Number	N.A.	Mandatory	delta	Sequence number type (7.22)
instrStatus	Instrument status	N.A.	Mandatory	none	Instrument status type (7.12)
moiInd	Moi indicator	N.A.	Optional	none	Moi indicator type (7.16)
volInd	Vola indicator	N.A.	Optional	none	Vola indicator type (7.35)
cmexInd	Capital adjustment indicator	N.A.	Optional	none	Cmex indicator type (7.4)
noEntriesPrcQty	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed values are: POTENTIAL AUCTION PRICE, MATCHING RANGE BID/ASK (7.7)

SNAPSHOT DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
>entryPrc	Potential auction price	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Potential auction quantity	N.A.	Mandatory	delta	Quantity type (7.21)
noEntriesPrc	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry Type. Allowed values are: OPENING, CLOSING, VALU-ATION, HIGH, LOW(7.7)
>entryPrc	Price	N.A.	Mandatory	delta	General price type (7.10)
noEntriesQty	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed value is: SURPLUS (7.7)
>entryQty	Surplus ask/ bid quantity	N.A.	Mandatory	delta	Quantity type (7.21)
noEntriesAtp	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed values are: LAST AUC-TION, LAST TRADE, CROSSED TRADE, LAST MIDPOINT, LAST BEST. (7.7)
>entryPrc	Traded price	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Traded quantity	N.A.	Mandatory	delta	Quantity type (7.21)
>totTrdQty	Total traded quantity	N.A.	Mandatory	delta	Quantity type (7.21)

SNAPSHOT DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
>entryTime	Time of entry	N.A.	Mandatory	delta	Time type (7.30)
>numTrades	Number of trades	N.A.	Mandatory	delta	Number of values type (7.18)
noEntries-Depth	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed values are: BID, ASK, ASK PRC MARKET, BID PRC MARKET, EMPTY BOOK (7.7)
>entryPrc	Price for the level below	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Quantity offered at above price	N.A.	Mandatory	delta	Quantity type (7.21)
>numOrders	Number of orders	N.A.	Mandatory	delta	Number of values type (7.18)
>entryPrcLvl	Level of price in ODB	N.A.	Mandatory	none	Entry level type (7.6)

5.2.2.2 Order Book Delta/Incremental

The order book delta messages will be disseminated over the delta stream.

The following principles are valid for the design of the delta messages.

- Delta/incremental messages include partial price-level information (depth ≥ 1).
- Orders are aggregated per price level and not distributed individually.
- Delta/incremental messages provide complete information about only one instrument for one or more levels.
- Delta messages carry statistical and orderbook information, no trade information.
- Price levels are provided explicitly (by a number) and do not need to be inferred through the price itself.
- The order book has a valid state, after all instructions of the repeating group *Entries-Depth* has been processed.

- Every delta message has a sequence number for synchronization purposes.
- Each delta message has a reference to the last All Trade Price sequence number for that instrument. If the field *lastTpSeqNum* set to 0 for an instrument, it implies that there have been no trades in this instrument.
- During the pre-trading and post-trading phases when no price levels exist for an instrument, an empty book will be disseminated.

For more information on the FAST implementation, refer to the XML template in section 12.

The delta data has four different groups. The number of entries of the repeating group will always be filled, if there is no new information for that group, the number will be set to 0.

- Orderbook Data. "Depth" group. Similar in structure to the snapshot message orderbook group, only difference being the *updateAction* field needed to know in what way to update the orderbook. When orderbook entries are sent, the field *noEntriesDepth* will be set to the number of entries being sent, *entryType* will be equal to "BID_PRC" or "ASK_PRC" and all the fields in the repeating group will be filled. Market Orders have *entryType* = ASK_PRC_MARKET / BID_PRC_MARKET and a price = 0.
- Surplus. "Qty" group. Identical in structure to the snapshot message "Qty" group. This repeating group will be filled when there is a surplus, either bid or ask. Furthermore this group contains total traded quantity.
- Statistics. "Prc" group. Identical in structure to the snapshot message "Prc" group. This repeating group will be filled whenever there is a new daily high, daily low, opening price, closing price, valuation price, last trade price and last auction price.
- Potential Auction Price. "PrcQty" group. Identical in structure to the snapshot message "PrcQty" group. When a potential auction price is generated, this group will be sent with the corresponding price and quantity.

When mapping Enhanced Broadcast Solution fields to FIX, these four parts can be combined to one repeating group (in order to be FIX compliant). This single repeating group will then have a varying number of fields depending on whether the data is some statistical information, an indication of a surplus, price information, an orderbook entry or some trade information. The distinct four parts has been chosen for clarity and efficiency.

The total number of delta data entries in a message is the sum of the number of entries in each of the following four repeating groups, price-quantity (noEntriesPrcQty), price (noEntriesPrc), quantity (noEntriesQty) and orderbook (noEntriesDepth).

The following values are valid for the *updateAction* field:

- New: Creating a price level, adds the new price at the specified entryPrcLvl say x. All price levels y where y >= x are shifted to y + 1.
- Change: Changing a price level, replaces the quantity of the price level specified by the entryPrcLvl with the information sent in the message.
- Delete: Deleting a price level, removes the price at the level specified by entryPrcLvl say x. All price levels y where y > x are shifted to y - 1.
- Delete From: Deletes all price levels from entryPrcLvl >= x to maximum price levels maintained in the order book for the instrument.

- Delete Thru: Deletes all price levels from price level = ‘1’ to entryPrcLvl = ‘x’. All price levels y where y > x are shifted to y - x.
- For all delete actions (delete, delete from, delete thru) the fields entryPrc, entryQty and numOrders have to be ignored in the repeating group noEntriesDepth.

DELTA/INCREMENTAL DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)
isix	Instrument identifier	N.A.	Mandatory	delta	Isix type (7.15)
SeqNum	Sequence Number	N.A.	Mandatory	delta	Sequence number type (7.22)
lastTpSeqNum	Sequence Number	N.A.	Mandatory	delta	
instrStatus	Instrument status	N.A.	Mandatory	none	Instrument status type (7.12)
moiInd	Moi indicator	N.A.	Optional	none	Moi indicator type (7.16)
volInd	Vola indicator	N.A.	Optional	none	Vola indicator type (7.35)
cmexInd	Capital adjustment indicator	N.A.	Optional	none	Cmex indicator type (7.4)
gapIndicator	flag to indicate gap (some prior trades may not be reported)	3	Optional	constant	Gap indicator type (7.9)
noEntriesPrcQty	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)

DELTA/INCREMENTAL DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed values are: POTENTIAL AUCTION PRICE, MATCHING RANGE BID/ASK (7.7)
>entryPrc	Potential auction price	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Potential auction quantity	N.A.	Mandatory	delta	Quantity type (7.21)
noEntriesPrc	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry Type. Allowed values are: OPENING, CLOSING, VALU-ATION, HIGH, LOW, LAST TRADE PRICE, LAST AUCTION PRICE (7.7)
>entryPrc	Price	N.A.	Mandatory	delta	General price type (7.10)
noEntriesQty	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed value is: SURPLUS, TOTAL TRADED QUANTITY (7.7)
>entryQty	Surplus ask/ bid quantity	N.A.	Mandatory	delta	Quantity type (7.21)
noEntries-Depth	Number of entries	N.A.	Mandatory	none	Number of values type (7.18)

DELTA/INCREMENTAL DATA MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
>entryType	Type of entry	N.A.	Mandatory	none	Entry type. Allowed values are: BID, ASK, ASK PRC MARKET, BID PRC MARKET, EMPTY BOOK (7.7)
>entryPrc	Price for the level below	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Quantity offered at above price	N.A.	Mandatory	delta	Quantity type (7.21)
>numOrders	Number of orders	N.A.	Mandatory	delta	Number of values type (7.18)
>entryPrcLvl	Level of price in ODB	N.A.	Mandatory	none	Entry level type (7.6)
>updateAction	What operation will be performed with this entry	N.A.	Mandatory	none	Price action type (7.20)

Xetra works on the basis of “units of work”. A single unit of work may consist of multiple trades, each of which generates a discrete trade message (All Trade Price). Each unit of work always delivers only a single delta message, which includes a summary position of all trades occurring within that unit of work. The summary position contains of the fields total traded quantity and last trade price. Total traded quantity is the cumulated quantity of the instrument for the current business day. The last trade price refers to the last price within the unit of work.

The trade summary position is mapped to the repeating groups EntriesQty (total traded quantity) and EntriesPrc (last trade price). For further information, please refer to section 11.2.2.2.

Each incoming order or quote results in one unit of work during Continuous Trading. Each Exchange State change results in one single unit of work.

Members requiring a full view of all reported trades must therefore use the All Trade Price messages, not the delta messages for this purpose.

The delta message sequence numbers, field *seqNum* increments sequentially only within the context of a single source identifier. Therefore, this field can only be used when comparing messages coming from the same source.

lastTpSeqNum indicates the highest trade price sequence number included in the order book resulting from this delta message.

Although trade price sequence numbers are assigned to trade prices sequentially, there is not necessarily a one-to-one correlation between trade price events and deltas. A single unit of

work on the host may result in many trades (which are reported individually) but only a single orderbook delta, delivered at the end of the unit of work.

This field should therefore not be expected to increment sequentially.

For more information on sequence numbers, see section 9.2.

Orderbook "Trimming"

Enhanced Broadcast solution does not send explicit delete messages beyond mktDepth. Client applications must, however, allow for temporary growth beyond mktDepth until all actions of a message are processed.

5.2.3 Trade Information

5.2.3.1 All Trade Price Message

The Xetra All Trade Price subscription is the Xetra public broadcast data stream for dissemination of information about every single trade.

All Trade Price messages will be disseminated over the All Trade Price stream.

Following points should be taken into account:

- The reversal of a trade generates **a message with delete trade indicator** with the appropriate trade quantity. As the trades of an auction are accumulated (the trade price is the same for all trades) and distributed as one trade, the (total) trade quantity has to be calculated in case of a trade reversal for a trade during the auction phase.
- Trade information is sequenced per instrument.
- Xetra works on the basis of "units of work". A single unit of work may consist of multiple trades, each of which generates a discrete trade message (All Trade Price). Each unit of work always delivers only a single delta message, which includes a summary position of all trades occurring within that unit of work.

For more information on the FAST implementation, refer to the XML template in section 12.

ALL TRADE PRICE MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
timestamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)

ALL TRADE PRICE MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
isix	Instrument identifier	N.A.	Mandatory	delta	Isix type (7.15)
gapIndicator	flag to indicate gap (some prior trades may not be reported)	3	Optional	constant	Gap indicator type (7.9)
noEntriesAtp	Number of entries	N.A.	Mandatory	N.A.	Number of values type (7.18)
>entryType	Type of entry	N.A.	Mandatory	none (trade)	Entry type. Allowed values are: LAST TRADE PRICE, CROSSED PRICE, BEST PRICE, MID-POINT PRICE (7.7)
>entryPrc	Price @ the above level	N.A.	Mandatory	delta	General price type (7.10)
>entryQty	Quantity offered @ above price	N.A.	Mandatory	delta	Quantity type (7.21)
>entryTime	Time of entry	N.A.	Mandatory	delta	Time type (7.30)
>tranMtchIdNo	Internal trade identifier	N.A.	Mandatory	delta	Transaction Id type (7.33)
>tpSeqNum	Sequence number	N.A.	Mandatory	delta	Sequence number type (7.22)
>actnCod	Action code	N.A.	Mandatory	none	Action code type (7.32)

As with delta message sequence numbers, tpSeqNum increments sequentially only within the context of a single source identifier. The field *tpSeqNum* (and *lastTpSeqNum* in the delta message) must therefore only be compared when the source is the same.

5.2.3.2 Trade Reversals

The Xetra Enhanced Broadcast Solution features trade reversal broadcasts. Trade reversals are disseminated over All Trade Price stream. The fields entryType and tpSeqNum are set to 0.

Trade reversals do not affect the Snapshot and the Delta stream. Notably the following fields are not updated:

- Last trade price
- Daily high price of the instrument
- Daily low price of the instrument

5.2.4 State Changes

This section covers status information that applies to exchange and system changes.

The Xetra state change information subscription is used by the application to subscribe to the Xetra public broadcast data stream for dissemination of Xetra state change information.

The state change messages are sent over the state changes stream both cyclically and event-driven. The current system and exchange states are sent cyclically. An event occurs as soon as an exchange or system change happens (see 7.25).

The messages will not be sequenced.

For more information on the FAST implementation, refer to the XML template in section 12.

ALL TRADE PRICE MESSAGE STRUCTURE					
Field Name	**Descriptive Name**	**Presence Map Bit**	**Presence**	**FAST Operator**	**Field Format**
TID	Template ID	1	Mandatory	copy	Template ID type (7.29)
time-stamp	Timestamp	N.A.	Mandatory	delta	Timestamp type (7.31)
srcId	Source identifier	2	Mandatory	copy	Source ID type (7.24)
state	Exchange or System State	3	Mandatory	copy	State type (7.25)

6. Appendix - Glossary of Terms

Term	Explanation
Ask	A price in the order book at which a financial instrument can be bought.
Bid	A price in the order book at which a financial instrument can be sold.
Datagram	Datagram is a unit of information sent by the source. A datagram could be split into one or more network packets.
Delta Message	A delta message carries only the changes in the information sent earlier on the entity that it describes.
Empty Book	An order book which does not contain any information (price levels).
ETS	Enhanced Transaction Solution
FAST	FIX Adapted for STreamingSM (FAST ProtocolSM).
FIX	Financial Information eXchange.
Gap	Missing information related to trades or changes to the order book.
ITA2	Encoding International Telegraph Alphabet (Standard) 2.
Last Auction Price	This is the price of the last auction for the instrument.
Last Trade Price	Last trade price is the price of the last (most recent) trade for the instrument.
Live-Live Concept	Live - live concept means that the same data is disseminated in parallel over two distinct sets of streams.
Multicast	Many-to-many network architecture as against one-to-one (unicast).
Opening Price	This is the price of the first trade for the instrument. This price is recoverable through the snapshot stream.
Order	A contractually-binding request to other market participants to buy or sell a specific quantity of a financial instrument at a defined price.
Order Book (ODB)	Contains all current orders for an instrument, according to their trading restrictions and execution conditions.
Packet	Packet is generally referred to a single set of data bits carried by the network.
PIM	Protocol Independent Multicast.

Term	Explanation
Potential Auction Price	This is a theoretical value published in case of a cross order book, when the best bid and best ask prices are not revealed. These single prices represents both the buy and sell side of the ODB. This price is unrecoverable.
Price	Public price information (bid/ask price, bid/ask quantities, traded quantity, daily high/low, etc).
Quote	Simultaneous entry of a limit buy and a limit sell order for the same instrument.
Snapshot Message	A snapshot message carries complete (all available) information about the entity that it describes.
Stream	A stream is a collection of one or more instruments and is the basic unit of subscription.
UDP	User Datagram Protocol.
VALUES	Virtual Access Link Using Exchange Services. It is the programmable interface providing connectivity to Deutsche Börse's electronic trading platforms (Eurex and Xetra®). VALUES API provides a single point of entry to the full range of the exchange functionality.

7. Appendix - Data Field Dictionary

The following section provides specific information on the Xetra Enhanced Broadcast Solution data fields, listing valid values, ranges or giving an example value for the field.

7.1 Address Type

Description:	The multicast address of the channel.
Type:	ASCII Character String

Value Range	Meaning
“224.0.0.0” ... “239.255.255.255”	IPv4address

Where sent:

- Instrument Reference Data Message
- Maintenance Reference Data Message

7.2 Book Type

Description:	Type of Book
Type:	Unsigned long with NULL support

Value Range	Meaning
2	Price depth

Where sent:

- Instrument Reference Data Message

7.3 Business Date Type

Description:	Current business date.
Type:	ASCII Character String

Value Example
e.g. “20070101”

Where sent:

- Start / End Service Message

7.4 Cmex Indicator

Description:	Indicator for CUM/EX events or capital adjustment
Type:	ASCII Character String

Value	Meaning	FIX Corporate Action (292)
" " CMEX_IND_NO_CHG , CMEX_IND_REST	No change / Reset	Not sent
"C" - CMEX_IND_CUM	Cum dividend	<no standard value>
"A" - CMEX_IND_ADJ	Capital adjust-ment	<no standard value>
"E" - CMEX_IND_EX	Ex dividend	"A"

Where sent:

- Snapshot Message
- Delta/Incremental Message

7.5 Currency Code Type

Description:	Currency code of the instrument. This field identifies the currency in ISO 4217 representa-tion.
Type:	ASCII Character String

Value Examples	Meaning
“EUR”	Euro
“GBP”	British Pound
“USD”	US Dollar

Where sent:

- Instrument Reference Data Message

7.6 Entry Level Type

Description:	The type of entry level
Type:	Unsigned long

Value Range	Meaning
1	Top of book
1 ... 50	Depth level

Where sent:

- Snapshot Message
- Delta / Incremental Message

7.7 Entry Type

Description:	Type of entry
Type:	Unsigned long

For a mapping to FIX of these values, see section 11.2.2.3.

Value		Meaning
0-	EMPTY_ENTRY	No entry
1-	ASK_PRC	Ask price
2-	BID_PRC	Bid price
3-	EMPTY_BOOK	Empty Orderbook
4-	LAST_TRD_PRC	Last trade price
5-	OPENING_PRC	Opening price
6-	LAST_AUC_PRC	Last auction price
7-	VAL_PRC	Valuation price
8-	CLOSE_PRC	Closing price
9-	CROSSED_PRC	Crossed trade price
10-	BEST_PRC	BEST trade price
11-	MPO_PRC	Midpoint trade price
12-	PTNL_AUCT_PRC	Potential auction price

13-	MTCH_RNG_ASK	Matching range ask
14-	MTCH_RNG_BID	Matching range bid
15-	SRP_BID	Surplus bid
16-	SRP_ASK	Surplus ask
20-	HIGH_PRC_ENTRY	Daily high price
21-	LOW_PRC_ENTRY	Daily low price
22-	TOT_TRD_QTY	Total traded quantity
23-	ASK_PRC_MARKET	Ask market order
24-	BID_PRC_MARKET	Bid market order

Where sent:

- Snapshot Message
- Delta/Incremental Message
- All Trade Price Message

7.8 Exchange Id Type

Description:	The exchange code where the instrument is traded. International standard (ISO 10383) for codes for exchanges and market identification (MIC).
Type:	ASCII Character String

Value Examples	**Meaning**	
XETR	Xetra (Frankfurt Stock Exchange)	Sent via Instrument Reference Data Stream
XDUB	Dublin (Irish Stock Exchange)	
XETI	Xetra International Market	
XEUB	Eurex Bonds	
XETR	Xetra (Frankfurt Stock Exchange)	Sent via Maintenance Reference Data Stream
XDUB	Dublin (Irish Stock Exchange)	
XETI	Xetra International Market	

Where sent:

- Instrument Reference Data Message
- Maintenance Reference Data Message

7.9 Gap Indicator Type

Description:	A flag to indicate a gap in the information.
Type:	ASCII Character String

Value Range	Meaning
Y	Possible gap in operation

Where sent:

- Delta / Incremental Message
- All Trade Price Message

7.10 General Price Type

Description:	Price field. Float field expressed as exponent (4 bytes) and mantissa (8 bytes) fields.
Type:	Scaled Number

Value Example
exponent = -2 mantissa = 2555 resulting value = 25.55

Where sent:

- Instrument Reference Data Message
- Snapshot Message
- Delta/Incremental Message
- All Trade Price Message

7.11 Instrument Group Type

Description:	Instrument group will usually be disseminated via the same stream.
Type:	ASCII Character String

Value Example
DAX1

Where sent:

- Instrument Reference Data Message

7.12 Instrument Status Type

The status of the instrument is comprised in the field *instrStatus*, the mapping to FIX is not easy since the instrument status field bonds more than one FIX hierarchy level into one.

FIX considers trading subsessions to be the planned or even scheduled phases of a trading session which have a (possibly random) pre-defined duration. Within a phase, there is an instrument specific trading status reflecting the state of an instrument, e.g. if it is available for trading or halted.[4]

FIX Hierarchy:

TradingSessionID (336)	(=Day)
TradingSessionSubID (625)	(Pre-Trading, Opening, Trading, Closing...)
SecurityTradingStatus (326)	(Halt, Ready to trade, Not available...)

The mapping done in this implementation is a flattening of the structure defined in FIX with the two fields TradingSessionSubID (625) and SecurityTradingStatus (326).

The session, FIX's TradingSessionID (326), will always have the value "Day" so (for reasons of space) it won't appear in the tables below.

The subsession, FIX's TradingSessionSubID (625), needs to have the following user defined values since the values in FIX don't cover all Enhanced Broadcast Solution possibilities. Following FIX rules, the user defined are defined starting at 100 and upward.

[4] For further information please refer to section 3.1.3 of the FIX document "Exchanges/ECN Working Group Recommended Best Practices - Phase 1" http://www.fixprotocol.org/documents/3005/EEWG%20Recommended%20Best%20Practices%20-%20Phase%201%20V1.0.pdf

TradingSessionSubID	
Value	**Status**
100	Start
101	Pre-Call
102	Crossing Period
103	Closing Crossing Period
104	End Auction Call
105	Auction Call
106	Opening Auction IPO
107	Intra Day Auction IPO
108	IPO
109	Quote Driven IPO
110	End-of-day Auction
111	Pre-Orderbook Balancing
112	Orderbook Balancing
113	In Between Auction
114	Manual Intra Day Auction
115	Add
116	Delete
117	Call
118	Continuous Auction

The states "ADD" and "DELETE" (new defined values 115 and 116) refer to the adding and deleting of an instrument.

Also the status, FIX's SecurityTradingStatus, needs to have the following values defined since the values in FIX don't cover all Enhanced Broadcast Solution possibilities. Following FIX rules, the user defined values are defined starting in 100 and upward.

SecurityTradingStatus	
Value	**Status**
100	Available for Trading
101	Call
102	Freeze

SecurityTradingStatus	
Value	**Status**
103	Pre-Orderbook Balancing
104	Orderbook Balancing
105	Suspend
106	VOLA
107	Pre-Call

Enhanced Broadcast Solution State		Description	TradingSessionSubID (625)	SecurityTradingStatus (326)
0-	START	Start	100	17
1-	PRETR	Pre Trading	1	21
2-	QPREC	Pre-call	101	N.A.
3-	QPRCX	Crossing Period	102	N.A.
4-	QPRCC	Closing Crossing Period	103	N.A.
5-	OCALL	Opening Auction Call	2	101
6-	CALL	Intra Day Auction Call	6	101
7-	CCALL	Closing Auction Call	4	101
8-	ECALL	End Auction Call	104	N.A.
9-	QCALL	Auction Call	105	N.A.
10-	OIPO	Opening Auction IPO Call	106	101
11-	OFRZ	Opening Auction IPO Freeze	106	102
12-	IIPO	Intra Day Auction IPO Call	107	101
13-	IFRZ	Intra Day Auction IPO Freeze	107	102
14-	QIPO	IPO	108	N.A.
15-	QPRZ	Quote Driven IPO Freeze	109	N.A.
16-	OPOBB	Opening Auction Pre-Orderbook Balancing	2	103
17-	IPOBB	Intra Day Auction Pre-Orderbook Balancing	6	103

Enhanced Broadcast Solution State		Description	TradingSessionSubID (625)	SecurityTradingStatus (326)
18-	CPOBB	Closing Auction Pre-Orderbook Balancing	4	103
19-	EPOBB	End-of-day Auction Pre-Orderbook Balancing	110	103
20-	QPOBB	Pre-Orderbook Balancing of quote driver auction	111	N.A.
21-	OOBB	Opening Auction Orderbook Balancing	2	104
22-	IOBB	Intra Day Auction Orderbook Balancing	6	104
23-	COBB	Closing Auction Orderbook Balancing	4	104
24-	EOBB	End-of-day Auction Orderbook Balancing	110	104
25-	QOBB	Orderbook Balancing	112	N.A.
26-	TRADE	Continuous Trading	3	100
27-	BETW	In Between Auctions	113	N.A.
28-	POSTR	Post Trading	5	18
29-	ENDTR	End Of Trading	7	18
30-	HALT	Halt	3	2
31-	SUSP	Suspend	3	105
32-	VOLA	Volatility Interruption	3	106
35-	ADD	Add	115	18
36-	DEL	Delete	116	18
38-	CUNF	Call Unfreeze	117	100
39-	XPREC	Continuous Auction Pre-Call	118	107
40-	XCALL	Continuous Auction Call	118	101
41-	XFRZ	Continuous Auction Freeze	118	102

Where sent:

- Snapshot Message
- Delta / Incremental Message

7.13 Instrument Type

Description:	Type of instrument
Type:	ASCII Character String

Value Example	Meaning
BAS	Basis Instrument
BON	Bond
EQU	Equity
WAR	Warrant

Where sent:

- Instrument Reference Data Message

7.14 ISIN Code Type

Description:	The ISIN code of the instrument. The ISIN code follows standard in ISO 6166 (http://www.anna-nna.com).
Type:	ASCII Character String

Value Example	Meaning
DE0005810055	Deutsche Börse AG

Where sent:

- Instrument Reference Data Message

7.15 Isix Type

Description:	Internal identifier assigned to each instrument.
Type:	Unsigned long

Value Range
1 ... 4294967295

Where sent:

- Beacon Message

- Instrument Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message

7.16 Market Order Interruption Indicator

Description:	A flag to indicate a market order interruption.
Type:	ASCII Character String

Value		Meaning
" "		No market order interruption
"P" -	MOI_IND_POT	Potential market order interruption
"M" -	MOI_IND_MOI	Market order interruption
"X" -	MOI_IND_EXP	Volatility interruption after market order interruption

Where sent:

- Snapshot Message
- Delta/Incremental Message

7.17 Mnemonic Type

Description:	The abbreviated form of the instrument name
Type:	ASCII Character String

Value Example	Meaning
"DB1"	Deutsche Börse AG.

Where sent:

- Instrument Reference Data Message

7.18 Number of Values Type

Description:	A general field carrying a number
Type:	Unsigned long

Value Range
0 ... 4294967295

Where sent:

- Start / End of Instrument / Maintenance Reference Data Message
- Instrument Reference Data Message
- Maintenance Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message

7.19 Port Type

Description:	The port where the data is disseminated.
Type:	Unsigned long

Value Range
0 ... 65535

Where sent:

- Instrument Reference Data Message
- Maintenance Reference Data Message

7.20 Price Action Type

Description:	Describes the update action for the price entry in orderbook
Type:	Unsigned long

Value		Meaning
1-	CRE	New
2-	CHG	Change
3-	DEL	Delete
4-	DEL_FRM	Delete From. Delete from level ‘X’ to maximum levels.
5-	DEL_THR	Delete Through. Delete from level 1 to level ‘X’.

Where sent:

- Delta / Incremental Message

7.21 Quantity Type

Description:	The quantity or volume field. Float field expressed as exponent (4 bytes) and mantissa (8 bytes) fields.
Type:	Scaled Number

Value Range
exponent = -1 mantissa = 2500 resulting value = 250.0

Where sent:

- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message

7.22 Seqence Number Type

Description:	Sequence number to assist in identifying data loss. Note: Not all Xetra Enhanced Broadcast Solution messages are sequenced.
Type:	Unsigned long

Value Range
0 ... 4294967295

Where sent:

- Version Information Message
- Beacon Message
- Instrument Reference Data Message
- Maintenance Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message

7.23 Set Id Type

Description:	Technical grouping of instruments.
Type:	Unsigned long

Value Range
1 ... 1007

Where sent:

- Instrument Reference Data Message

7.24 Source Id Type

Description:	Identifier of the message disseminating source (host). Members should maintain the sequence numbers of the incoming messages per disseminating source.
Type:	Unsigned long

Value Range
0 ... 4294967295

Where sent:

- Version Information Message
- Beacon Message
- Instrument Reference Data Message
- Maintenance Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message
- State Changes Message

7.25 State Type

Description:	Identifier for Exchange and System states depending on whether the change applies to the system or to trade model 8 (Trading Model Continuous Auction)
Type:	Unsigned long

Value	Meaning	Refers to
1-	System Online	System
2-	System in Batch	System
3-	System Halted	System
4-	Fast Market Off for Equities	System
5-	Fast Market On for Equities	System
6-	Fast Market Off for Bonds	System
7-	Fast Market On for Bonds	System
8-	Fast Market Off for Warrants	System
9-	Fast Market On for Warrants	System
10-	Fast Market Off for Basis Instruments	System

11-	Fast Market On for Basis Instruments	System
12-	Exchange Start	Trading Model Continuous Auction
13-	Exchange Pre-trade	Trading Model Continuous Auction
14-	Exchange Trade	Trading Model Continuous Auction
15-	Exchange End-trade	Trading Model Continuous Auction
16-	Exchange Post-trade	Trading Model Continuous Auction
17-	Exchange Halt	Trading Model Continuous Auction

Where sent:

- State Changes Message

7.26 Stream Depth Type

Description:	The stream depth type indicates the defined order book depth for the channel.
Type:	Unsigned long with NULL support

Value Range
0 ... 50

Where sent:

- Instrument Reference Data Message

7.27 Stream Service Type

Description:	The stream service type indicates whether the stream is part of service A or service B of the Xetra Enhanced Broadcast Solution.
Type:	ASCII Character String

Value Example	Meaning
“A”	Service A
“B”	Service B

Where sent:

- Instrument Reference Data Message
- Maintenance Reference Data Message

7.28 Stream Type

Description:	The type of the stream.
Type:	ASCII Character String

Value		Meaning
“1”	SNAPSHOT	Snapshot
“2”	DELTA	Delta / Incremental
“3”	ALL TRADE PRICE	All Trade Price

Where sent:

- Instrument Reference Data Message

7.29 Template Id Type

Description:	The identifier used to indicate the FAST template to be used for decoding.
Type:	Unsigned long

Value	Meaning
1	Version Information Message
2	Beacon Message
3	Instrument Reference Data
4	Maintenance Reference Data
6	Snapshot Message

7	Delta/Incremental Message
9	All Trade Price Message
10	State Changes Message
120	FAST Reset Message
128	Start of Service Message
129	End of Service Message
130	Start of Instrument Reference Data Message
131	End of Instrument Reference Data Message
132	Start of Maintenance Reference Data Message
133	End of Maintenance Reference Data Message

Where sent:

- Version Information Message
- Beacon Message
- Instrument Reference Data Message
- Maintenance Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message
- State Changes Message
- Start / End Service Message
- Start / End Instrument Reference Data Message
- Start / End Maintenance Reference Data Message

7.30 Time Type

Description:	Representation of time in ASCII.

Type:	ASCII Character String

Value Range	Meaning
“HHMMSSCC”	e.g. “15535900” represents the time 15:53:59.00

Where sent:

- Snapshot Message
- All Trade Price Message

7.31 Timestamp Type

Description:	The timestamp when the message was created by the Enhanced Broadcast Solution, in the format milliseconds since midnight CET/CEST.
Type:	Unsigned long

Value Range	Meaning
0...86399999	Milliseconds since midnight

Where sent:

- Beacon Message
- Instrument Reference Data Message
- Maintenance Reference Data Message
- Snapshot Message
- Delta / Incremental Message
- All Trade Price Message
- State Changes Message
- Start / End Service Message
- Start / End Instrument Reference Data Message
- Start / End Maintenance Reference Data Message

7.32 Action Code Type

Description:	This field indicates what type of action should be performed in case of last trade price information.
Type:	Unsigned long

Value	Meaning
4-	Add
5-	Update
6-	Delete

Where sent:

- All Trade Price Message

7.33 Transaction Id Type

Description:	Internal transaction matching identifier
Type:	Unsigned long

Value Range
1 ... 4294967295

Where sent:

- All Trade Price Message

7.34 Version Number Type

Description:	The Xetra Enhanced Broadcast Solution stamps every outgoing message with the version number indicating the level of operation. This field allows existence of more than one broadcasting service with overlapping multicast addresses in the future. The receiver is expected to process only the datagrams carrying the correct service version number.
Type:	Unsigned long

Value	Meaning
0 ... 4294967295	Interface service version

Where sent:

- Version Information Message

7.35 Vola Indicator Type

Description:	Indicates whether an instrument has a potential volatility interruption or an actual volatility interruption or neither.
Type:	ACSCII Character String

Value	Meaning
“E”	Extended volatility interruption
“F”	Freeze
“P”	Potential volatility interruption
“V”	Volatility interruption
“X”	Expired
“ ”	No change

Where sent:

- Snapshot Message
- Delta / Incremental Message

8. Appendix - Interface Limits

Although the protocol is very generic and supports unlimited data size, depending on the scope of the message and the resulting data property, some limits can be stated for the Xetra Enhanced Broadcast Solution.

Limit	Descriptive	Name / Message	Value
MAX LIVE LIVE SERVICES	The Enhanced Broadcast Solution interface disseminates data simultaneously in duplicate over two services, 'A' and 'B'	noStreams / Maintenance Reference Data	2
MAX DELTA CHANNELS	Maximum delta/incremental channels per instrument, 5*2 (live-live)	noOfSeqNum / Snapshot Message	10
MAX SNAPSHOT CHANNELS	Maximum snapshot channels per instrument	N.A.	2
MAX ALL TRADE PRICE CHANNELS	Maximum All Trade Price channels per instrument	N.A.	2
MAX STREAM ENTRIES	Maximum number of channel entries for an instrument. All Trade Price, Snapshot and incremental together (1 All Trade Price + 1 snapshot + max. 5 delta) * 2 for live-live services.	noStreams / Instrument Reference Data	14
MAX ODB DEPTH	Maximum depth of order book supported by Enhanced Broadcast Solution.		50
MAX ODB ENTRIES	Maximum entries for order book data. (MAX ODB DEPTH * 2)	noEntriesDepth / Snapshot and Delta	100
MAX PRICE ENTRIES	Maximum entries for price data (opening price, last auction price, high price, low price, closing price, valuation price)	noEntriesPrc / Snapshot and Delta	7
MAX PRICE/ QUANTITY ENTRIES	Maximum entries for price and quantity data (potential auction price; matching range bid/ask)	noEntriesPrc / Snapshot and Delta	2
MAX QUANTITY ENTRIES	Maximum entries for quantity data (surplus bid/ask)	noEntriesPrc / Snapshot and Delta	2

Limit	Descriptive	Name / Message	Value
MAX ALL TRADE PRICE ENTRIES	Maximum trade prices in one message (last trade price, crossed price, last auction price, midpoint price, best price)	noEntriesAtp / Snapshot and All Trade Price	5
MAX STRING SIZE	Maximum size of any string disseminated through the interface. Enhanced Broadcast Solution disseminates mainly binary data.	N.A.	16

9. Appendix - How to Use

This section explains how member applications should use the Xetra Enhanced Broadcast Solution interface. A member application will be used to explain this in more detail.

Please note that the operations shown below are basic and members are expected to integrate the operations described below into their trading system.

Due to multiple environments, the same instrument (ISIN) can be processed on different market sources. The instrument identifier (ISIX) is only distinct within its context. It means that the same instrument identifier processed on different market sources can represent different instruments. For that reason the version number in the version message (see chapter 5.1.1) has to be considered to determine the market source.

9.1 Start of Day

9.1.1 Connect to the Reference Data Stream

Member applications must connect to the reference data stream for each market source and receive the reference data messages as explained in chapter 4. (Broadcast Streams) before they start receiving and processing any data (snapshot, delta, All Trade Price broadcasts and system state changes).

The address for this first stream will be published by Xetra. Through this stream, instrument reference data and maintenance reference data is distributed.

Steps involved:

- Join the reference data stream multicast address.
- Wait until the start of instrument or maintenance reference data message is received.
- Once this message is received, start processing all subsequent UDP datagrams from this multicast address until the end of instrument (or maintenance) reference data message is received.
- Wait until the start of maintenance reference data message is received, if the instrument reference data has already been received or wait until the start of instrument reference data message is received, vice versa. Both types of reference data have to be received. It does not matter which one is received and processed first.
- Start processing all subsequent UDP datagrams until the corresponding end of reference data message is received.
- Leave this stream.

This information has to be stored for future use. The instrument reference data contains information on sets, instruments, streams, channels and the corresponding multicast addresses. The maintenance reference data contains information on the corresponding multicast addresses of state changes.

Members must use the information obtained to identify the sets and instruments in which they are interested. Members are expected to join only the required streams.

9.1.2 Receive Snapshot

Members have to first join the snapshot streams for the instruments which they require and after receiving the order book snapshots, members should leave this stream.

Steps involved to receive orderbook snapshots:

- Connect to the delta channels (first) and buffer the orderbook deltas while they receive the snapshots, so that no delta message is lost.
- Join the snapshot streams for the instruments.
- Wait until the datagram carrying the orderbook snapshot for the instrument is received and discard all unwanted packets (see section 9.2 on synchronization).
- Apply the snapshot to the members' copy of the orderbook.
- Leave all the snapshot streams after receiving the initial snapshots.

Members may have to join more than one multicast address if the instruments of their interest are not broadcast on the same stream, multicast address

9.1.3 Process Delta

Members are expected to stay connected to this stream throughout the day.

The following steps need to be performed for the delta streams:

- Join the delta streams of interest. This could involve joining multiple channels (multiple multicast addresses).
- The top of book channel disseminates trade information for the instrument.
- Process datagrams containing data related to the instruments of interest. Discard all unwanted datagrams.
- Apply the incoming deltas to the local copy of the order book.

Please note that due to a limited number of multicast address and port pairs information of more than one instrument group can be disseminated on one multicast address and port pair for snapshot and incremental streams. Members will have to filter the information for the sets and instruments of their interest.

9.2 Messages Sequence Numbers Synchronization

Due to the unreliable nature of the UDP protocol, the order in which the broadcasts arrive, whether snapshots, deltas and/or All Trade Prices is random. Furthermore the UDP packets can be duplicated at the network.

Member applications must be prepared for the fact that multicast does not guarantee the correct sequencing of messages.

To be able to keep track of what has been received in each of these three streams (All Trade Price, deltas and snapshots), sequence numbers have been added to the orderbook and trade messages. These sequence numbers along with the source identifier (where is this message coming from) can be used to detect messages lost.

9.2.1 Snapshot - Delta Synchronization

Order book snapshots carry consolidated sequence numbers, field *consolSeqNum*, for each of the configured delta streams, providing the receiver with sufficient information to position the snapshots exactly amongst the delta messages in each stream.

Each delta message for a specific instrument has a unique sequence number. This sequence number is assigned by the source and is useless unless the source is also taken into account. The sequence number (field *seqNum*) increments sequentially, always within the context of a single source identifier. Therefore, these sequence numbers can be used for synchronization purposes whenever they are coming from the same source.

If any gaps are detected in the sequence numbers on the delta stream and they are not filled after a period of time (some time must be allowed for the unreliability of the UDP interface), member applications can either join the snapshot stream again and receive the latest order book snapshot or try rebuilding their copy of the order book with the next delta messages for the instrument.

Each snapshot message has a reference to the latest delta sequence number in each stream whose information it comprises.

With this information, the member knows as soon as he receives a snapshot, which delta messages are already included in the snapshot information and therefore can be thrown away since they will not add any new information. And, on the contrary, which messages have information more recent than the snapshot and must be taken into account to have an up-to-date orderbook.

For example, assuming a scenario like the one in the figure below, sequence numbers coming all from the same source and for the same instrument:

	Delta	Snapshot
t1	42	...
t2	43	...
t3	44	...
t4	45	44
t5	...	...
t6	46	...
t7	47	...
t8	...	48
t9	48	...
t10	49	...
t11	50	...
t12	...	50

During the period t1-t12, deltas message sequence numbers from 42 to 50 and three snapshots are received.

The member subscribes to the snapshot and delta streams for a specific instrument at time t1 and starts receiving messages, first come deltas. When a snapshot arrives, it states that, for that stream, the snapshot has consolidated the information up to delta message 44, so the first three received deltas can be ignored (delta messages numbers 42, 43 and 44) whereas delta message number 45 has to be incorporated into the global picture.

Then delta message 46 arrives at t6 and delta message 47 arrives at t7 and both changes are applied to the orderbook.

When snapshot message arrives at t8 with consolidated delta sequence number 48[5], we know that it has some information that we are missing, so we take it.

Then at t9, delta message with sequence number 48 arrives and we ignore it.

Then delta messages with sequence numbers 49 and 50 arrive and their information is saved.

Then snapshot message with delta sequence number 50 arrives and we ignore it because we have already received everything it has, via the delta stream.

9.2.2 Delta - All Trade Price Synchronization

Xetra works on the basis of “units of work”. Each incoming order or quote results in one unit of work during Continuous Trading. Each Exchange State change results in one single unit of work.

Each unit of work always delivers only a single delta message, which includes a summary position of all trades occurring within that unit of work. The summary position contains the fields total traded quantity and last trade price. Total traded quantity is the cumulated quantity of the instrument for the current business day. The last trade price refers to the last price within the unit of work.

A single unit of work may consist of multiple trades, each of which generates a discrete trade message (All Trade Price). Xetra Best, Midpoint and Crossed prices are not included in the summary position (last trade price and total traded quantity) of the delta message. Members requiring a full view of all reported trades must therefore use the All Trade Price messages, not the delta messages for this purpose.

Field *lastTpSeqNum*

The field *lastTpSeqNum* in the delta message indicates the highest trade price sequence number included in the order book resulting from this message.

Although trade price sequence numbers are assigned to trade prices sequentially, there is not necessarily a one-to-one correlation between trade price events and deltas. A single unit of work on the host may result in many trades (which are reported individually) but only a single orderbook delta, delivered at the end of the unit of work.

Therefore, for the synchronization between these two streams, this sequence number field must not be expected to increment sequentially.

The field *lastTpSeqNum* is set to 0 for instruments in which there have been no trades.

9.2.3 Snapshot - All Trade Price Synchronization

The field *lastTpSeqNum* in the Snapshot message is a reference to All Trade Price sequence number. It contains the highest sequence number that is considered by the current Snapshot message.

The field was introduced to be able to recover the field totTrdQty from the Snapshot stream.

The synchronization between the Snapshot and All Trade Price stream works the same way as the synchronization between Delta stream and All Trade Price stream.

[5] This case is highly improbable since delta messages are created and sent before snapshot, but it could happen due to problems in the transmission.

9.3 Order Book Recovery

Due to the inherently unreliable nature of the UDP protocol, occasional loss of UDP packets in the network is expected. Loss of data packets can be detected from non-sequential numbers of messages for an instrument.

The order book will be available for recovery purposes on the snapshot streams throughout the day.

9.3.1 Loss of Packets

If the member joins both the interface services A and B, packets may be received from either of the two services. A permanent loss of packet(s) is established when a member detects a sequence number gap.

If a copy of the order book cannot be generated, a recovery is required and the following steps need to be performed.

Please note that as a property of multicast networks, packets may not be delivered in the order they are sent by the information source.

9.3.1.1 Receive Snapshot for an Instrument

Join the snapshot stream of the instrument to receive the current ODB snapshot:

- Join the snapshot stream for the instrument. Stay connected to the delta channel and buffer all incoming deltas.
- Wait for the packets carrying snapshot information message on the instrument and discard all the other packets.
- Apply the snapshot to the local copy of the ODB. Every snapshot carries the consolidated sequence numbers for all the delta channels.
- Apply the buffered deltas that were issued after the snapshot.

9.3.1.2 Leave the Snapshot Stream

Leave the snapshot channel and continue working with the deltas.

9.3.1.3 Continue Operation with the Delta Stream

Maintain the orderbook by processing all incoming deltas from the delta stream.

9.4 Events Demanding Special Response

Due to the transparent operation of the Xetra Enhanced Broadcast Solution, there are situations when the interface will indicate that there has been some technical interruption or rearrangement of services at Xetra.

Members are expected to adjust to the prevailing situation at that point of time.

Listed below are the situations that may require special response from the member applications. Members should tailor their applications to respond to these events which will be classified as "Normal" or "Exceptional".

Please note that this list is not exhaustive so any changes to this list will be published.

9.4.1 Gap Indicator

Type	Normal
Where sent	The Gap Indicator field is disseminated in ODB incremental/delta and trade information messages for an instrument.
Value	Y
Description	ODB incremental/delta message If sent and the value is “Y”, it indicates that the Xetra Enhanced Broadcast Solution interface may have missed some of the latest orderbook changes. This situation can occur if the Xetra Enhanced Broadcast Solution data generators were busy with other instruments when there were more than one ODB update for an instrument. The next ODB delta message for such an instrument is sent out with the gap indicator set to indicate this event. This message will contain the latest orderbook update, so that the members’ orderbook view will be consistent. When the gap indicator is not sent, it should be assumed that there is no gap, that Xetra has sent all the order book updates for the instrument.
	Trade information message When this field carries value “Y”, it indicates that some intermediate trades had occurred for the instrument which may not be reported due to heavy processing load. When the gap indicator is not sent it should be assumed that Xetra has sent information on all trades for the instrument.

9.4.2 Source Identifier

Type	Exceptional
Where sent	Every message sent out of the Xetra Enhanced Broadcast Solution interface carries a source identifier field.
Value	Changes from the previous value being sent for the instrument.
Description	When the source of information to the Xetra Enhanced Broadcast Solution changes, namely the Xetra trade matching system, this value is changed accordingly. There can be situations when this value alternates between two or more values. Members should maintain the market information for each instrument per source identifier.

9.4.3 Sequence Number

Type	Normal
Where sent	The Xetra Enhanced Broadcast Solution sequences Delta and All Trade Price messages per instrument per source.

Value	Non-consecutive sequence numbers.
Description	This can occur due to the unreliable nature of the UDP protocol. As the interface sequences all messages per instrument per source, any message loss will be known to the member immediately. If members wish to recover from this situation, they can join the snapshot channel for the instrument to receive the recent orderbook snapshot and then continue with the delta channel. Note that while connecting to the snapshot channel the client application has to remain connected to the delta channels and buffer all the incoming order book deltas. The order book snapshot message carries the last consolidated sequence number for every delta channel. After the snapshot is received the client application has to discard all the deltas already included in the snapshot and then apply the remaining deltas to the order book view carried by the snapshot to create the recent market picture once again.

Type	Exceptional
Where sent	The Xetra Enhanced Broadcast Solution sequences Delta and All Trade Price messages per instrument per source.
Value	Sequence numbers abruptly restart.
Description	This can occur due to two reasons: (1) If the source of market data, namely the order matching system, changes. Every time the Xetra Enhanced Broadcast Solution interface starts receiving data from a new source, it restarts sequencing the messages. (2) Technical malfunction. If there is some technical malfunction and if the interface is unable to recover the old market information, the Xetra Enhanced Broadcast Solution interface can choose to restart sequencing the messages.

9.4.4 Start/End of Service Message

Type	Exceptional
Where sent	On start and end of the interface services.
Value	Messages sent within the trading day.
Description	Due to technical malfunctions Xetra could restart some of the interface services. This will result in an additional start and end of service messages. Members should treat this as an exceptional situation and respond to this event in an appropriate manner. The only purpose of the start and end of service messages is to alert members about the window in which data will be disseminated by the interface.

9.4.5 Trade Price Sequence Number

Type	Exceptional
Where sent	Snapshot and Delta messages carry the Trade Price Sequence Number
Value	Trade Price Sequence Number duplication.
Description	Under exceptional conditions, it is possible the same Trade Price Sequence Number to be sent for different trades. This only ever occurs as a result of a specific type of technical interruption to XETRA. When a duplicate Trade Price Sequence Number is received in the Snapshot or Delta message, the earlier trade was not actually posted, and should be discarded. No ATP was sent for the earlier occurrence of the Trade Price Sequence Number. The resolution of the Trade Event timestamp means that it should always be possible to determine which of the trades occurred first, regardless of the order in which they were received on the network.

9.4.6 Host Fail-over

Internally, the Xetra Enhanced Broadcast Solution platform is distributed across multiple host machines. During normal processing, each set and its associated instruments are processed by a single host, and it is this host that generates the Xetra Enhanced Broadcast Solution messages for these instruments.

If a host fails, responsibility for the instruments managed by that host is assigned to one or more of the remaining hosts, which also take over the broadcasting of Xetra Enhanced Broadcast Solution information for the affected instruments. This is referred to as a fail-over.

When the failed host is returned to service, a fail-back occurs which sees responsibility for the fail-over instrument returning to the original host. A fail-back will not always occur as soon as a failed host returns to service. It may be delayed until a transaction for a fail-over instrument occurs.

During such a fail-over or fail-back, depending on the mode of failure, it is possible for there to be a certain amount of “flutter” as responsibility is passed back and forth to clear queued transactions.

For technical reasons, it is also possible – though very rare – for this to occur during normal processing, without a node failure having occurred. This is an extremely rare occurrence, typically caused by a transient network-related problem within Xetra.

This information is important for client programmers because all Xetra Enhanced Broadcast Solution sequence numbers are maintained on a per-host basis. This applies to all fields identified as "Sequence Number Type" in the message descriptions (see section 7.24).

Hosts are opaquely but uniquely identified to members through the field *srcId*, present in all Xetra Enhanced Broadcast Solution messages.

It is important that clients maintain sequence numbers only in the context of a single source identifier, *srcId*. It is never appropriate to perform any operation (such as a comparison) on sequence numbers from different sources.

During a change of host (either as a result of a fail-over or a fail-back), a number of events occur that require special handling by the client code:

- Messages for the affected instruments start arriving with a different source identifier.
- The first sequence number of all types receive from the new host no longer follows the sequence established by the previous host.
- They will typically start from 1, but no assumption must be made about this behaviour as there are certain circumstances under which it is not true.
- Special handling requires the field *seqNum* located in Version Information Message. After a host fail-over the internal limit counter in the member application for *seqNo* of the failed host must be set to 0 after certain time interval (e. g. 5 minutes). The reason for that is, if the failed host restarts, the field *seqNo* can start from 1 again. If a member application is rejecting all *seqNo* below the last received for srcId of the failed host, the later send messages are lost.

The only correct response to these events is for the client to invalidate its view of the order book until an explicit message has been received containing new information. This can either be as a result of a recovery from snapshots, as detailed in section 9.2 or from order book delta messages.

When such a fail-over occurs, Xetra Enhanced Broadcast Solution automatically generates a “full depth” delta message for all of the affected instruments. This message is effectively a snapshot on the delta channel, with an operation at every level, and is intended to enable members to rebuild their orderbooks with minimal overhead.

Assuming no message loss, the application of this full depth delta message following a change of source obviates the need to recover from snapshots during a fail-over.

For completeness though, the suggested logic when a fail-over is identified is as follows:

- Invalidate the order book for the affected instrument.
- If the message being processed is a full depth delta (i.e. if the order book is entirely valid after applying this message):
- No further recovery action is required.
- Otherwise:
- Initiate recovery from snapshots as explained in section 9.2.
- If the orderbook becomes entirely valid whilst waiting for a snapshot (perhaps due to the arrival of the full depth delta):
- Abandon the recovery from snapshots.

To summarize the basic guidelines:

1. Messages generated by one host must never be mixed with messages from another host. For example, a delta arriving from one host must never be applied to an order book constructed from messages originated by a different host.
2. By inference, an order book must only be constructed of messages from the same host (same *srcId*).

3. Any change in the source identifier must be treated in the same way as an unexpected backward jump in delta sequence numbers, the client's view of the order book must be considered invalid, and must not be relied upon until a valid value has been received from the new host.

4. Sequence numbers of all types only have meaning within the context of a single host (same *srcId*).

Failure to adhere to these guidelines will result in an unpredictable order book under fail-over conditions.

10. Appendix - Message Encoding

This chapter explains the FAST concepts important for understanding the Xetra Enhanced Broadcast Solution data encoding logic. The Xetra Enhanced Broadcast Solution encoding adheres to the FAST 1.1 specification.

Please note that FPL's Market Data Optimization Working Group may in future modify the FAST concepts. Therefore it is important to describe/understand the FAST concepts which the Xetra Enhanced Broadcast Solution interface is based on.

10.1 FIX Adapted for STreaming^{SM} (FAST Protocol^{SM})

FAST makes use of proven data compression techniques that take advantage of the knowledge about the data content and the data formats.

For the latest descriptions about FAST, please refer to the link http://www.fixprotocol.org/fast.

FAST proposes a two step data compression method instead of a generic compression technique. The compression techniques are customized to the data content and minimize the repetition of fields to ensure that only the required bytes are used for the transmission of data. These techniques can be applied not only to FIX messages but to messages in any format.

Please note that this document does not contain the complete FAST specification but only explains the concepts relevant to the Xetra Enhanced Broadcast Solution.

10.1.1 Commonly Used Terms in FAST Context Explained

Term	Explanation
Application type	Represents the type of a group or a message in applications using FAST.
Field Instructions	Each field instruction has a name and a type. The name must be unique within the group. The type can be either primitive or sequence or group. Example: **<uInt32 name="timestamp" />**
Group	A group is a named type comprising an unordered set of fields. A group appearing at the top most level of a stream is also referred to as a message.

Term	Explanation
Instructions	There are two categories of instructions: • Field instructions specify how to encode fields of the instance to the stream. • Template reference instructions provide the means for defining parts of a template by reference to other templates. Encoding and decoding are performed with in the context of an instruction. An instruction context consists of: • set of templates • current template • set of application types • current application type • set of dictionaries • optional initial value Each field instruction has a name and a type. The name identifies the context of the application type. The optional presence attribute indicates whether the field is mandatory or optional. If the attribute is not specified, the field is mandatory.
Presence map	A sequence of bits. Fields of the presence map segment use the bits as specified by the current template. Not every field needs a bit in the presence map (see FAST documentation). When present, bit n indicates whether the corresponding field in the template is sent (‘1’) or not (‘0’).
Primitive field	A field that is not a group or sequence and can have a field operator. This operator specifies the optimization operation for the field.
Primitive type	Are ASCII strings, Unicode string, uInt32, int32, uInt64, int64, decimal (8 bytes mantissa plus 4 bytes exponent) and byte vector.

Term	Explanation
Segments	Consist of the following: • segment identifier • presence map • set of field instructions In general, a segment is a complete set of information. A FAST message is the highest level segment. A segment continuation bit is used to link data bytes together belonging to a field. This is the Most Significant Bit (MSB) of a data byte. If this is unset(0), it indicates that the following data byte is part of the current field and if set(1), it indicates that the current byte is the last byte of the data field.
Sequence	Comprises a length and an ordered set of elements. Each element is of group type and must not have identical group types.
Template	Specifies how to encode an instance of an application type or a part thereof as a stream of bytes.

10.2 Field Instructions

Field instructions refer to how the fields will be sent in the information stream. For further information, please refer to the FAST Specification Version 1.1:

http://www.fixprotocol.org/documents/3066/FAST Specification 1 x 1.pdf.

10.3 Field Operators

Field operators are used to remove the existing redundancies in the data formats. Message data fields can be related to one another. Message templates will be the meta-data for the message and will be provided earlier. When the messages arrive, the receiver application will have a complete knowledge of the message layout and needs to work only with the template and the incoming message to determine the field value.

Field instructions can optionally have operators which specify how the receiver program should determine the value of the field.

For further information, please refer to section 6 of the FAST Specification Version 1.1.

10.4 Transfer Encoding

Transfer Encoding refers to how the fields will be encoded in an information stream.

For further information, please refer to section 10 of the FAST Specification Version 1.1.

11. Appendix - FIX mapping

The Xetra Enhanced Broadcast Solution messages are FIX compatible but not fully FIX compliant for reasons of performance. They can be mapped to FIX compliant messages in a straightforward, context-free manner.

This section provides a reference mapping of the Xetra Enhanced Broadcast Solution interface messages to the FIX fields.

11.1 Financial Information eXchange (FIX) Protocol

FIX is a globally accepted messaging standard for areas of pre-trade, trade and post-trade (presettlement), including a market data exchange format (http://www.fixprotocol.org).

FIX specifies various standard messages with optional and mandatory fields for the purpose of financial data interchange.

11.2 Message Mappings to FIX

For complete FIX message descriptions refer to http://fixprotocol.org/FIXimate3.0/index.html

FIX only provides two types of response messages for market data, i.e. snapshot and incremental updates. The snapshot message is intended for distribution of complete data for a single security whereas the incremental message allows sending individual data elements. The data can be order book information (top of book and market depth) as well as statistical data such as high/low values and reference prices.

Reference data and status information related to securities is provided in additional, separate messages.

FIX messages have “mandatory fields”, “conditionally required fields”, “optional fields” and possibly "user-defined fields" and contain metadata (tags) specifying the subsequent field value. Optional fields not occurring in the data stream are not part of the message (neither the tag nor the actual value).

Please note that the Xetra data requirements are not exactly satisfied by standard FIX layouts. Hence not every field in the Enhanced Broadcast Solution interface has a (single) corresponding FIX tag, in FIX 5.1. However, the mapping suggests user-defined fields wherever there currently is a gap. These gaps might be closed by means of standard fields in subsequent releases of FIX through the submission of gap analysis documents to the FIX Global Technical Committee.

The Xetra Enhanced Broadcast Solution interface sends the following type of messages to the members:

- Instrument Reference Data
- Maintenance Reference Data
- Snapshot Messages
- Delta/Incremental Messages
- All Trade Price Messages
- State Change Messages

- Technical Beacon Messages
- Functional Beacon Messages

The following table shows which FIX formats will be used for each outgoing message.

Message	Content
MarketDataIncrementalRefresh	Deltas
MarketDataSnapshotFullRefresh	Snapshots, All Trade Prices
SecurityDefinition	Reference Data
TradingSessionStatus	State Changes

The following sections try to give an overview of the different templates that will be used with the new Enhanced Broadcast Solution and the data that each template will comprise.

The tables have the following structure, the rows are the fields used for each message, the columns indicate as follows:

- Enhanced Broadcast Solution: Name of the field in the software implementation
- VALUES: Name of the field in VALUES
- FIX Field / FIX Tag: Name and number of the FIX field that best represents this field
- FG: Column to indicate a FIX gap (Y=Yes, blank=No). Only filled when the mapping to FIX can not be done following a minimum standard
- Type: Type used for the implementation
- Size: Size (in bytes) used for the implementation

- PR: Presence Required (X=Yes, blank=No)
- Content: Possible values for the field
- Comment: Field description

11.2.1 Reference Information

The FIX message layout used for sending instrument and maintenance reference data is SecurityDefinition.

11.2.1.1 Instrument Reference Data Information

INSTRUMENT REFERENCE DATA MESSAGE LAYOUT							
Enhanced Broadcast Solution	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	SendingTime (52)		long	4	X	Milliseconds since midnight	FIX defines the data type of time as UTC-Timestamp but more efficient formats can be used for representation when sending FIX over FAST.
srcId	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
seqNum	MsgSeqNum (34)		long	4	X		Transmission sequence number.
isix	SecurityIDSource (22) SecurityID (48)		long	4	X	For FIX: 22=M	Instrument identifier.
isin	SecurityIDSource (22) SecurityID (48)		char	12	X	For FIX: 22=4	ISIN. International Security Identification Number.

INSTRUMENT REFERENCE DATA MESSAGE LAYOUT							
Enhanced Broadcast Solution	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
instMnem	Symbol (55)		char	5	X		The abbreviated form of the instrument name.
exchId	SecurityExchange (207)		char	4	X	For a list of values, see 7.8	International standard (ISO 10383) for codes for exchanges and market identification (MIC).
instGrp	SecurityGroup (1151)		char	4	X	See 7.11	Identifies an instrument group.
instTypCod	CFICode (461)		char	3	X	See 7.13	Instrument type.
currCode	Currency (15)		char	3	X	For a list of values, see 7.5	International standard (ISO 4217) for currency names.
ticSiz	MinPriceIncrement (969)		long	4	X		Tick size, minimum price increase.
setId	SecondarySecurity-Group (TBD)	Y	long	4	X	Instrument Set ID	Technical grouping of instruments.
noOfStreams	NoMDFeedTypes (1141)		long	4	X		Number of streams for current instrument.
>streamType	MDFeedType (1022)		char	1	X	For a list of values, see 7.28	Information sent via this stream.
>streamService	MDFeedService (TBD)[1]	Y	char	1	X	"A" or "B"	Live-live concept implementation (Service A or B).

INSTRUMENT REFERENCE DATA MESSAGE LAYOUT							
Enhanced Broadcast Solution	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>inetAddr	MDFeedAddress (TBD)[1]	Y	char	15	X	nnn.nnn.nnn.nnn	Total number of IP addresses that might be sent is 14, 2 for the two live-live snapshot streams, 2 for the two live-live All Trade Price and 10 for the delta streams live-live (5 for each).
>port	MDFeedPort (TBD)[1]	Y	long	4	X		UDP port number. Always relates to an IP address.
>mktDepth	MarketDepth (264)		long	4	X		Max depth of corresponding channel. Not sent for All Trade Price.
>mdBookType	MDBookType (1021)		long	4		For FIX: 1021=2	Added to support FIX compliance. Will be sent for Snapshots and Delta. Not sent for All Trade Price.

[1] Suggested custom field name, currently not standard FIX field.

11.2.1.2 Maintenance Reference Data Information

MAINTENANCE REFERENCE DATA MESSAGE LAYOUT							
Enhanced Broadcast Solution	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	SendingTime (52)		long	4	X	Milliseconds since midnight	FIX defines the data type of time as UTC-Timestamp but more efficient formats can be used for representation when sending FIX over FAST.
srcId	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
seqNum	MsgSeqNum (34)		long	4	X		Transmission sequence number.
exchId	SecurityExchange (207)		char	4	X	For a list of values, see 7.8	International standard (ISO 10383) for codes for exchanges and market identification (MIC).
noOfStreams	NoMDFeedTypes		long	4	X	2	Number of streams for current instrument.
>streamService	MDFeedService (TBD)[1]	Y	char	1	X	"A" or "B"	Live-live concept implementation (Service A or B).
>inetAddr	MDFeedAddress (TBD)[1]	Y	char	15	X	nnn.nnn.nnn.nnn	IP address of the All Trade Price, corresponding stream. Total number of IP addresses that might be sent is 14, 2 for the two live-live snapshot streams, 2 for the two live-live All Trade Price and 10, 5 for each delta live-live.

MAINTENANCE REFERENCE DATA MESSAGE LAYOUT							
Enhanced Broadcast Solution	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>port	MDFeedPort (TBD)[1]	Y	long	4	X		UDP port number. Always relates to an IP address.

[1] Suggested custom field name, currently not standard FIX field.

11.2.2 Order Book Information

11.2.2.1 Order Book Snapshot

All snapshot messages are based on the standard FIX message MarketDataSnapshotFullRefresh.

For a detailed description of the FIX field mapping and dependencies within a market data entry of a repeating group, please refer to section 11.2.2.3. It explains which fields should be filled with which values for a specific market data entry type.

The status of the instrument is comprised in the field *prcsStsValCod*, the compression isn't FIX compliant since it bonds two FIX hierarchy levels into one. It's a flattening of the structure defined in FIX with the TradingSessionID, TradingSessionSubId and SecurityTradingStatus. For a detailed description of the different instrument states and the closest mapping to FIX, refer to section 7.12.

The total number of snapshot data entries in this message is the sum of the number of entries in each of the following five repeating groups, price-quantity (noEntriesPrcQty), price (noEntriesPrc), quantity (noEntriesQty), All Trade Price (noEntriesAtp) and orderbook (noEntriesDepth).

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	N.A.	SendingTime (52)		long	4	X	Timestamp	Timestamp for all market data entries in this message. FIX has additional field MDEntryTime (273) within the repeating group to allow different timestamps which are not needed here.
srcId	N.A.	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
isix	N.A.	SecurityID (48) SecurityIDSource (22)		long	4	X	For FIX: 22=”M”	Instrument identifier.
Start of Repeating Group for Delta Channels								
noOfSeqNum	N.A.	NoOfMsgSeqNums (TBD)[1]	Y	long	4	X	5	Number of delta channels.
consolSeqNum	N.A.	LastMsgSeqNum Processed (369)		long	4	X	Numeric	Sequence number to relate to the sequence number sent in the delta message.
End of Repeating Group for Delta Channels								
lastTpSeqNum	N.A.	ApplID (1180) ApplLastSeqNum (1350)		long	4		Numeric For FIX: 1180="Trade"	Sequence number to relate to All Trade Price messages.

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
instrStatus	prcsStsValCod	TradingSessionSubID (625) SecurityTradingStatus (326) SecurityType (167)		char	1	X	For a list of valid values and a mapping to FIX, refer to 7.12.	Current status of an instrument.
moiInd	moiInd	SecurityTradingEvent (1174)[2]		char	1		For a list of valid values, refer to 7.16.	Market order interruption indicator.
volInd	volInd			char	1		For a list of valid values, refer to 7.35.	Indicates whether an instrument has a potential volatility interruption or an actual volatility interruption or neither during auction.
cmexInd	cmexInd	CorporateAction (292)		char	1		For a list of valid values, refer to 7.4.	Indicates whether the last trade price was influenced by a capital adjustment.
Start of Repeating Group **EntriesPrcQty**, FIX Component Block <MDFullGrp>								
noEntriesPrcQty	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	PTNL_AUCT_PRC MTCH_RNG_ASK MTCH_RNG_BID	Indicates the kind of price and quantity the fields entryPrc and entryQty are carrying. To see the mapping to FIX of this value, refer to 11.2.2.3.
>entryPrc	auctPrc	MDEntryPx (270)		int64	8	X	Price	Price determined during auction call or at orderbook balancing.

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryQty	auctQty	MDEntrySize (271)		int64	8	X	Quantity	Quantity that could be traded at the auction price calculated during auction call phase.
End of Repeating Group **EntriesPrcQty**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesPrc**, FIX Component Block <MDFullGrp>								
noEntriesPrc	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	CLOSE_PRC VAL_PRC OPENING_PRC HIGH_PRC_ENTRY LOW_PRC_ENTRY	Indicates what kind of price is entryPrc. To see the mapping to FIX of these values, refer to 11.2.2.3.
>entryPrc	clsPrc	MDEntryPx (270)		int64	8	X	Price	Most recent closing price (end of day auction).
	valPrc							Intra day closing auction price.
	opnPrc							Most recent opening price (opening auction).
	dlyHghPrc							Highest trade price of current business day.[3]
	dlyLowPrc							Lowest trade price of current business day.[3]

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
End of Repeating Group **EntriesPrc**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesQty**, FIX Component Block <MDFullGrp>								
noEntriesQty	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	SRP_ASK SRP_BID	Indicates what kind of price is entryQty. To see the mapping to FIX of these values, refer to 11.2.2.3.
>entryQty	srpQtyAsk	MDEntrySize (271)		int64	8	X	Quantity	The surplus quantity for the ask side.
	srpQtyBid							The surplus quantity for the bid side.
End of Repeating Group **EntriesQty**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesAtp**, FIX Component Block <MDFullGrp>								
noEntriesAtp	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	LAST_TRD_PRC CROSSED_PRC LAST_AUC_PRC	Describes what kind of price and quantity the fields entryPrc, entryQty and entryTime are carrying. To see the mapping to FIX of these values, refer to 11.2.2.3.

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryPrc	lstTrdPrc	MDEntryPx (270)		int64	8	X	Price	Most recent price of a completed trade.
	crossTrdPrc							Price of a crossed trade.
	lstAuctPrc							Auction price of a completed trade.
>entryQty	lstTrdQty	MDEntrySize (271)		int64	8	X	Quantity	Most recent recorded size of a completed trade.
	crossTrdQty							Quantity applied to a crossed trade.
	lstAuctQty							Quantity traded in the last auction.
>totTrdQty	instOneDayQty	TradeVolume (1020)		long	4		Numeric signed 9	Cumulative volume of units traded in the day.
>entryTim	tranTim	MDEntryTime (273)		char	8		Time	Time recorded on matching/execution of a buy and a sell order, forming a trade.
	crossTrdTim							Trade time of a crossed trade.
	auctTim							Time of the last auction.
>numTrades	noTotTrdQty	TotalNumOfTrades (6139)		long	4		Numeric signed 9	The cumulative total of the number of trades for the current day in a given instrument.
End of Repeating Group **EntriesAtp**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesDepth**, FIX Component Block <MDFullGrp>								

SNAPSHOT MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
noEntriesDepth	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	ASK_PRC BID_PRC EMPTY_BOOK ASK_PRC_MARKET BID_PRC_MARKET	Indicates whether the fields entryPrc, entryQty and numOrders refer to an ask or a bid entry or if the orderbook is empty. To see the mapping to FIX of these values, refer to 11.2.2.3
>entryPrc	bstBidPrc	MDEntryPx (270)		int64	8	X	Price	Bid price at given level entryPrcLvl.
	bstAskPrc							Ask price at given level entryPrcLvl.
>entryQty	bstBidQty	MDEntrySize (271)		int64	8	X	Quantity	Total quantity of all orders which have the best bid price.
	bstAskQty							Total quantity of all orders which have the best ask price.
>numOrders	numOrdrBid	NumberOfOrders (346)		long	4	X	Numeric signed 4	Number of orders per limit on the bid side.
	numOrdrAsk							Number of orders per limit on the ask side.
>entryPrcLvl	N.A.	MDPriceLevel (1023)		long	4	X	Numeric	Depth of entry in orderbook.
End of Repeating Group **EntriesDepth**, FIX Component Block <MDFullGrp>								

[1] Suggested custom field name, currently not a standard FIX field.
[2] Standard FIX tag that is currently not part of this message.
[3] Only sent with new value caused by update.

11.2.2.2 Order Book Delta / Incremental

All delta messages are based on the standard FIX message MarketDataIncrementalRefresh.

The total number of delta data entries in this message is the sum of the number of entries in each of the following four repeating groups, price-quantity (noEntriesPrcQty), price (noEntriesPrc), quantity (noEntriesQty) and orderbook (noEntriesDepth).

DELTA / INCREMENTAL MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	N.A.	SendingTime (52)		long	4	X	Timestamp	Timestamp for all market data entries in this message. FIX has additional field MDEntryTime (273) within the repeating group to allow different timestamps which are not needed here.
srcId	N.A.	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
isix	N.A.	SecurityID (48) SecurityIDSource (22)		long	4	X	For FIX: 22="M"	Instrument identifier. In FIX, this field is inside the repeating group.
seqNum	N.A.	MsgSeqNum (34)		long	4	X	Numeric	Sequence number for snapshots to be able to reference the last delta message they include.
lastTpSeqNum	N.A.	ApplID (1180) ApplLastSeqNum (1350)		long	4		Numeric For FIX: 1180="Trade"	Sequence number to relate to All Trade Price messages.

DELTA / INCREMENTAL MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
instrStatus	prcsStsValCod	SecurityTradingStatus (326)		char	1	X	For a list of valid values and a mapping to FIX, refer to 7.12.	Current status of an instrument.
moiInd	moiInd	SecurityTradingEvent (1174)[1]		char	1		For a list of valid values, refer to 7.16.	Market order interruption indicator.
volInd	volInd			char	1		For a list of valid values, refer to 7.35.	Indicates whether an instrument has a potential volatility interruption or an actual volatility interruption or neither during auction.
cmexInd	cmexInd	CorporateAction (292)		char	1		For a list of valid values, refer to 7.4.	Indicates whether the last trade price was influenced by a capital adjustment.
gapIndicator	N.A.	RefreshIndicator (1187)		char	1		For FIX, 1187 = "Y"	When this gap is set, it indicates that some prior trades may not have been reported.
Start of Repeating Group **EntriesPrcQty**, FIX Component Block <MDFullGrp>								
noEntriesPrcQty	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	PTNL_AUCT_PRC MTCH_RNG_ASK MTCH_RNG_BID	Indicates the kind of price and quantity the fields entryPrc and entryQty are carrying. To see the mapping to FIX of this value, refer to 11.2.2.3.

DELTA / INCREMENTAL MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryPrc	auctPrc	MDEntryPx (270)		int64	8	X	Price	Price determined during auction call or at orderbook balancing.
>entryQty	auctQty	MDEntrySize (271)		int64	8	X	Quantity	Quantity that could be traded at the auction price calculated during auction call phase.
End of Repeating Group **EntriesPrcQty**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesPrc**, FIX Component Block <MDFullGrp>								
noEntriesPrc	N.A.	noMDEntries (268)		long	4	X	Numeric	
>entryType	N.A.	MDEntryType (269)		long	4	X	CLOSE_PRC VAL_PRC OPENING_PRC HIGH_PRC_ENTRY LOW_PRC_ENTRY LAST_TRD_PRC LAST_AUC_PRC	Indicates what kind of price is entryPrc. To see the mapping to FIX of these values, refer to 11.2.2.3.
>entryPrc	auctPrc	MDEntryPx (270)		int64	8	X	Price	Price determined during auction call or at orderbook balancing.
End of Repeating Group **EntriesPrc**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesQty**, FIX Component Block <MDFullGrp>								
noEntriesQty	N.A.	noMDEntries (268)		long	4	X	Numeric	

DELTA / INCREMENTAL MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryType	N.A.	MDEntryType (269)		long	4	X	SRP_ASK SRP_BID TOT_TRD_QTY	Indicates what kind of price is entryQty. To see the mapping to FIX of these values, refer to 11.2.2.3.
>entryQty	auctQty	MDEntrySize (271)		int64	8	X	Quantity	Quantity that could be traded at the auction price calculated during auction call phase.
End of Repeating Group **EntriesQty**, FIX Component Block <MDFullGrp>								
Start of Repeating Group **EntriesDepth**, FIX Component Block <MDFullGrp>								
noEntriesDepth	N.A.	noMDEntries (268)		long	4	X	Numeric	Number of entries for the orderbook data
>updateAction	N.A.	MDUpdateAction (279)		char	1	X	New Change Delete Delete thru Delete from	Change only applies to the quantity. Delete From deletes all price levels >= n Delete Thru deletes all price levels from 1 to n and all price levels m>n are shifted to m-n.
>entryType	N.A.	MDEntryType (269)		char	1	X	ASK_PRC BID_PRC EMPTY_BOOK ASK_PRC_MARKET BID_PRC_MARKET	Indicates whether the fields entryPrc, entryQty and numOrders refer to an ask or a bid entry or if the orderbook is empty. To see the mapping to FIX of these values, refer to 11.2.2.3.
>entryPrc	bstBidPrc	MDEntryPx (270)		int64	8	X	Price	Bid price at given level entryPrcLvl.
	bstAskPrc							Ask price at given level entryPrcLvl.

DELTA / INCREMENTAL MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryQty	bstBidQty	MDEntrySize (271)		int64	8	X	Quantity	Total quantity of all orders which have the best bid price.
	bstAskQty							Total quantity of all orders which have the best ask price.
>numOrders	numOrdrBid	NumberOfOrders (346)		long	4	X	Numeric signed 4	Number of orders per limit on the bid side.
	numOrdrAsk							Number of orders per limit on the ask side.
>entryPrcLvl	N.A.	MDPriceLevel (1023)		long	4	X	Numeric	Depth of entry in orderbook.
End of Repeating Group **EntriesDepth**, FIX Component Block <MDFullGrp>								

[1] Standard FIX tag that is currently not part of this message.

11.2.2.3 Entry Type Mapping to FIX

The following table shows which FIX fields should be filled and with which value for each of the different entry types. The information contained in the single Enhanced Broadcast Solution field does not always correspond to a single FIX MDEntryType and requires additional FIX fields (TradeCondition, QuoteCondition) to be used.

Enhanced Broadcast Solution	FIX Fields and Values							
	MDEntryType (269)	MDEntryPx (270)	MDEntry Size (271)	MDEntry Time (273)	NumberOf Orders (346)	TradeCondition (277)	QuoteCondition (276)	OrdType (40)
EMPTY_ENTRY					-	-	-	
ASK_PRC	1 = Ask / Offer	Price	Quantity	-	<value>	-	-	2 = Limit
BID_PRC	0 = Bid	Price	Quantity	-	<value>	-	-	2 = Limit
EMPTY_BOOK	J = Empty Book				-	-	-	-
LAST_TRD_PRC	2 = Trade	Price	Quantity	Time	-	U=Exchange Last	-	-
OPENING_PRC	4 = Opening Price	Price	-	-	-	-	-	-
LAST_AUC_PRC	Q = Auction Clearing Price	Price	Quantity	Time	-	-	-	-
VAL_PRC[1]	Intraday + 5 = Closing Price	Price	-	-	-	-	-	-
CLOSE_PRC	5 = Closing Price	Price	-	-	-	-	-	-
CROSSED_PRC	2 = Trade	Price	Quantity	Time	-	X = Crossed	-	-
PTNL_AUCT_PRC	Q = Auction Clearing Price	Price	Quantity	-	-	-	I = Non-Firm[2]	-
MTCH_RNG_ASK	1 = Ask / Offer	Price	-	-	-	-	F = CROSSED	-

Enhanced Broadcast Solution	FIX Fields and Values							
	MDEntryType (269)	MDEntryPx (270)	MDEntry Size (271)	MDEntry Time (273)	NumberOf Orders (346)	TradeCondition (277)	QuoteCondition (276)	OrdType (40)
MTCH_RNG_BID	0 = Bid	Price	-	-	-	-	F = CROSSED	-
SRP_BID	0 = Bid	-	Quantity	-	-	-	Z=Order Imbalance	-
SRP_ASK	1 = Ask / Offer	-	Quantity	-	-	-	Z=Order Imbalance	-
DAY_QTY		-	Quantity	-	-	-	-	-
HIGH_PRC_ENTRY	N = Session High Bid	Price	-	-	-	-	-	-
LOW_PRC_ENTRY	O = Session Low Offer	Price	-	-	-	-	-	-
TOT_TRD_QTY	B = Trade Volume	-	Quantity	-	-	-	-	-
ASK_PRC_MARKET	1 = Ask / Offer	-	-	-	-	-	-	1 = Market
BID_PRC_MARKET	0 = Bid	-	-	-	-	-	-	1 = Market

[1] valPrc is the intraday closing auction price. To know it's intraday, we refer to the instrument state in prcsStsValCod
[2] QuoteCondition will be set to "I" when the price is still an indicative price. When the auction is closed, the price will be an Auction Clearing Price

11.2.3 All Trade Price Information

The FIX message format used for this stream is the same as for the snapshots, MarketDataSnapshotRefresh.

ALL TRADE PRICE MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	tranTime	SendingTime (52)		long	4	X	Timestamp	Timestamp for the All Trade Price message. Individual trades have their own timestamp within the repeating group, field MDEntryTime (273).
srcId	N.A.	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
isix	N.A.	SecurityID (48) SecurityIDSource (22)		long	4	X	For FIX: 22="M"	Instrument identifier.
gapIndicator	N.A.	RefreshIndicator (1187)		char	1		For FIX, 1187 = "Y"	When this gap is set, it indicates that some prior trades may not have been reported.
Start of Repeating Group								
noEntries	N.A.	noMDEntries (268)		long	4	X	Numeric	Number of repeating groups in message
>entryType	prcTypInd	MDEntryType (269)		long	4	X	LAST_TRD_PRC CROSSED_PRC BEST_PRC MPO_PRC	Indicates what type of trade it is.
>entryPrc	tradMtchPrc	MDEntryPx (270)		int64	8	X	Price	Matching price

ALL TRADE PRICE MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
>entryQty	trdQty	MDEntrySize (271)		int64	8	X	Quantity	Quantity of trade
>entryTime	N.A.	MDEntryTime (273)		char	8	X	Time	Time of trade
> tranMtchIdNo	N.A.	TradeMatchID (880)		long	4		Numeric	Match identifier
>tpSeqNum	N.A.	MsgSeqNum (34)		long	4	X	Numeric	Sequence number for deltas to be able to reference each trade information they include.
>actnCod	N.A.	TradeCondition (277)		char	1		See action code type	This field indicates what type of action should be performed in case of last trade price information.
End of Repeating Group								

11.2.4 State Changes

The FIX message layout used is TradingSessionStatus.

STATE CHANGES MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
timestamp	tranTime	SendingTime (52)		long	4	X	Timestamp	Timestamp for the change in this message.

STATE CHANGES MESSAGE LAYOUT								
Enhanced Broadcast Solution	**VALUES**	**FIX Field (FIX Tag)**	**FG**	**Type**	**Size**	**PR**	**Content**	**Description**
srcId	N.A.	SenderCompID (49) SenderSubID (50)		long	4	X	For FIX: 49=Xetra	Logical node number.
state	N.A.	TradingSessionSubID (625)		long	4		For a list of possible values, see section 7.25. For FIX: 625 = 1	*TradingSessionSubID* is a required field in this type of message, it will always represent Day. Exchange and system states are combined in this field whereas they need to be separated in FIX. Exchange states map to TradSesStatus and system states (for fast market information) to TradSesEvent. TradSesStatus is a required field and can be set to 0 (unknown) to indicate system states.
		TradSesStatus (340)				X		
		TradSesEvent (1368)						

To be able to do the complete mapping to FIX for the field TradSesStatus, two user-defined values have to be defined (system batch and start of Trading Model Continuous Auction). The following table lists the values and shows the mapping.

Enhanced Broadcast Solution Value	Description	TradSesStatus (340)
1	Exchange Online	100 (user-defined value)
2	Exchange in Batch	101 (user-defined value)
3	Exchange Halted	1 (Halted)
4...11	Fast Market On/Off	0 (Unknown)
12	Exchange Start	102 (user-defined value)
13	Exchange Pre-trade	4 (Pre-Open)
14	Exchange Trade	2 (Open)
15	Exchange End-trade	5 (Pre-Close)
16	Exchange Post-trade	3 (Closed)
17	Exchange Halt	1 (Halted)

The field TradSesEvent is not mandatory in the message, it will only be sent, when dealing with a system state change. A whole list of new user-defined values is necessary for the mapping, as shown in table below:

Enhanced Broadcast Solution Value	Description	TradSesEvent (1368)
4	Fast Market Off for Equities	100
5	Fast Market On for Equities	101
6	Fast Market Off for Bonds	102
7	Fast Market On for Bonds	103
8	Fast Market Off for Warrants	104
9	Fast Market On for Warrants	105
10	Fast Market Off for Basis	106
11	Fast Market On for Basis	107

12. Appendix - FAST Message Templates

There are various ways in which the Xetra Enhanced Broadcast Solution message layouts could be described. FAST Market Data Optimization Working Committee provides message templates in many layouts such as Compact Notation, XML notation, etc.

This section describes the Xetra Enhanced Broadcast Solution message templates in XML format. Please note that:

- These are only additional formats of message representation and the tabular representation of messages described in section 5 (Structure of Messages) should be considered as master layouts in case of discrepancies.
- Fields not explicitly mentioned as optional are mandatory, unless they are suppressed by FAST compression.

Since we are suggesting a mapping to FIX in section 11, auxiliary identifiers have been used to specify the FIX tag number for each field where available (only for the main messages). Suggested user-defined fields (tag number "TBD") can be given a tag number above 10000 within the template. Enhanced Broadcast Solution fields that map to multiple FIX fields need special handling during the mapping process.

XML Notation

```
<!-- FAST Message Templates -->
<templates ns="https://business.deutsche-boerse.com/irj/portal/EnhancedBroadcastSolution"
        xmlns="http://www.fixprotocol.org/ns/fast/td/1.1">

   <!-- FAST Reset Message -->
   <template name="FASTReset" id="120"/>

   <!-- Version Information Message -->
   <template name="VersionInformation" id="1">
            <typeRef name="versioninformation"/>
            <uInt32 name="versNo"/>
            <uInt32 name="srcId"/>
            <uInt32 name="seqNum"/>
   </template>

   <!-- Start of Xetra Service Message -->
   <template name="StartService" id="128">
            <typeRef name="startservice"/>
            <uInt32 name="timestamp"/>
            <string name="busDate"/>
```

```
</template>

<!-- End of Xetra Service Message -->
<template name="EndService" id="129">
        <typeRef name="endservice"/>
        <uInt32 name="timestamp"/>
        <string name="busDate"/>
</template>

<!-- Start of Instrument Reference Data Message -->
<template name="StartRefData" id="130">
        <typeRef name="startreference"/>
        <uInt32 name="timestamp"/>
        <string name="busDate"/>
        <uInt32 name="noOfMsg"/>
</template>

<!-- End of Instrument Reference Data Message -->
<template name="EndRefData" id="131">
        <typeRef name="endreference"/>
        <uInt32 name="timestamp"/>
        <string name="busDate"/>
        <uInt32 name="noOfMsg"/>
</template>

<!-- Start of Maintenance Reference Data Message -->
<template name="StartMtnData" id="132">
        <typeRef name="startreference"/>
        <uInt32 name="timestamp"/>
        <string name="busDate"/>
        <uInt32 name="noOfMsg"/>
</template>

<!-- End of Maintenance Reference Data Message -->
<template name="EndMtnData" id="133">
        <typeRef name="endreference"/>
        <uInt32 name="timestamp"/>
        <string name="busDate"/>
```

```
        <uInt32 name="noOfMsg"/>
</template>

<!-- Beacon Message -->
<template name="BeaconMessage" id="2">
        <typeRef name="beaconmessage"/>
        <uInt32 name="timestamp">
                <delta/>
        </uInt32>
        <uInt32 name="srcId">
                <copy/>
        </uInt32>
        <uInt32 name="seqNum"/>
        <uInt32 name="isix">
                <delta/>
        </uInt32>
</template>

<!-- Xetra Instrument Reference Data Message -->
<template name="InstrumentReferenceData" id="3">
        <typeRef name="instrumentreferencedata"/>
        <uInt32 name="timestamp" id="52">
                <delta/>
        </uInt32>
        <uInt32 name="srcId" id="50">
                <copy/>
        </uInt32>
        <uInt32 name="seqNum" id="34">
                <increment value="1"/>
        </uInt32>
        <uInt32 name="isix" id="48">
                <delta/>
        </uInt32>
        <string name="isin" id="455">
                <delta/>
        </string>
        <string name="instMnem" id="55"/>
        <string name="exchId" id="207">
```

```
            <copy/>
        </string>
        <string name="instGrp" id="1151">
            <copy/>
        </string>
        <string name="instTypCod" id="461">
            <copy/>
        </string>
        <string name="currCode" id="15">
            <copy/>
        </string>
        <decimal name="ticSiz" id="969">
            <delta/>
        </decimal>
        <uInt32 name="setId" id="TBD">
            <copy/>
        </uInt32>
        <sequence name="MDFeedTypes">
            <length name="noOfStreams" id="1141"/>
            <string name="streamType" id="1022"/>
            <string name="streamService"/>
            <string name="inetAddr">
                <delta/>
            </string>
            <uInt32 name="port" id="TBD">
                <delta/>
            </uInt32>
            <uInt32 name="mktDepth" id="264" presence="optional"/>
            <uInt32 name="mdBookType" id="1021" presence="optional"/>
        </sequence>
    </template>

    <!-- Maintenance Reference Data Message -->
    <template name="MaintenanceReferenceData" id="4">
        <typeRef name="maintenancereferencedata"/>
        <uInt32 name="timestamp" id="52">
            <delta/>
        </uInt32>
```

```
            <uInt32 name="srcId" id="50">
                <copy/>
            </uInt32>
            <uInt32 name="seqNum" id="34">
                <increment value="1"/>
            </uInt32>
            <string name="exchId" id="207">
                <copy/>
            </string>
            <sequence name="MDFeedTypes">
                <length name="noOfStreams" id="1141"/>
                <string name="streamService"/>
                <string name="inetAddr">
                    <delta/>
                </string>
                <uInt32 name="port" id="TBD">
                    <delta/>
                </uInt32>
            </sequence>
        </template>

        <!-- Snapshot Message -->
        <template name="InsideMarketSnapshotInformation" id="6">
            <typeRef name="insidemarketsnapshot"/>
            <uInt32 name="timestamp" id="52">
                <delta/>
            </uInt32>
            <uInt32 name="srcId" id="50">
                <copy/>
            </uInt32>
            <uInt32 name="isix" id="48">
                <delta/>
            </uInt32>
            <sequence name="NoOfChannelSeqNum">
                <length name="noOfSeqNum"/>
                <uInt32 name="consolSeqNum" id="369">
                    <delta/>
                </uInt32>
```

```
            </sequence>
            <uInt32 name="lastTpSeqNum" id="1350">
                  <delta/>
            </uInt32>
            <uInt32 name="instrStatus" id="326"/>
            <string name="moiInd" id="1174" presence="optional"/>
            <string name="volInd" id="1174" presence="optional"/>
            <string name="cmexInd" id="292" presence="optional"/>
            <sequence name="EntriesPrcQty">
                  <length name="noEntriesPrcQty" id="268"/>
                  <uInt32 name="entryType" id="269"/>
                  <decimal name="entryPrc" id="270">
                        <delta/>
                  </decimal>
                  <decimal name="entryQty" id="271">
                        <delta/>
                  </decimal>
            </sequence>
            <sequence name="EntriesPrc">
                  <length name="noEntriesPrc" id="268"/>
                  <uInt32 name="entryType" id="269"/>
                  <decimal name="entryPrc" id="270">
                        <delta/>
                  </decimal>
            </sequence>
      <sequence name="EntriesQty">
                  <length name="noEntriesQty" id="268"/>
                  <uInt32 name="entryType" id="269"/>
                  <decimal name="entryQty" id="271">
                        <delta/>
                  </decimal>
            </sequence>
            <sequence name="EntriesAtp">
                  <length name="noEntriesAtp" id="268"/>
                  <uInt32 name="entryType" id="269"/>
                  <decimal name="entryPrc" id="270">
                        <delta/>
                  </decimal>
```

```
                <decimal name="entryQty" id="271">
                    <delta/>
                </decimal>
                <decimal name="totTrdQty" id="1020">
                    <delta/>
                </decimal>
                <string name="entryTime" id="273">
                    <delta/>
                </string>
                <uInt32 name="numTrades" id="6139">
                    <delta/>
                </uInt32>
            </sequence>
            <sequence name="EntriesDepth">
                <length name="noEntriesDepth" id="268"/>
                <uInt32 name="entryType" id="269"/>
                <decimal name="entryPrc" id="270">
                    <delta/>
                </decimal>
                <decimal name="entryQty" id="271">
                    <delta/>
                </decimal>
                <uInt32 name="numOrders" id="346">
                    <delta/>
                </uInt32>
                <uInt32 name="entryPrcLvl" id="1023"/>
            </sequence>
        </template>

        <!-- Delta / Incremental Message -->
        <template name="InsideMarketDeltaInformation" id="7">
            <typeRef name="insidemarketdeltamessage"/>
            <uInt32 name="timestamp" id="52">
                <delta/>
            </uInt32>
            <uInt32 name="srcId" id="50">
                <copy/>
            </uInt32>
```

```
<uInt32 name="isix" id="48">
        <delta/>
</uInt32>
<uInt32 name="seqNum" id="34">
        <delta/>
</uInt32>
<uInt32 name="lastTpSeqNum" id="1350">
        <delta/>
</uInt32>
<uInt32 name="instrStatus" id="326"/>
<string name="moiInd" id="1174" presence="optional"/>
<string name="volInd" id="1174" presence="optional"/>
<string name="cmexInd" id="292" presence="optional"/>
<string name="gapIndicator" id="1187" presence="optional">
        <constant value="Y"/>
</string>
<sequence name="EntriesPrcQty">
        <length name="noEntriesPrcQty" id="268"/>
        <uInt32 name="entryType" id="269"/>
        <decimal name="entryPrc" id="270">
                <delta/>
        </decimal>
        <decimal name="entryQty" id="271">
                <delta/>
        </decimal>
</sequence>
<sequence name="EntriesPrc">
        <length name="noEntriesPrc" id="268"/>
        <uInt32 name="entryType" id="269"/>
        <decimal name="entryPrc" id="270">
                <delta/>
        </decimal>
</sequence>
<sequence name="EntriesQty">
        <length name="noEntriesQty" id="268"/>
        <uInt32 name="entryType" id="269"/>
        <decimal name="entryQty" id="271">
                <delta/>
```

```
                </decimal>
            </sequence>
            <sequence name="EntriesDepth">
                <length name="noEntriesDepth" id="268"/>
            <uInt32 name="entryType" id="269"/>
                <decimal name="entryPrc" id="270">
                    <delta/>
                </decimal>
                <decimal name="entryQty" id="271">
                    <delta/>
                </decimal>
                <uInt32 name="numOrders" id="346">
                    <delta/>
                </uInt32>
                <uInt32 name="entryPrcLvl" id="1023"/>
                <uInt32 name="updateAction" id="279"/>
            </sequence>
        </template>

        <!- All Trade Price Message -->
        <template name="AllTradePrice" id="9">
            <typeRef name="atpmessage"/>
            <uInt32 name="timestamp" id="52">
                <delta/>
            </uInt32>
            <uInt32 name="srcId" id="50">
                <copy/>
            </uInt32>
            <uInt32 name="isix" id="48">
                <delta/>
            </uInt32>
            <string name="gapIndicator" id="1187" presence="optional">
                <constant value="Y"/>
            </string>
            <sequence name="EntriesAtp">
                <length name="noEntriesAtp" id="268"/>
                <uInt32 name="entryType" id="269"/>
                <decimal name="entryPrc" id="270">
```

```
                        <delta/>
                    </decimal>
                    <decimal name="entryQty" id="271">
                        <delta/>
                    </decimal>
                    <string name="entryTime" id="273">
                        <delta/>
                    </string>
                    <uInt32 name="tranMtchIdNo" id="880">
                        <delta/>
                    </uInt32>
                    <uInt32 name="tpSeqNum" id="34">
                        <delta/>
                    </uInt32>
                    <uInt32 name="actnCod" id="277"/>
                </sequence>
            </template>

            <!-- State Changes Message -->
            <template name="StateChangesMessage" id="10">
                <typeRef name="statechangesmessage"/>
                <uInt32 name="timestamp" id="52">
                    <delta/>
                </uInt32>
                <uInt32 name="srcId" id="50">
                    <copy/>
                </uInt32>
                <uInt32 name="state" id="340">
                    <copy/>
                </uInt32>
            </template>

        </templates>
```