DEUTSCHE BÖRSE
CASH MARKET

T7 Release 5.0

Market Data and Reference Data Interfaces

An Introduction

Version	1.0
Date	30 September 2016

Abstract

This document provides first information for Xetra Members regarding the Migration to T7. The purpose of this document is to provide an introduction of the T7 Market and Reference Data Interfaces to participants.

Keywords

Xetra, T7, EMDI, MDI, EOBI, RDI, RDF

Table of Contents:

1 List of Abbreviations, Acronyms and Definitions

Please find a list of all the abbreviations used in the document. The first time an abbreviation is introduced in the document it is written in brackets after the phrase.

ETI	Enhanced Trading Interface
MDI	Market Data Interface
EMDI	Enhanced Market Data Interface
EOBI	Enhanced Order Book Interface
RDI	Reference Data Interface
RDF	Reference Data File
CRE	Common Report Engine
FAST	FIX Adapted for Streaming (FAST Protocol) (FAST Protocol℠). FIX Adapted for streaming is a standard which has been developed by the Data Representation and Transport Subgroup of FPLs Market Data Optimization Working Group. FAST uses proven data redundancy reductions that leverage knowledge about data content and data formats
FIX	Financial Information eXchange. The Financial Information eXchange (“FIX”) Protocol is a series of messaging specifications for the electronic communication of trade-related messages
Trader GUI	Graphical User Interface for access to trading functions for orders and OTC trade entry
Admin GUI	Graphical User Interface for Administration functions, for example, user maintenance
Participant	The Xetra concept of a member will be represented by a “participant” within T7
Xetra®	Cash Market Electronic Trading system of DBAG
T7	Trading System developed by Deutsche Börse Group

2 Introduction

Xetra offers public market and reference data via three interfaces as part of the T7 architecture. All interfaces distribute information via UDP multicast; following FIX 5.0 SP2 semantics and are FAST 1.2 encoded (except EOBI). If any messages are lost, complete recovery is possible because every message is published on two identical services (A and B) with different multicast addresses (live-live principle). In the unlikely case that a message is lost on both services, participants can take advantage of the respective snapshot message and rebuild the order book.

Please note that on T7 a product is equal to an instrument for the cash market in the most cases. However, ETF's and ETP's can be grouped as one product. In this document both terms will be used.

The Market Data Interfaces are:

- **Enhanced Order Book Interface (EOBI):** This interface provides the entire visible order book, by publishing information on each individual order and quote side, along with state information in real-time and in un-netted manner.
- **Enhanced Market Data Interface (EMDI)**: This interface provides un-netted price level aggregated market data. The updates of the order book are delivered for all order book changes up to a given price level; all on-exchange "Book" trades are reported individually.
- **Market Data Interface (MDI)**: This interface provides netted price level aggregated market data. Updates of the order book are sent at regular intervals; they are not provided for every order book change and are sent significantly less frequently than the EMDI. "Book" trades are not reported individually, but statistical information (daily high/low price, last trade price and quantity) is provided instead.

Additionally the netted and un-netted interfaces provide the following information to the participant:

- Instrument states
- Quote and Cross requests

Reference data is provided via:

- **Reference Data Interface (RDI)**: This interface provides reference data which is available for products and instruments traded on T7. Note that every tradable object is referenced by a unique identifier; hence the reference data information is absolutely essential for any trading application.
- **Reference Data File (RDF)**: Reference data is delivered as a start-of-day file.

2.1 Purpose

The document enables participants to understand the main concepts of the market and reference data interfaces in order to select the interfaces that fit their needs best. In addition it should help members and vendors to create a high level effort estimation of necessary changes to their software and infrastructure. It contains sufficient information for business units to make a decision between the following:

- The Enhanced Order Book Interface, Enhanced Market Data Interface and Market Data Interface
- The Reference Data Interface and Reference Data from File

Differences in functionality between EOBI, EMDI and MDI are described in separate sub chapters.

Since the objective of this document is an introduction to the public interfaces, it does not describe the message layouts for all messages but instead focuses on the design principles of the interfaces for receiving market and reference data. A complete manual containing all message layouts will be made available at a later point in time on the Xetra website.

2.2 Readership

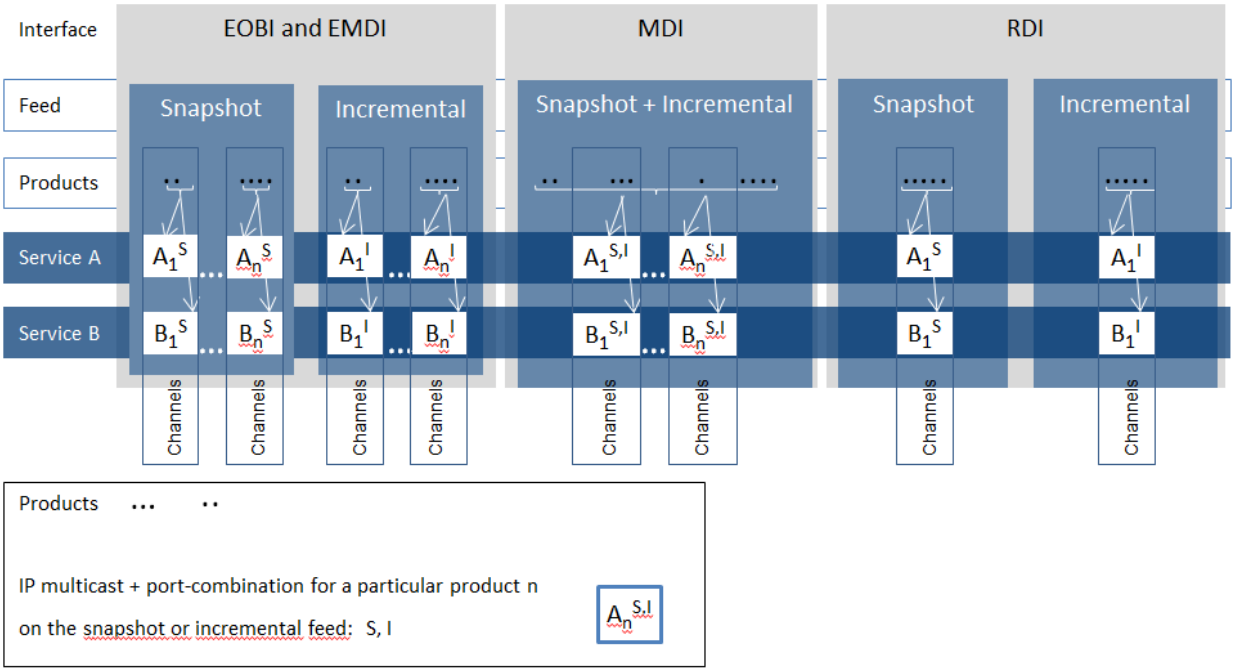
The target audience of this interface introduction is experienced software developer support staff that may be involved in development/support activities for the *T7 Market & Reference Data Interfaces*.

Prior knowledge of developing for a cash market is beneficial but not a prerequisite. Knowledge in a programming language is expected. This manual does not attempt to cover basic knowledge of programming techniques and software development.

3 Differences between the interfaces

The following figure illustrates the three multicast based public interfaces and their feeds. The reference data for each product in T7 is published on exactly one snapshot channel, indicated by (A_1^S) and one incremental channel (A_1^I). EOBI and EMDI have multiple channels that contain snapshots (A_1^S) to (A_n^S) or incremental updates (A_1^I) to (A_n^I). MDI has snapshots and incrementals combined over multiple channels ($A_1^{S,I}$) to ($A_n^{S,I}$).

Note that all channels are sent on two different multicast addresses via different physical connections (Service A and B). Service A and B are identical in terms of the provided information, e.g. the packet contents, sequence numbers and sequence in which packets are sent is the same.



The snapshot and incremental messages for the **EOBI and EMDI** are delivered via separate feeds (out-of-band) and need to be synchronized. Each feed consist of several channels, each of which delivers the information for a group of products.

In contrast to the un-netted market data, the snapshot and incremental messages for the **MDI** are sent on one feed only (in-band), therefore there is no need to synchronize both messages. The feed is also divided into several channels.

The snapshot feed for **RDI** is delivered via separate channels (out-of-band) and need to be synchronized. In contrast to the order book information the snapshot feeds are not divided into further channels. Please note that the RDI incremental channel is not applicable for the Cash market.

3.1 Choosing between EOBI, EMDI and MDI

All market data interfaces on T7 deliver information via multicast feeds.

EOBI provides the entire visible order book for an instrument, providing superior transparency and efficiency, together with high throughput at minimal latency.

EMDI and MDI, un-netted and netted, report a price-level aggregated order book, but they have different bandwidth requirements and service levels.

- **Enhanced Order Book Interface (un-netted)**[1] disseminates all visible orders and quotes without any depth restriction, when the order books are open, along with order executions and state information via incremental messages in un-netted manner. This interface is designed for participants that rely on low-latency and at a high throughput with a high band-width network. Furthermore, snapshot messages always carry existing visible orders and quotes without any depth restriction at the time of sending.
  In EOBI information is sent in form of fixed-length binary encoded messages.
- **Enhanced Market Data Interface (un-netted)** disseminates every order book change up to the configured depth and all on-exchange trades without netting the market data incremental feed. This interface is designed for participants that rely on low-latency order book updates and data completeness. The un-netted market data is partitioned over several channels; each channel provides information about a group of products. As the market becomes busier, the number of messages (and therefore bandwidth usage) increases.
- **Market Data Interface (netted)** has a lower bandwidth requirement compared to the un-netted version. This interface is designed for participants who do not need to see every order book update, this has the advantage of keeping the infrastructure costs low. Snapshot and incremental updates are sent via the same IP multicast address and port combination.

The following table shows the major differences between EOBI, EMDI and MDI:

Characteristic	EOBI	EMDI	MDI
Netting interval/market depth	Provides un-netted, entire visible order book information for all instruments.	All visible order book changes and on-exchange trades without netting. Fully configurable, similar to existing Xetra Enhanced Broadcast Solution.	Configurable netting interval in milliseconds.
Snapshot processing	Snapshot messages required for recovery only. Snapshot and incremental messages must be synchronized during start-of-day or during recovery.	Snapshot messages required for recovery only. Snapshot and incremental messages must be synchronized during recovery.	No synchronisation of messages required as in-band delivery is used but snapshot messages must be processed.
Snapshot/incremental sequence numbers	Individual sequence number range per product.	Individual sequence number range per product.	One sequence number range per product.

[1] available in co-locations only

Characteristic	EOBI	EMDI	MDI
Trade Volume Reporting	It can be calculated by using incrementals. Snapshot messages contain trade statistics/volume information for recovery.	Each on-exchange trade is reported separately. Trades appear first followed by order book updates in one template.	Statistical information (daily high/low and total traded quantity) and last trade per netting interval information only.
Packet Header	No explicit performance indicator. It can be calculated by using Matching Engine-In and packet header timestamp.	A Performance Indicator is provided for incremental messages within the Packet Header.	No Performance Indicator.
Functional beacon message	A functional beacon message on a product level + the last valid MsgSeqNum.	A *functional beacon* message on a product level + the last valid *MsgSeqNum*.	Snapshots act as *functional beacon* message, hence no *functional beacon* messages are provided.

3.2 Choosing between the Reference Data Interface (RDI) and Reference Data File (RDF)

The Reference data is provided via the **Reference Data Interface (RDI)** and in file form as compressed **Reference Data Files (RDF)** in FIXML-layout, updated regularly via the Common Report Engine (CRE).

The initial reference data file generated at start-of-day contains the “reference data snapshots” available from the previous day.

Participants have the choice between the two different reference data sources. However, it is assumed that bandwidth conscious users will use **RDF** for start-of-day processing and intraday re-starts. The reference data file is provided once the system is available.

The following table shows the main difference between the reference data messages and the reference data from File:

Area	RDI: Message based	RDF: File based
Reference Data	High bandwidth users can use the multicast based Reference Data Interface.	Low bandwidth users can use Start-Of-Day Reference Data Files.

4 Overview of the Public Interfaces on T7

This chapter describes the public market data information provided by the market and reference data interfaces.

4.1 Infrastructure requirements

Market and Reference Data Interfaces disseminate market information over the Xetra multicast network. A router which is capable of handling IP multicast is required for accessing this interface. The multicast management protocol is IGMPv2. When utilizing IGMPv3, the IGMPv2 compatibility mode must be enabled.

4.2 Overview of the various message types

The various message types can be divided into “Service Messages” and “Data Messages”.

4.2.1 RDI

Service messages:

- **Technical heartbeat message** is sent out periodically by the Xetra system on every multicast address and on a specific port assigned for the *technical heartbeat*; it consist of a FAST reset message only. The purpose of the heartbeat message is network related only (to keep Spanning Tree alive).
- **Functional beacon message** indicates the availability of exchange services (gap detection) on the RDI incremental channel (not applicable for Xetra).

Data messages:

- **Market data report message** flags the start and end of reference data. Each message is flagged by a start/stop event identifier.
- **Product and Instrument snapshot messages** contain snapshots of product and instrument specific reference data.

4.2.2 EOBI/EMDI/MDI

Service messages:

- **Technical heartbeat** is sent out on all multicast addresses and on a specific port of EOBI/EMDI/ MDI. The description is the same as for RDI.
- **Functional beacon message** (EOBI/EMDI) This message indicates the availability of exchange services (gap detection) on incremental channels. No functional heartbeats are sent for the MDI because the snapshots act as a functional heartbeat.

EMDI/MDI Data messages:

- **Depth snapshot message** is used to send a snapshot of the current state of the order book and statistical information about on-exchange trades. This message can be used whenever the order book needs to be rebuilt.
- **Depth incremental message** is used to get un-netted (EMDI) or netted (MDI) price level updates of the initial order book.
- **Product and Instrument state change messages** provide state information for a product or instrument.
- **Cross and quote request message** is sent once a participant announces the intention to enter a cross trade or quote request.

EOBI Data messages:

- Product Summary, Instrument Summary, Order Snapshot
- Order Add, Modify, Modify Same Priority, Delete and Mass Delete
- Execution Summary, Full and Partial Order Execution, Trade Reversal and Report
- Auction Clearing Price (indicative auction price) and Auction BBO
- Product and Instrument State Change Messages
- Cross and Quote Request Message, Top Of Book

4.3 What is not included in these interfaces

The following information is **not** provided via the new interfaces:

- For auctions, the best bid/ask prices are disseminated at price level 1. If an order book is crossed during auction, a potential auction price is calculated. Order book depths are **not** delivered during auctions, only top of book information is disseminated.
- Market Supervision News is **not** provided. This information is available via the ETI in recoverable form.
- Retransmission functionality is **not** provided, but recovery is possible from the respective other service (A or B). In case a message is lost a snapshot can be used to rebuild the order book.

4.4 FIX over FAST

FIX messages are sent out in FAST 1.2 encoded format. The receiving software decodes the FAST messages according to the FAST 1.2 rules.

After the decoding process, the actual FIX message can be built by applying the FIX structure to the decoded message.

Participants need a standard FAST template based decoder in order to be able to use the EMDI, MDI and RDI. Alternatively participants can use their own FAST decoder implementation.

In EOBI information is sent in form of fixed-length binary encoded messages.

4.5 Freedom of choice

The T7 market and reference data interfaces can be accessed using any platform capable of receiving multicast data feeds. Participants can use any operating system, compiler version or programming language in order to develop or use specific third party applications that are tailored to their requirements.

4.6 Testing

It is recommended to test the functionality application logic sufficiently in a simulation environment. Receiving applications must be able to cope appropriately with a variety of Xetra service fail-over scenarios. For this purpose, special test scenarios are offered in the simulation environment.

4.7 Hours of operation/availability of messages

The product state “Pre-Trading” (TradingSessionSubID (625) = 1) for many of the Xetra instruments begins at 7:30 CET.

“Post-Trading” (TradingSessionSubID (625) = 5) for some Xetra instruments lasts until 20:30 CET.

Xetra is available from approximately 6:00 CET. It is recommended to start applications between 6:30 CET and 7:20 CET.

Market data messages are sent from the time a product changes to the state “Start-Of-Day” and lasts until market and reference data services are shutdown. During that period depth snapshots are sent.

The reference data is independent to any product state, so it has its own schedule. Receiving applications are expected to stay connected from instrument state “Start-Of-Day” until instrument state “End-Of-Day”.

The following table provides further details about the availability of messages per instrument state:

State	Market Data Orderbook	Market Data State Info
Continuous	Yes	Yes
Auction	Yes	Yes
Freeze	Yes	Yes
Book	No	Yes
Restricted	No	Yes
Closed	No	Yes

5 Description of an entire trading day

This chapter describes an entire trading day, from the start until the end of trading. The following steps need to be taken to prepare for and to receive market data:

- Receive reference data at start of day.
- Build the initial order book.
- Update the order book.

5.1 Start of day operation

Before processing any market data, receiving applications need to retrieve technical and functional information via the RDI. Alternatively, reference data can be received in file format (RDF). A detailed description of the reference data feeds and messages is provided in section 8.1 Reference data messages.

At start-up, it is required to process reference data and create the initial order book baseline.

5.2 Receiving reference data via RDI at start of day

At the start of the business day, receiving applications need to join the static multicast address/port of the reference data interface in order to receive the following messages:

- **Product snapshot** to receive the product details and subsequent stream address information.
- **Instrument snapshot** to receive instrument details.

Port information and multicast addresses for the reference data feeds as well as the address ranges for market data will be published in the document “Exchange and Settlement Network Access”. Port information and multicast addresses for market data feeds are delivered as part of the reference data feeds.

Further detailed information about reference data is provided in section 8.2 General reference data rules. However, the basic steps in order to receive reference data are the following:

1. Listen to reference data snapshot feed. Ignore all the messages until you reach the market data report message denoting the beginning of a snapshot. Take note here of:
   - *MDReportCount (2536)*, containing the number of reference data snapshot messages in the initial snapshot cycle

   Process all messages of the snapshot until you encounter the *market data report* message denoting the end of the snapshot cycle.
2. Store the reference data information for future use.
3. Join the market data snapshot and incremental feed of EOBI/EMDI or the market data feed of MDI.
4. Leave the reference data snapshot feeds.

5.3 Receiving reference data from file (RDF) at start of day

Participants with low bandwidth connections may retrieve the start-of-day reference data in a file based format via the Common Report Engine.

5.4 Build the initial order book

Participants first have to build the initial order book. The order book has to be maintained per instrument.

5.4.1 Build the initial order book with EOBI

In order to build an initial order book, participants subscribe to the EOBI incremental and snapshot channels.

On the EOBI incremental channel, order messages are used by participants to maintain the order book, while explicit state change messages are provided to communicate current instrument state.

Instrument reference data information required to process the EOBI market data is provided by the RDI and/or RDF, similar to the current procedure for EMDI.

The following data is provided via the EOBI snapshot channel for a specific *LastMsgSeqNumProcessed*:

- Product State information,
- Instrument State information,
- Trade Statistics per instrument,
- All visible orders in the order book.

The visible orders are sent alternating between buy and sell sides, where orders at the same price level are sorted by order time priority from the oldest to the newest order. The visible order book is disseminated per price level in a zig-zag manner, meaning both the sides (Bid and Offer) at each price level are disseminated before moving on to the next price level. During an auction instrument state, EOBI snapshot channel will broadcast auction information, Best Bid and Offer (BBO) or the auction clearing price (indicative auction price).

5.4.2 Build the initial order book with EMDI

The following procedure is recommended for the EMDI:

1. Join the market data incremental feed for the instruments of interest, and buffer the market data incremental messages.
2. Join the corresponding snapshot feed.
3. Wait until the *depth snapshot* message for the required instrument is received and use this snapshot to build the order book baseline.
4. Discard all buffered *depth incremental* messages with a message sequence number less than or equal to the one indicated in field *LastMsgSeqNumProcessed (369)* of the *depth snapshot* message.
5. Apply all *depth incrementals* (buffered & subsequent) for this instrument with *MsgSeqNum* > *LastMsgSeqNumProcessed.*
6. Keep processing snapshots and repeating step 3 and 4 until all required instruments are received and no further depth incremental are being buffered.
7. Leave the snapshot feed, but stay connected to the incremental feed throughout the day, this is in order to receive updates to the current order book.

5.4.3 Build the initial order book with MDI

The following sequence is recommended for **MDI:**

1. Join the market data feed for the instruments of interest.
2. Wait until the snapshot message for the instruments are received. This gives the desired order book baseline.

The field *LastMsgSeqNumProcessed (369)* in MDI snapshots is not applicable because snapshots and incrementals are sent in-band and don't need to be synchronized with each other.

Note: MDI applications must process *depth snapshots* beside the *depth incrementals* because the snapshots might contain new information. If the *RefreshIndicator (1187)* is set to *Y = Mandatory refresh* the depth snapshot contains order book information that has not been sent in a depth incremental.

5.5 Update the order book

Every update in the form of a *depth incremental* or *depth snapshot* message contains the price level and the actual price to which the instruction needs to be applied. The receiver application can update information at a particular level with the new information.

Once participants have built the current order book it needs to be continuously updated.

5.5.1 Update the order book with EOBI

Once the initial order book is built, the order book must be maintained using the corresponding order book updates received on the EOBI incremental channel. Any incoming incremental messages with a sequence number higher than the *LastMsgSeqNumProcessed* received in the snapshot message should be applied to the order book after the full snapshot message is processed.

Order Add, Order Modify, Order Modify Same Priority, Order Delete, Full and Partial Order Execution messages are used to add, delete or update individual orders in the order book. Each order could be uniquely identified by its *SecurityID* or instrument identifier, its *TrdRegTSTimePriority* or priority time stamp and its *Side*.

5.5.2 Update the order book with EMDI

Order book update for **EMDI:**

- Keep applying all *depth incremental* messages to the current order book.

Note: *Depth snapshot* messages are sent on a different channel to the *depth incremental* messages. Changes to the order book are also sent using the *depth snapshot* messages but the information is also provided with the incremental messages. Snapshot messages don't need to be processed unless the order book needs to be recreated.

5.5.3 Update the order book with MDI

Order book update for **MDI:**

- Keep applying all *depth incremental* as well as *depth snapshot* messages to the current order book.

Every incremental message can carry different message types which could be sent as FIX snapshot or incremental instructions with the "update action" (New, Change, Delete, Delete From, Delete Through, Overlay) which provide clear instructions on what operation the receiver must perform.

Note: The *depth snapshot* messages for MDI are sent on the same channel as the *depth incremental* messages. If the *RefreshIndicator (1187)* is set to *Y=Mandatory refresh*, changes to the order book are processed into the *depth snapshot* messages and not provided as separate *depth incremental* messages.

6 Recovery

Due to the unreliable nature of UDP multicast it is possible that some packets may be delayed, arrive in incorrect order or might be missing. Furthermore the UDP packets may be duplicated at the network level. Receiving applications need to be capable of handling these issues. This chapter describes the scenarios which might occur and provides a guideline on how a receiving application needs to react to those scenarios.

Recovery actions are possible on a packet level by using the respective other service (A or B). In case a packet is lost on both services (A and B) clients can create a new current order book by using snapshot information.

Troubleshooting an issue starts with the recovery method on the packet level, followed by an explanation on how to create a new valid order book in case this recovery method fails.

6.1 Detecting duplicates and gaps by means of the Packet Header

The concept of the Packet Header allows receiving applications to identify **identical** packets between Service A and Service B by a simple memory comparison. Another important function of the Packet Header is to identify **gaps** by means of the *PacketSeqNum* which can be retrieved just by decoding the packet header.

For EOBI the packets will contain an *ApplSeqNum* instead of the *PacketSeqNum*, which can be used in the same way to detect duplicates and missing packets. Note that for EOBI the packets have contiguous sequence numbers per multicast address/port combination.

For EMDI, MDI and RDI packets with the same *SenderCompID* (field length: 1 Byte) have contiguous sequence numbers per multicast address/port combination. This means that the packet header sequence numbers can be used not only to spot duplicates but also to spot missing packets.

The benefit of having a product-level sequence number on top of the packet-level sequence numbers is that it allows participants to see whether what was missed is actually something they are interested in on product level. There are **two recovery options** when a missing packet is detected using the *packet header* sequence number.

Example: A single multicast address carries instrument groups DAX1 and ETC1, but a participant is only interested in DAX1.

I. Pessimistic approach: The receiving application assumes that DAX1 is part of the missing packet: It immediately starts recovery actions just by decoding the *packet header.*

- **Advantage:** Recovery is triggered immediately when observing a missing *PacketSeqNum* without decoding the entire message.
- **Disadvantage:** The recovery might not be necessary, if the instrument group of interest is not part of the message which is inside the lost packet.

II. Optimistic approach: The receiving application assumes that DAX1 is not part of the missing packet: It waits for the next message to see whether there is actually a broadcast missing for DAX1 before triggering recovery actions.

- **Advantage:** This approach allows the participant to recover only instrument groups of interest
- **Disadvantage:** The receiving application needs to wait for the next message. However, the next packet may not contain a message for the instrument group in question.

6.2 How to recover data via the respective other service (A or B)

Feeds are replicated onto two services, "Service A" and "Service B", and carried on different multicast addresses. This feature provides a possibility to recover missed packets, and participants are advised to join both services.

In each of the following tables, the "Time" column is entirely arbitrary and is intended to show only the sequence of events and in some cases the relative delay between dependent events.

The following table explains the design concept for Service A and B. The table contains the field *MsgSeqNum* from the message itself. However, it could also contain the field *PacketSeqNum* from the *Packet Header*.

Time	Service A: MsgSeqNum	Message	Time	Service B: MsgSeqNum	Message
10:30:00	206	New 151@4	10:29:59	206	New 151@4
10:30:05	207	Delete 151@5	10:30:07	207	Delete 151@5
	Lost		10:30:12	208	New 151@5
10:30:10	209	New 152@4	10:30:13	209	New 152@4

As the example shows, the same information is delivered on Service A and B. While *MsgSeqNum*=208 is missing on Service A, it is provided in Service B.

Ideally a receiving application processes packets from both Service A and B simultaneously and would take into account the message that arrives first and discards the second (identical) message.

In the unlikely event that the message has neither been received via Service A nor Service B, the receiver is required to initiate a loss of data scenario:

- The order book needs to be recreated by using the **depth snapshot** messages in conjunction with the **depth incremental** messages. This procedure is similar to the Start Up procedure. Please see section 5.4 Build the initial order book.
- The maximum expected recovery interval for a particular feed can be obtained from the product snapshot message of the RDI snapshot feed (field: *MDRecoveryTimeInterval (2565)*).

6.3 Packets not delayed (no recovery)

Let's start with a simple case:

Time	MsgSeqNum	Message
10:30:00	132	New 151@4
10:30:04	133	Delete 151@5
10:30:39	134	New 152@4

In this example, messages arrive in the correct order. The message was not delayed between Xetra and the receiving application. There is no special requirement on the application; the message can be processed in the same order as they arrive.

6.4 Delayed packets

Multicast does not guarantee that the order in which packets are received is the same as the order in which they are sent. For instance, the Market Data Interface sends incremental messages in ascending *MsgSeqNum* order, but they might arrive in reversed order at the receiving application.

Consider the following example:

Time	MsgSeqNum	Message
10:30:00	206	New 151@4
10:30:04	208	Delete 151@5
10:30:10	207	New 152@4

In this example, message 207 is delayed within the network, allowing message 208 to arrive first. A correct communications layer responds as follows:

1. Releases message 206 to the application immediately on arrival.
2. On arrival of 208, recognizes that 207 is missing.
3. Starts an appropriate timed operation to trigger the recovery actions if the out-of-sequence message 207 fails to arrive in reasonable time.
4. Assuming that 207 arrives within that reasonable time, releases 207 and then 208 to the application in that order and cancels the times recovery action.

6.5 Missing packets

All lost packets start life as “delayed” packets, as illustrated in the preceding case. The communications layer of the receiving application is responsible for deciding when to declare a network packet as lost. In the following example it is assumed that *MsgSeqNum*=207 from the example above does not arrive within the allowed time. Therefore it is considered as lost:

Time	MsgSeqNum	Message
10:30:00	206	New 151@4
	lost	
10:30:04	208	Delete 151@5
10:30:10	209	New 152@4

The correct behavior in this instance is:

- Release message 206 immediately on arrival.
- Hold on to 208 because it is out-of-sequence, and initiate timer-based recovery actions.
- Hold on to 209 for the same reason. Timer-based recovery actions are already pending, so do not reset the timer.
  - Even though message 209 is a “New” operation, it may be unsafe to apply 208 and 209 because we do not know what 207 contained.
- If the missing message (207) fails to arrive within the allowed time:
  - Initiate recovery from the respective other service (A or B) for message (207). If this works then release (207) and then all messages with higher *MsgSeqNum’s*.

- In case the recovery from the respective other service (A or B) fails: initiate recovery via snapshots. After applying the new snapshot apply all incremental messages with *MsgSeqNum* higher than *LastMsgSeqNumProcessed* of the snapshot message.

6.5.1 Recovery (EOBI)

Participants can utilize the EOBI snapshot channel to obtain the corresponding lost information, i.e., rebuild the initial order book, determine trade statistics and instrument states.

In the event of a packet loss on both services of an EOBI channel, recovery on the participant side can be achieved by recovering the order book information via the EOBI snapshot channel. The EOBI snapshot channel is synchronized with the EOBI incremental channel through the use of message sequence numbering. Participants should subscribe to the Order Book Snapshot channel while buffering incoming messages from the EOBI incremental channel. Any incoming message from the EOBI incremental channel with a *MsgSeqNum* higher than the value of the *LastMsgSeqNumProcessed* field received in the Product Summary snapshot message should be applied to the order books after the full product snapshot is processed.

6.5.2 Exchange Failover – Restart of EOBI

A fail-over of an EOBI incremental channel will be recognizable by the following features:

- The *ApplSeqNum* in the Packet Header is reset to 1.
- The *MsgSeqNum* in the Message Header will continue to be incremented, ideally without a gap.

When a participant receives packets on a specific multicast address with an unexpected (lesser or equal) sequence number (either on service A or service B), it is advised, that the participant subscribes to the EOBI snapshot channel again to rebuild the order book. Note that, because of the unreliable nature of the UDP protocol, packets may arrive out of sequence.

For a full restart of the EOBI service,

- The *ApplSeqNum* in the Packet Header is reset to 1.
- The *MsgSeqNum* in the Message Header is reset to 1.

In case of a full restart of the EOBI service, participants must wait for the first message after the restart to be certain that a restart was executed. It is expected, that a full restart of EOBI feed will take much longer than the configured heartbeat interval.

The field *ApplSeqResetIndicator* is always set in the Packet Header of the first few incremental messages after a (re-)start.

6.5.3 Recovery (EMDI)

Depth snapshot and *depth incremental* messages are distributed via separate channels for the EMDI. For instance, *depth incremental* messages could be sent on multicast address A_2^I port x and the snapshot message on multicast address A_2^S with port y (see chapter 3 Differences between the interfaces).

While an incremental message is sent when there was a change of the order book (event-driven), snapshot messages are sent at fixed intervals regardless of whether the order book has changed since the last snapshot (time-driven).

Each message sequence number (field: MsgSeqNum) on the market data incremental feed is unique and contiguous by product across messages. Therefore the sequence number can be used to detect losses. If any gap of the arriving sequence numbers is detected and this gap cannot be filled by using the respective other service (A or B) the receiving application should initiate a snapshot recovery.

The following example shows missing depth incremental messages (MsgSeqNum's 208-209) and depth snapshots (with LastMsgSeqNumProcessed) which relate to the missing message. MsgSeqNum's for the depth snapshot do not exist, which is indicated with "N/A" in the table.

MesgSeqNum	Instrument Group	LastMsgSeqNum Processed	Message Type	Channel
205	A		Quote request	A_1^I
206	A		depth incremental	A_1^I
207	A		depth incremental	A_1^I
Lost	A		depth incremental	A_1^I
Lost	A		depth incremental	A_1^I
210	A		depth incremental	A_1^I
1000	B		depth incremental	A_2^I
N/A	A	209	depth snapshot	A_1^S
211	A		depth incremental	A_1^I
N/A	B	1000	depth snapshot	A_2^S
1001	B		depth incremental	A_2^I

The appropriate recovery action for missing depth incrementals is the same as the logic described in section 5.4.2 Build the initial order book with EMDI.

Note: *Depth snapshot* messages are not sequenced, but they are still theoretically subject to out-of-order packet delivery. Applications must consider this in determining that their snapshot cycle is complete. The packet sequence number in the *packet header* can be used to detect out-of-order delivery.

6.5.4 Recovery (MDI)

Snapshot and incremental messages are sent on the same channel and carry a contiguous sequence number (field: *MsgSeqNum*) per product. The snapshot always carries the latest information and might carry new information, not already sent with an incremental message. The following table shows an example for the distribution of incremental and snapshot messages for two products:

MesgSeqNum	Instrument Group	Message Type	Channel
5	A	Quote request	$A_1^{S,I}$
6	A	depth Incremental	$A_1^{S,I}$
Lost	A	depth Incremental	$A_1^{S,I}$
25	B	depth Incremental	$A_2^{S,I}$
8	A	depth Incremental	$A_1^{S,I}$
9	A	depth snapshot	$A_1^{S,I}$
10	A	depth snapshot	$A_1^{S,I}$
11	A	depth Incremental	$A_1^{S,I}$
26	B	depth snapshot	$A_2^{S,I}$

MesgSeqNum	Instrument Group	Message Type	Channel
27	B	depth Incremental	$A_2^{S,I}$

If the market data incremental message for instrument group A with *MsgSeqNum*=7 is lost, a consistent order book can be rebuilt from the next snapshot message for instrument group A, in this case arriving with *MsgSeqNum*=9.

All depth incremental messages for instrument group A with a lower sequence number than the next market data snapshot message for instrument group A must be discarded, e.g. *MsgSeqNum*=8 (incremental) must be discarded as its effect is included in *MsgSeqNum*=9 (snapshot).

Since multicast doesn't guarantee the correct sequence of the incoming message, it is recommended to buffer all incoming incrementals while waiting for the next snapshot message. The buffered incrementals for instrument group A with *MsgSeqNum* ≥ 11 can be applied to the latest snapshot with *MsgSeqNum* = 10.

Note: *LastMsgSeqNumProcessed* is not necessary for recovery purpose in MDI.

6.5.5 Exchange Failover – Restart of EMDI/MDI/RDI

A new *SenderCompID* for a specific *MarketSegmentID* (EMDI/MDI) or a specific *MarketID* (RDI), available in the packet header and in each data message for incrementals and snapshots, indicates a fail-over of the market data feed.

Optimistic applications can try to continue with incrementals from the new *SenderCompID* as long as *MsgSeqNum's* are contiguous to the previous *SenderCompID*. However, e.g. if *MsgSeqNum's* from incremental channels are not contiguous to the previous *SenderCompID*, it is always safe to assume a complete restart of market data services on the exchange side and to initiate a recovery from snapshot channels immediately.

7 Various timestamps in Xetra and how to use them

T7 timestamps are taken at gateways, matching engines and market data servers, both in production and simulation. They are also provided through messages sent on EOBI, EMDI and MDI feeds. These can be used to analyze one way transport times. Timestamps are in UTC, and represented as nanoseconds past the UNIX epoch (00:00:00 UTC on 1 January 1970).

An incoming transaction is timestamped at:

Matching Engine:

- order book maintenance and execution,
- creation of direct responses as well as execution messages all for passive orders and quotes,
- creation of listener broadcast for standard orders.

Market Data (EOBI, EMDI and MDI):

- *SendingTime (TransactTime in EOBI)* for order book delta and snapshot messages,
- additionally timestamps from Matching Engine such as *MDEntryTime/TransactTime (TrdRegTSTimePriority/TransactTime in EOBI), RequestTime* and *AggresorTime,* etc. are provided on market data messages.

The following table lists the mapping of T7 timestamps:

Timestamp	Semantic	FIX fields	Description		
t_3	Gateway request in	RequestTime (5979)	Provides the time the application has read an inbound message on a gateway from the TCP socket.		
t_3'	Gateway request out	RequestOut (7764)	Provides the time the application has sent an outbound message from a gateway to the matching engine.		
t_4'	Gateway response in	ResponseIn (7765)	Provides the time the application has received an inbound message on a gateway from a matching engine.		
t_4	Gateway response out	Sending Time (52)	Provides the time the application has written an outbound message on a gateway to the TCP socket.		
t_7	Priority timestamp, Creation timestamp, Transaction timestamp, etc	TrdRegTSTimePriority (21008), ExecID (17), TransactTime (60), etc.	Taken when a transaction is functionally processed and is unique per product. It could be seen in either of the FIX fields depending on if it corresponds to fresh order or quote transaction, or as transaction timestamp for others.		
t_5	Matching engine in	TrdRegTSTimeIn (21002)	Provides the time the application has received an inbound message on a matching engine.		
t_6	Matching engine out	TrdRegTSTimeOut (21003)	Provides the time the application has sent an outbound message from a matching engine.		
t_8	EMDI out	SendingTime (byte vector)	Provides the sending time when EMDI has put the datagram on the wire.		
t_9			EOBI out	TransactTime (60)	Provides the sending time

8 Important topics with use cases and examples

The following section "Use Cases" describes situations which require special attention. Various examples are provided.

8.1 Reference data messages

Reference data provides technical and functional information about all product and instruments available in Xetra. Reference data messages are sent within different feeds:

- **Snapshot feed of RDI** provides a snapshot of all products and instruments, and is sent out on a regular basis throughout the day.

The following messages are sent via the snapshot feed of RDI:

- **Product and Instrument snapshot** for products and instruments available at start of day.
- **Market data report** indicates the start of reference data (MDReportEvent=3).
- **Market data report** indicates the end of reference data (MDReportEvent=4).

8.2 General reference data rules

8.2.1 Reference data snapshot feed

A complete snapshot cycle starts with the **market data report message** (MDReportEvent =3) followed by an **instrument snapshot** for each instrument.

A **market data report message** (MDReportEvent =4) indicates the end of a snapshot cycle.

8.2.2 Counters as part of the Market Data Report message

The message sequence numbers of the market data report messages preceding each snapshot cycle represent counters for the number of snapshots, incrementals and overall number of messages within the current cycle. The *market data report* message, type: **"StartOfReferenceData"**, contains the following sequence number fields:

- **MDReportCount (2536):** Number of reference data messages in the snapshot cycle, which is available at the start of day and remains constant throughout the operational hours of reference data service for the current business day.
- **LastMsgSeqNumProcessed (369):** This is the *MSGSeqNum* value of the last reference data message in the snapshot cycle.
- **TotNoInstrumentReports (2538):** Contains the number of instrument level messages sent in the snapshot cycle.

The market data report message, type: **"EndOfReferenceData"** marks the end of reference data messages and doesn't contain any counters.

8.3 Order book rules and mechanics

EOBI disseminates order book updates without any depth restrictions.

The Market Data Interfaces EMDI and MDI provide order book updates from level 1 to the maximum level. The maximum level is provided in the reference data, field *MarketDepth (264).* The order book can be constructed by the depth incremental messages or by the depth snapshot message.

All on-exchange trades and order book updates are reported via the same *depth incremental* messages. However, trades are always sent out prior to order book updates. The following design principle applies to order book updates:

- Each EMDI and EOBI packet relates only to a single product. In other words, although each packet may contain multiple messages, those messages will always relate to the same product. This does not apply to MDI where a single packet may relate to multiple products.
- EOBI reports every single order book update.
- In EMDI/MDI Orders are aggregated per price level and are not distributed individually.
- Changes to the book that result from one atomic action in the matching engine are disseminated in one *depth incremental* message for EMDI.
- Price levels are provided explicitly ((field: *MDPriceLevel* (1023)) and do not need to be derived through the price itself.
- During the product states “Start-Of-Day” and “Pre-Trading” or when no price levels exist, an empty book (MDEntryType=J) is disseminated for the depth snapshot message (not for incremental). In “Pre-Trading”, statistical information is sent in addition to an empty book.
- There can be multiple updates in one message. The bid-side is updated first then the ask side.
- Order book update instructions are sent for each order book side without a specific order of update actions but ordered by price level instead.
  - from best price (price level 1)
  - down to the worst price (max. price level configured per product).
  - if the resulting book depth, after each applied individual order book update instruction, is larger than the specified maximum product depth only the specified maximum product depth must be saved.
- For auctions, the best bid/ask prices are disseminated at price level 1 without depth information. Receiving applications need to delete a pre existing quantity when an absent value is received during a transition into an auction.
- A state transition to Freeze is sent as an instrument state change message and does not require any implicit action.
- Only the snapshot and incremental messages of MDI carry a common and contiguous sequence number per product. The incremental message of EMDI contains a contiguous sequence number per product across all messages, while the snapshot message provides the last sequence number (*LastMsgSeqNumProcessed*) sent in the incremental message.

Note: The order book is only valid after the entire incremental message has been fully processed.

8.3.1 New price level

When a new price level is created in the order book, a *depth incremental* message is sent with field *MDUpdateAction (279)* = 0 (“New”). This indicates that:

- The new price level is to be inserted at the specific price level.

- All existing rows in the order book at the specified and higher levels are to be incremented accordingly.
- Price levels exceeding the maximum specified depth in the EMDI/MDI must not be kept in memory.

Note: The field *MDPriceLevel (1023)* is used to identify which level is being inserted.

8.3.2 Change of a price level

A *depth incremental* message with *MDUpdateAction* = 1 ("Change") indicates:

- A change at a given price level.
- All fields on the specified side at the price level should be updated.

Note: *MDUpdateAction*="change" is sent only when the price does not change. An *MDUpdateAction* (279) "change" contains a price which can be used as a consistency check. However, it never contains a price that is different than the existing one on the current price level.

8.3.3 Deletion of a price level

A *depth incremental* message with *MDUpdateAction (279)* = 2 ("Delete") is used

- to delete a specified price level.

Note: All price levels greater than the deleted one should be decremented. The price of the price level to be deleted is also sent within the message and can be used as consistency check.

8.3.4 Deletion of multiple price levels from a given price level onwards

A *depth incremental* message with *MDUpdateAction (279)* = 4 ("Delete From") is used to

- Delete all price levels ≥ specified price level.

Note: All price levels from the specified one and up to the maximum need to be deleted.

8.3.5 Deletion of multiple price levels up to a given price level

A *depth incremental* message with *MDUpdateAction (279)* = 3 ("Delete Through") is used to

- Delete all price levels from 1 to the specified price level.

Note: All higher than the specified price levels are shifted down to fill the gap of the deleted price levels.

8.3.6 Overlay

A *depth incremental* message with *MDUpdateAction (279)* = 5 ("Overlay") is used to

- Change the price of a specified price level. Other parameters, e.g. quantity might also change.

Note: In contrast to the *MDUpdateAction*="Change" this instruction contains a price change.

8.4 Trade Volume Reporting (EMDI)

All on-exchange trades executed on T7 are reported via *depth incremental* messages. The *depth snapshot* messages contain statistical information about trades only. Trades can be identified in the incremental messages when *MDEntryType* is set to 2 (Trade).

EMDI only disseminates information about on-exchange trades.

When an order executes against the book at multiple price levels, this is reflected by a matching event with multiple match steps. Each Match Step has the trades at one price level and is represented by a unique *MDEntryID (278)* and published in the market data.

The field *MDEntryID (278)* is a unique id on product level for each business day. A match can result in more than one trade volume record with the same *MDEntryID (278).*

Every match step occurring in the exchange has an identifier in ETI that is provided in the field *FillMatchID (28708)* in the Execution Report (8), *QuoteEventMatchID (8714)* in the Quote Execution Report (U8) and *TrdMatchID (880)* in the Trade Capture Report (AE). This identifier allows participants to link trade capture reports and the corresponding execution report of the ETI with the market data incremental feed of the EMDI.

In case of a market data feed restart, the *MDEntryID (278)* is set to NULL in each *MDIncGrp* entry of the first *Depth Incremental* message after the *MsgSeqNum (34)* is reset to value 1. Member applications that look at the *TradeCondition (277)* value "Exchange Last" (=U) should also check whether an *MDEntryID (278)* is set before they use the *MDEntrySize (271)* to derive a new trade volume from the previous one. If *MDEntryID (278)* is absent then the trade provides the last valid trade prior to the restart and not a new trade after the restart.

The *AggressorTime (2445)* and *RequestTime (5979)* timestamps are provided for the incoming orders when they lead to an immediate execution. In some cases they are not published, for example for trades resulting from an auction uncrossing. It is also possible that the *AggressorSide (2446)* appears without *AggressorTime (2445)* information.

The *RestingCxlQty (28869)* is provided when a resting order is deleted due to a self match prevention (SMP) event. There may be a SMP event in the context of a trade. But there could also occur a pure SMP event. In this case, there is no *MDEntryID (278)* and the *MDEntrySize (271)* is zero.

The following 2 cases illustrate the *MDEntryID (278)* and how Trade Volume Reporting works.

8.4.1 Use case 1: Direct match of instruments

An incoming order is matched against two orders of the opposite side of the order book in different price levels.

Incoming buy order, 150@2885, DE0005140008

Existing Order book:

Bid	Ask
	50@2884 DE0005140008
	100@2885 DE0005140008

Trade Volume Reporting: Two trades are reported because two different price levels are involved in the matching process: First 50@2884 gets reported due to a higher matching priority of this price level; afterwards 100@2885.

Instrument	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell
DE0005140008	1	NEW	50@2884	U,R,AX,AY	Buy	1	1
DE0005140008	2	NEW	100@2885	U,AX	Buy	1	1

With: U="Exchange last", R="Opening price", AX="High price", AY="Low price", AW="Last auction price".

8.4.2 Use case 2: Opening Auction

After the uncrossing of the order book in an instrument at the end of an auction call phase, five orders on the buy side and 3 orders on the sell side of the order book have been matched. The *TradeCondition (277)* is set to AW for Auctions. The field *TrdType (828)* specifies the type of the auction. For trades outside the auction, *TrdType (828)* is not set

Existing Order Book during Auction:

Bid	Ask
30@24.39 DE0005140008	60@24.39 DE0005140008
25@24.39 DE0005140008	57@24.39 DE0005140008
20@24.39 DE0005140008	18@24.39 DE0005140008
55@24.39 DE0005140008	
5@24.39 DE0005140008	

Trade Volume Reporting: All orders are matching on the same price level. Therefore they are reported only once but different with *NumberOfBuyOrders (2449)*/*NumberOfSellOrders (2450)*. The *AggressorSide (2446)* is left empty because during an auction, orders are not considered to be aggressive.

The following *depth incremental* message is sent:

Instrument	MDEntryID	MDUpdateAction	size@prc	TradeCond.	TrdType	AggrSide	#Buy	#Sell
DE0005140008	1	NEW	135@24.39	U,R,AX,AY,AW	OPENING		5	3

The following *depth snapshots* belong to the *depth incremental* above:

Instrument	MDUpdateAction	size@prc	TradeCond.	TrdType
DE0005140008	NEW	135@24.39	U,R,AX,AY	
DE0005140008	NEW	135@24.39	AW	OPENING

In the snapshot, the last auction prices are published in dedicated entries for each auction type separately. Each additional trade from another auction type adds an entry in the snapshot up to a maximal number of four entries, one for each type of auction. If an auction trade gets reversed the respective snapshot entry for the auction trade does not get deleted.

8.5 Trade Volume Reporting (MDI)

MDI only provides statistical data (daily high/low price as well as total trade volume) for trades as well as the last traded price and quantity. Other information such as *NumberOfBuyOrders (2449)* and *NumberOfSellOrders (2450)* are not provided.

For each instrument participating in a trade, MDI reports the total traded volume.

9 Appendix

9.1 Xetra Enhanced Broadcast Solution delta

The following table provides an overview of the main areas where the un-netted market data (EOBI and EMDI) differ from the Xetra Enhanced Broadcast Solution. Please note that the most functional concepts of EMDI are similar to those of EOBI, there is no separate delta provided for EOBI.

Category	Xetra Area	Enhanced Broadcast Solution <-> EMDI
Genral Concepts	Packet Header	**Old:** Contains a sequence number field which can be used to detect duplicates between Service A and Service B. **New**: Contains a sequence number field which can be used to detect duplicates between Service A and Service B. Additionally, the sequence number field can be used for gap detection.
	Sequence numbers within the messages	**Old**: Sequence numbers are incremented per instrument and per source ID. **New**: Sequence numbers are incremented on a product level.
	Gap detection	**Old**: The sequence number field within the message can be used for gap detection. **New**: Sequence numbers of the Packet Header or message can be used for gap detection.
	Buffering of update instructions beyond max. order book depth	**Old**: Receiving applications need to allow temporary growth beyond the configured market depth until all actions of a message have been processed. **New**: It is not allowed to buffer update instructions beyond max. order book depth.
	Host failover/failback	**Old**: Source Id (field: srcId) can fail back to the old number. **New**: SenderCompID does never fail back. Instead a new (higher) number is provided.
	Layout for orders and trades	**Old**: Different message structure for orders (Delta message) and trades (Trade Info message). **New**: Trade Price information and orders are disseminated via the same messages (incrementals).
	Mapping between orders and trades	**Old**: Trade event sequence number is provided to link trade information to order book. **New**: Trade event sequence number is not provided. (Order book and trade information will be published in the same message, therefore it is not required to link trade with order book)
Syntax	FAST-decoding	**Old**: FAST1.1 supported. **New**: FAST1.2 supported.
	Types of update instructions	**Old**: New, Change, Delete, Delete From. Delete Through and Overlay not used. **New**: New, Change, Delete, Delete Through, Delete From and Overlay.
	Timestamps	**Old**: Timestamps are measured in CET. **New**: Timestamps are measured in UTC.

Category	Xetra Area	Enhanced Broadcast Solution <-> EMDI
	Granularity of timestamps	**Old**: High Resolution Timestamp (hiResTimestamp) are measured in microseconds from midnight. Low resolution timestamps are measured in nanoseconds from midnight. **New**: All timestamps are provided in high resolution (nanoseconds since 01/01/1970).

9.2 Xetra Market Data Interface delta

The following table provides an overview of the main areas where the MDI (T7) differs from the Xetra MDI (old) version.

Category	Xetra Area	Xetra MDI <-> MDI (T7)
Genral Concepts	Packet Header	**Old:** Contains a sequence number field which can be used to detect duplicates between Service A and Service B. **New**: Contains a sequence number field which can be used to detect duplicates between Service A and Service B. Additionally, the sequence number field can be used for gap detection.
	Gap detection	**Old**: The sequence number field within the message can be used for gap detection. **New**: Sequence numbers of the Packet Header or message can be used for gap detection.
	Timestamps	**Old**: Timestamps are measured in CET. **New**: Timestamps are measured in UTC.
	Granularity of timestamps	**Old**: High Resolution Timestamp (LastUpdateTime, timestamp) is measured in microseconds from midnight. Low resolution timestamps (MDEntryTime) are measured in milliseconds from midnight. **New**: All timestamps are provided in high resolution (nanoseconds since 01/01/1970).
	Types of update instructions	**Old**: New, Change, Delete, Delete From. Delete Through and Overlay not used. **New**: New, Change, Delete, Delete Through, Delete From and Overlay

9.3 Reference Data delta

The following table provides an overview of the main areas where RDI differs from the reference data provided by Xetra Enhanced Broadcast Solution.

Category	Xetra Area	Reference Data Xetra <-> RDI
Genral Concepts	Provision of reference data	**Old**: The Reference data is delivered as part of the Xetra Enhanced Broadcast Solution, via the Xetra Website, Xetra Member Section and Common Report Engine (CRE). **New**: Participants have the choice between the Reference Data Interface (RDI) and the Reference Data Files delivered via the Common Report Engine, Xetra Website and Xetra Member Section.

Category	Xetra Area	Reference Data Xetra <-> RDI
	Product state changes	**Old**: Dynamic reference data provides information about product state changes. **New**: No product/instrument state via Reference Data Incremental Feed.

10 Change log

The document contains the following changes compared to the previous versions.

No	Chapter, page	Date	Change
1	Genaral	30.09.2016	Initial version