

Eurex Exchange's T7

Eurex Market and Reference Data Interfaces

Manual

Version	4.0.4
Date	11. November 2016

©Eurex 2016

Deutsche Börse AG (DBAG), Clearstream Banking AG (Clearstream), Eurex Frankfurt AG, Eurex Clearing AG (Eurex Clearing) as well as Eurex Bonds GmbH (Eurex Bonds) and Eurex Repo GmbH (Eurex Repo) are corporate entities and are registered under German law. Eurex Zürich AG is a corporate entity and is registered under Swiss law. Clearstream Banking S.A. is a corporate entity and is registered under Luxembourg law. U.S. Exchange Holdings, Inc. is a corporate entity and is registered under U.S. American law. Deutsche Boerse Asia Holding Pte. Ltd., Eurex Clearing Asia Pte. Ltd. and Eurex Exchange Asia Pte. Ltd are corporate entities and are registered under Singapore law. Eurex Frankfurt AG (Eurex) is the administrating and operating institution of Eurex Deutschland. Eurex Deutschland and Eurex Zürich AG are in the following referred to as the "Eurex Exchanges".

All intellectual property, proprietary and other rights and interests in this publication and the subject matter hereof (other than certain trademarks and service marks listed below) are owned by DBAG and its affiliates and subsidiaries including, without limitation, all patent, registered design, copyright, trademark and service mark rights. While reasonable care has been taken in the preparation of this publication to provide details that are accurate and not misleading at the time of publication DBAG, Clearstream, Eurex, Eurex Clearing, Eurex Bonds, Eurex Repo as well as the Eurex Exchanges and their respective servants and agents (a) do not make any representations or warranties regarding the information contained herein, whether express or implied, including without limitation any implied warranty of merchantability or fitness for a particular purpose or any warranty with respect to the accuracy, correctness, quality, completeness or timeliness of such information, and (b) shall not be responsible or liable for any third party's use of any information contained herein under any circumstances, including, without limitation, in connection with actual trading or otherwise or for any errors or omissions contained in this publication.

This publication is published for information purposes only and shall not constitute investment advice respectively does not constitute an offer, solicitation or recommendation to acquire or dispose of any investment or to engage in any other transaction. This publication is not intended for solicitation purposes but only for use as general information. All descriptions, examples and calculations contained in this publication are for illustrative purposes only.

Eurex and Eurex Clearing offer services directly to members of the Eurex exchanges respectively to clearing members of Eurex Clearing. Those who desire to trade any products available on the Eurex market or who desire to offer and sell any such products to others or who desire to possess a clearing license of Eurex Clearing in order to participate in the clearing process provided by Eurex Clearing, should consider legal and regulatory requirements of those jurisdictions relevant to them, as well as the risks associated with such products, before doing so.
Eurex derivatives are currently not available for offer, sale or trading in the United States or by United States persons (other than EURO STOXX 50® Index Futures, EURO STOXX 50® ex Financials Index Futures, EURO STOXX® Select Dividend 30 Index Futures, EURO STOXX® Index Futures, EURO STOXX® Large/Mid/Small Index Futures, STOXX® Europe 50 Index Futures, STOXX® Europe 600 Index Futures, STOXX® Europe 600 Banks/Industrial Goods & Services/Insurance/Media/Travel & Leisure/Utilities Futures, STOXX® Europe Large/Mid/Small 200 Index Futures, Dow Jones Global Titans 50 IndexSM Futures (EUR & USD), DAX®/Mini-DAX®/MDAX®/TecDAX® Futures, SMIM® Futures, SLI Swiss Leader Index® Futures, MSCI World (FMWO, FMWP, FMWN)/Europe (FMEU, FMEP)/ Europe Value/Europe Growth/Emerging Markets (FMEM, FMEF, FMEN)/Emerging Markets Latin America/Emerging Markets EMEA/Emerging Markets Asia/China Free/India/Japan/Malaysia/South Africa/Thailand/AC Asia Pacific ex Japan Index Futures, TA-25 Index Futures, Daily Futures on TAIEX Futures, VSTOXX® Futures, Gold and Silver Futures as well as Eurex FX, property and interest rate derivatives).

Trademarks and Service Marks
Buxl®, DAX®, DivDAX®, eb.rexx®, Eurex®, Eurex Bonds®, Eurex Repo®, Eurex Strategy WizardSM, Euro GC Pooling®, FDAX®, FWB®, GC Pooling®„GCPI®, MDAX®, ODAX®, SDAX®, TecDAX®, USD GC Pooling®, VDAX®, VDAX-NEW® and Xetra® are registered trademarks of DBAG.

All MSCI indexes are service marks and the exclusive property of MSCI Barra.

ATX®, ATX® five, CECE® and RDX® are registered trademarks of Vienna Stock Exchange AG.
IPD® UK Annual All Property Index is a registered trademark of Investment Property Databank Ltd. IPD and has been licensed for the use by Eurex for derivatives.
SLI®, SMI® and SMIM® are registered trademarks of SIX Swiss Exchange AG.
The STOXX® indexes, the data included therein and the trademarks used in the index names are the intellectual property of STOXX Limited and/or its licensors Eurex derivatives based on the STOXX® indexes are in no way sponsored, endorsed, sold or promoted by STOXX and its licensors and neither STOXX nor its licensors shall have any liability with respect thereto.
Dow Jones is a service mark of Dow Jones & Company, Inc. All derivatives based on these indexes are not sponsored, endorsed, sold or promoted by Dow Jones & Company, Inc. Dow Jones & Company, Inc. does not make any representation regarding the advisability of trading or of investing in such products.
Bloomberg Commodity IndexSM and any related sub-indexes are service marks of Bloomberg L.P.
All references to London Gold and Silver Fixing prices are used with the permission of The London Gold Market Fixing Limited as well as The London Silver Market Fixing Limited, which for the avoidance of doubt has no involvement with and accepts no responsibility whatsoever for the underlying product to which the Fixing prices may be referenced. PCS® and Property Claim Services® are registered trademarks of ISO Services, Inc.
Korea Exchange, KRX, KOSPI and KOSPI 200 are registered trademarks of Korea Exchange Inc.
Taiwan Futures Exchange and TAIFEX are registered trademarks of Taiwan Futures Exchange Corporation. Taiwan Stock Exchange, TWSE and TAIEX are the registered trademarks of Taiwan Stock Exchange Corporation.
BSE and SENSEX are trademarks/service marks of Bombay Stock Exchange (BSE) and all rights accruing from the same, statutory or otherwise, wholly vest with BSE.
Any violation of the above would constitute an offence under the laws of India and international treaties governing the same. The names of other companies and third party products may be trademarks or service marks of their respective owners.

Contents

I General Overview 5

1 List of abbreviations 5

2 Introduction 6
2.1 Purpose of this document . . . 7
2.2 Main audience . . . 7
2.3 Data feeds . . . 7
2.3.1 Reference data interface . . . 8
2.3.2 Market data interfaces . . . 8
2.4 Interface version number . . . 9
2.5 Deutsche Börse customer support . . . 9
2.6 Further reading matter for this topic . . . 10
2.7 How to read this document . . . 10

3 Differences between the interfaces 12
3.1 Distribution sequence for Eurex EMDI . . . 13
3.2 Distribution sequence for Eurex MDI / Eurex RDI . . . 13
3.3 Choosing between the Eurex EMDI and the Eurex MDI . . . 14
3.4 Choosing between the Eurex RDI and Eurex RDF . . . 15

4 Overview of the Eurex Public Interfaces 17
4.1 Infrastructure requirements . . . 17
4.2 Trading states . . . 17
4.2.1 Product State Changes . . . 17
4.2.2 Instrument State Changes . . . 18
4.3 Overview of the various message types . . . 19
4.3.1 Eurex RDI . . . 19
4.3.2 Eurex EMDI/MDI . . . 19
4.4 What is not included in these interfaces . . . 20
4.5 FIX over FAST . . . 20
4.6 Freedom of choice . . . 20
4.7 Testing . . . 21
4.8 Hours of operation/availability of messages . . . 21

II How to guide 22

5 FIX/FAST-Implementation 22
5.1 Structure of Messages . . . 22
5.2 FAST terminology . . . 23
5.2.1 FAST reset message . . . 23
5.2.2 Presence Map (PMAP) . . . 23
5.2.3 Template ID (TID) . . . 23
5.2.4 Dictionaries . . . 24
5.2.5 Stop bit encoding . . . 24
5.2.6 FAST operators . . . 24
5.3 Decoding the FAST-message . . . 25
5.4 Transfer decoding . . . 25
5.5 Composing the Actual FIX-Message . . . 25
5.6 New features in FAST version 1.2 . . . 25
5.7 Data types . . . 26
5.8 FAST version 1.1 compatible templates . . . 26

6 Description of a typical trading day 28
6.1 Start of day operation . . . 28
6.2 Receiving reference data via Eurex RDI at start of day . . . 28
6.3 Receiving reference data file (RDF) at start of day . . . 29
6.4 Build the initial order book . . . 30
6.4.1 Build the initial order book with the Eurex EMDI . . . 30
6.4.2 Build the initial order book with the Eurex MDI . . . 31
6.5 Update the order book . . . 31
6.5.1 Update the order book with the Eurex EMDI . . . 31
6.5.2 Update the order book with the Eurex MDI . . . 32

7 Recovery **33**
7.1 Detecting duplicates and gaps by means of the packet header . . . 33
7.2 How to recover data via the respective other service (A or B) . . . 33
7.3 Delayed packets . . . 34
7.4 Missing packets . . . 35
7.4.1 Recovery (Eurex EMDI) . . . 36
7.4.2 Recovery (Eurex MDI) . . . 38

8 Various time stamps in EUREX and how to use them **39**

9 Important topics with use cases and examples **41**
9.1 Reference data messages . . . 41
9.2 General reference data rules . . . 42
9.2.1 General structure of the snapshot cycle . . . 42
9.2.2 Counters as part of the market data report message . . . 43
9.2.3 Use case 1: Reference data at the start of the reference data service . . . 44
9.2.4 Use case 2: Reference data after intraday addition of complex instruments . . . 44
9.2.5 Use case 3: Reference data on the next business day . . . 44
9.2.6 Use case 4: Failover or restart of Eurex RDI . . . 45
9.2.7 Use case 5: Chronological order of messages for complex instrument creation . . . 45
9.2.8 Use case 6: Chronological order of messages for complex instrument deletion . . . 45
9.2.9 Use case 7: Variance Futures Status messages . . . 46
9.2.10 Use case 8: Product pool . . . 46
9.2.11 Use case 9: Flexible instruments . . . 46
9.2.12 Use case 10: Total Return Futures Status messages . . . 46
9.3 General order book rules and mechanics . . . 47
9.3.1 Determination of the price sources . . . 49
9.3.2 Top Of Book . . . 50
9.3.3 New price level . . . 51
9.3.4 Change of a price level . . . 52
9.3.5 Overlay . . . 53
9.3.6 Deletion of a price level . . . 54
9.3.7 Deletion of multiple price levels from a given price level onwards . . . 55
9.3.8 Deletion of multiple price levels up to a given price level . . . 56
9.4 TES Trades . . . 57
9.5 Manual Trade Entry and Trade Reversal (Eurex EMDI) . . . 57
9.5.1 Manual Trade Entry (by Market Supervision) (Eurex EMDI) . . . 57
9.5.2 Trade Reversal (by Market Supervision) (Eurex EMDI) . . . 58
9.6 Trade Volume Reporting (Eurex EMDI) . . . 59
9.6.1 Use case 1: Direct match of simple instruments . . . 60
9.6.2 Use case 2: Direct match of complex instruments . . . 61
9.6.3 Use case 3: Complex versus simple order match . . . 62
9.6.4 Use case 4: Complex versus simple/complex match . . . 63
9.6.5 Use case 5: Opening auction . . . 64
9.6.6 Use case 6: Self match prevention (order is totally canceled) . . . 65
9.6.7 Use case 7: Self match prevention (order is partially canceled) . . . 66
9.7 Trade Volume Reporting (Eurex MDI) . . . 67
9.8 Failure of the market data feed/ matching engine . . . 68
9.8.1 Normal processing . . . 68
9.8.2 Market data feed fail-over (Eurex EMDI) . . . 69
9.8.3 Market data feed fail-over (Eurex MDI) . . . 70
9.8.4 Market data feed restart (Eurex EMDI) . . . 71
9.8.5 Market data feed restart (Eurex MDI) . . . 71
9.8.6 Failure of the matching engine . . . 71
9.9 Trading states for a sample business day . . . 72
9.9.1 Start-Of-Day . . . 72
9.9.2 Pre-Trading . . . 72
9.9.3 Opening Auction . . . 72
9.9.4 Continuous Trading . . . 73
9.9.5 Intraday Expiry . . . 73
9.9.6 Closing Auction . . . 74
9.9.7 Post-Trading . . . 74
9.9.8 End-Of-Day . . . 74

10 Fine tuning client applications 75
10.1 Buffer size . . . 75
10.2 Packet and message processing . . . 75
10.3 Application level . . . 76
10.3.1 Discarding duplicate packets witin the live-live environment . . . 76
10.3.2 Order book processing . . . 76
10.3.3 Optimal processing of desired products (Eurex EMDI) . . . 76

III Reference 78

11 Detailed data feed description and layout 78
11.1 Service messages . . . 78
11.1.1 FAST reset message . . . 78
11.1.2 Packet header (Eurex EMDI) . . . 78
11.1.3 Packet header (Eurex MDI /Eurex RDI) . . . 80
11.1.4 Functional beacon message . . . 80
11.1.5 Technical heartbeat message . . . 81
11.1.6 Market data report message . . . 81
11.2 Reference data messages . . . 82
11.2.1 Product snapshot message . . . 82
11.2.2 Instrument snapshot message . . . 87
11.2.3 Instrument incremental message . . . 92
11.2.4 Variance futures status message . . . 93
11.2.5 Total return futures status message . . . 95
11.3 Market data messages . . . 97
11.3.1 Depth snapshot message . . . 97
11.3.2 Depth incremental message . . . 102
11.3.3 Top Of Book Implied message . . . 105
11.3.4 Product state change message . . . 106
11.3.5 Mass instrument state change message . . . 106
11.3.6 Instrument state change message . . . 109
11.3.7 Quote request message . . . 111
11.3.8 Cross request message . . . 112
11.3.9 Complex instrument update message . . . 113
11.3.10 Flexible instrument update message . . . 115
11.4 Data files . . . 116
11.4.1 Reference data from file (Eurex RDF) . . . 116
11.4.2 File name format of the reference data files . . . 116
11.4.3 Reference data file on the next business day . . . 117
11.4.4 Reference data file after a failover or restart of Eurex RDI . . . 117
11.4.5 What receiving applications need to do . . . 117

12 Multicast addresses 118
12.1 Reference data snapshot feed . . . 118
12.2 Reference data incremental feed . . . 118
12.3 Reference data for Eurex Enhanced Order Book Interface . . . 118

13 FAST templates 119

14 Appendix 120
14.1 Example for a XML FAST template . . . 120
14.2 Example for determination of the price source . . . 121
14.2.1 Fully implied . . . 121
14.2.2 Fully outright on level 1 . . . 122
14.2.3 Partially implied . . . 123
14.2.4 Several fully implied orders at Best Market . . . 124
14.3 FIXML mapping table . . . 125

15 Change log 129

Part I
General Overview

1 List of abbreviations

The table below shows all the abbreviations and definitions used in this document.

Eurex EMDI	Eurex Enhanced Market Data Interface
Eurex MDI	Eurex Market Data Interface
Eurex EOBI	Eurex Enhanced Order Book Interface
Eurex RDI	Eurex Reference Data Interface
Eurex RDF	Eurex Reference Data File
Eurex ETI	Eurex Enhanced Transaction Interface
FAST	FIX Adapted for STreaming (FAST Protocol) (FAST Protocol ^{SM}). 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.
In-band	Incrementals and snapshots are delivered in the same channel.
IPS	Inter-Product Spread.
Match event	Part of the matching event having a unique match price.
Out-of-band	Incrementals and snapshots are delivered on different channels.
Simple instruments	Single leg outright contracts
Complex instruments	Any combination of single leg outright contracts, e.g. Future Time Spreads.
Flexible instruments	User-defined simple instrument for TES trading.
Live-live concept	The concept whereby data is disseminated simultaneously via two separate channels called "Service A" and "Service B".
Off-book trades	Trades performed "Over the Counter".
PMAP	Presence Map
TES	Trade Entry Service
ToB	Top of Book
T7	Eurex Exchange's trading system developed by Deutsche Börse Group

2 Introduction

Eurex offers public market and reference data via three interfaces as part of the Eurex Exchange's T7. All three interfaces distribute information via UDP multicast.

The Eurex Market Data Interfaces are:

- The **Eurex Enhanced Market Data Interface (Eurex EMDI):** This interface provides *un-netted* market data. The updates of the order book are delivered for all order book changes up to a given level; all on-exchange trades are reported individually.
- The **Eurex Market Data Interface (Eurex MDI):** This interface provides *netted* market data. The 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 Eurex EMDI. On-exchange trades are not reported individually, however statistical information (daily high/low price, last trade price and quantity) is provided instead.
- **Eurex Enhanced Order Book Interface (Eurex EOBI):** This interface provides the entire visible order book, by publishing information on each individual order and quote along with state information in *un-netted* manner. All on-exchange trades are not reported individually. See "Eurex Enhanced Order Book Interface - Manual ".

The Eurex EMDI and Eurex MDI provide the following information to the participants:

- Price level aggregated order book depth and trade statistics.[1]
- Product and instrument states.
- Quote requests and cross requests.
- Information on newly created complex and flexible instruments.

Reference data is sent by:

- The **Eurex Reference Data Interface (Eurex RDI)**: This interface provides reference data for products and instruments that are available for trading on the Eurex Exchange's T7. The reference data is delivered on a product and instrument level. Every tradable object is referenced by a unique identifier, for this reason the reference data information is absolutely essential for any trading application.
- The **Eurex Reference Data File (Eurex RDF)**: Reference data is delivered as a start-of-day file and as regularly updated[2] intraday files.

Eurex EMDI, Eurex MDI and Eurex RDI publish market and reference data information following FIX 5.0 SP2 semantics and are FAST 1.2[3] encoded. 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 concept). In the unlikely case that a message is lost on both services, participants can take advantage of the respective snapshot messages and rebuild the order book.

The scope of this manual is Eurex EMDI, Eurex MDI, Eurex RDI and Eurex RDF. For details regarding Eurex Enhanced Order Book Interface (Eurex EOBI), please see "Eurex Enhanced Order Book Interface - Manual ".

Eurex EMDI, Eurex MDI, Eurex RDI and Eurex RDF do not offer any layout-level backward compatibility feature between two releases, and within the lifetime of a release Eurex reserves the right to change the behavior of some fields in the different layouts.

[1] Eurex Off-Book trades replay, settlement prices and open interest information are provided by the Eurex Extended Market Data Service.

[2] Currently with an update interval of 5 minutes.

[3] FAST 1.1 templates are provided as well.

2.1 Purpose of this document

The purpose of this document is to provide guidance for programmers during development of applications that read the *Eurex Market & Reference Data Interfaces*.

It covers a complete reference for the three multicast based public interfaces, describes the general business behaviour and provides concepts for the implementation.

The most recent version is available at:

www.eurexchange.com > Technology > T7 > System documentation > Release 4.0 > Market and Reference Data Interfaces.

2.2 Main audience

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

Prior knowledge of developing for a derivatives market is beneficial but not a prerequisite. Knowledge in a programming language is expected. Programmers who have no experience in a market data interface environment can gain a basic understanding of the feed behaviour by reading Part II (How to guide). This manual does not attempt to cover basic knowledge of programming techniques and software development.

2.3 Data feeds

All interfaces deliver public reference and market data in the form of snapshots and incrementals as can be seen in Figure 1. The two public market data interfaces, the **Eurex EMDI** for a high bandwidth network and the **Eurex MDI** for a low bandwidth network, disseminate information across the Eurex network to the receiving application. The **Eurex RDI** is considered for participants with a high bandwidth network while the **Eurex RDF** should be used if only a low bandwidth network is available.

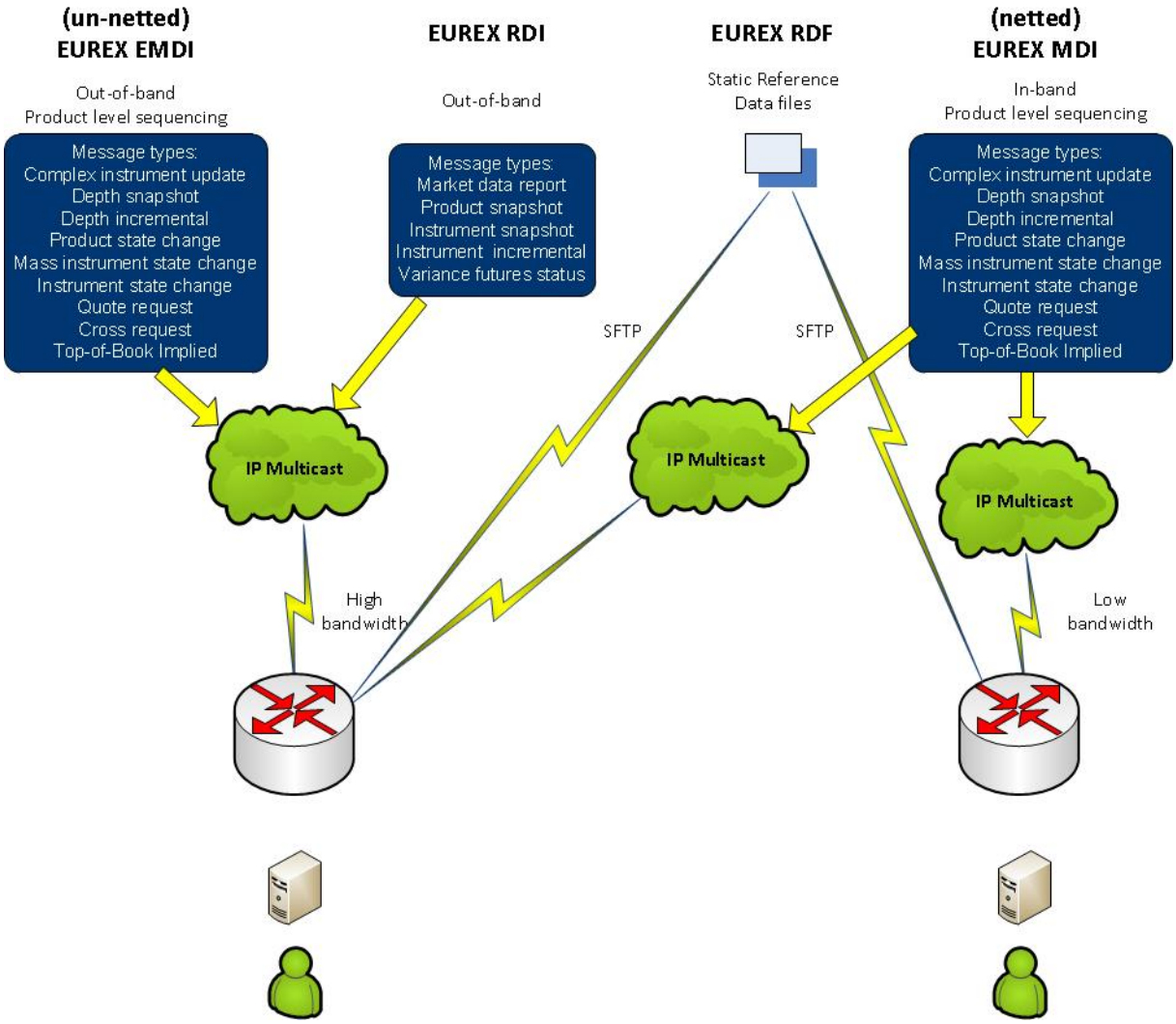


Figure 1: *Reference data and Market data interfaces*

2.3.1 Reference data interface

Public reference data delivered by Eurex RDI contains the technical configuration, e.g. multicast address and port combinations for both market data interfaces for all products and instruments.

There are separate reference data feeds per marketID.

Multicast addresses and port information do not change during trading hours.

The **reference data snapshot feed** contains two message types: Constant number of snapshots and a variable number of incrementals.

The **reference data incremental feed** delivers reference information about intraday created complex instruments.

2.3.2 Market data interfaces

The Eurex EMDI and the Eurex MDI disseminate public market data information in the form of incrementals (event driven) and snapshots (time driven).

The **market data snapshot feed** can be used to recover lost market data or build up the current order book. Receiving applications are not expected to be permanently subscribed to this feed.

The **market data incremental feed** should be subscribed throughout the trading day for receiving order book updates. All incoming messages should be applied to the copy of the order book maintained by the member applications in order to have the latest information.

2.4 Interface version number

Each of the interfaces described in this manual has a version number. The version numbers are listed at the beginning of the FAST XML template files.

This manual relates to the following interface version numbers:

- Eurex EMDI: 4.0.101
- Eurex MDI: 4.0.102
- Eurex RDI: 4.0.226

2.5 Deutsche Börse customer support

Eurex support is available 24hrs on business days and may be contacted as follows:

	Eurex Contact List	
Technical Support (global)	VIP number	cts@deutsche-boerse.com
Technical Support (USA)	VIP number	cts@deutsche-boerse.com
Service Portal	Create problem ticket	member.eurexchange.com > Technical Connection > Problem Tickets
Functional Helpdesk (Equity/Index)	+49-69 2111 1210	eurextrading@eurexchange.com
Functional Helpdesk (Fixed Income)	+49-69 2111 1240	eurextrading@eurexchange.com

Table 1: *Eurex contact list*

2.6 Further reading matter for this topic

This document is designed as an independent learning and reference manual. However, for background information related to network connectivity, FAST/FIX messages or trading related information (functional), further documents are recommended.

The documents listed below provide useful information.

FAST- and FIX-related documents:

- **FAST specification documents**: Explains all FAST rules in detail. FAST 1.2 is the summary of the FAST 1.1 specification plus the extension Proposal.
  www.fixprotocol.org > fastspec
- **FIX specification documents**: FIX-messages and FIX-tags
  www.fixprotocol.org > Specifications
- **FIX-Tags**: Specifies all FIX-Tags
  www.fixprotocol.org > FIXimate3.0

EUREX related documents:
www.eurexchange.com > Technology > T7 > System documentation > Release 4.0

- **Functional & Interface Overview**: This document provides an overview of the Eurex Exchange's T7. It describes the major functional and system changes, and provides a high level description of the interface landscape.
- **Eurex Enhanced Order Book Interface Manual**: This manual describes the concepts and the messages used by this interface.
- **Eurex Enhanced Trading Interface**: It contains a detailed description of the concepts and messages used by this trading interface.
- **Eurex Extended Market Data Service**: This document provides the information about the add-on services for market and reference data, e.g. intraday settlement prices, open interest, etc...
- **Functional Reference**: Provides a detailed description of the business functionality that is available in Eurex Exchange's T7.

2.7 How to read this document

This manual covers the Eurex EMDI and Eurex MDI as well as the Eurex RDI. Differences in functionality between the Eurex EMDI and the Eurex MDI are described in separate sub sections, while being represented by different text colors: **(Eurex EMDI)** and **(Eurex MDI)**.

For example, section 7.4.2, Recovery (Eurex MDI), on page 38 refers to the "netted" Eurex MDI only. Participants who are interested in the "un-netted" Eurex EMDI can ignore this sub chapter. This document consists of three parts:

- **Part I** (General Overview) introduces the interface for beginners.
- **Part II** (How to guide) provides methods and hands-on guidance.
- **Part III** (Reference) is a comprehensive reference with details on various message layouts in table format. A typical table would be the following:

Delivered on: reference data snapshot feed

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	User defined message Value: 0 — Description: Beacon
<GroupName > (optional) group starts				
<SequenceName > sequence starts				
...	...	...	...	...
...	...	...	...	...
<GroupName > sequence ends				
<SequenceName > (optional) group ends				

Table 2: *Typical FIX message description*

Interpreting the fields above:

- **Delivered on**: Specifies the feed which delivers the specific message. A message can be delivered on more than one feed.
- **Tag**: Describes the FIX Tags
- **Field Name**: Describes the FIX-name.
- **Req'd**: Describes whether or not the field is included within the message after FAST-decoding, purely from the FIX-point of view. This does not refer to a FAST-rule, e.g. operators or Presence Map (PMAP) in FAST.
- **Data Type**: FAST data type. This information is also provided in the XML FAST templates.
- **Description**: This column contains an explanation of the FIX-field and it's "valid values" in table format for this particular message.
- **GroupName, SequenceName**: The names correspond with the groups and sequences defined in the FAST XML templates.

Cross references to other chapters within this document and the glossary are provided in blue color.
Example: More information is provided in section 9.1, Reference data messages.

In this document, the terms "incrementals" and "snapshots" are used in various contexts. Within this document "incrementals" and "snapshots" refer either to messages of the market data feed or to messages of the reference data feed. The actual meaning can be inferred from the context.

Note: Important statements made in this manual are highlighted with a shadow box.

3 Differences between the interfaces

A feed is a message flow of logically grouped messages, e.g. the *depth incremental* and *product state change* messages for a particular product are grouped together within the incremental feed of Eurex EMDI. The following diagram illustrates the available feeds for the three multicast based public interfaces:

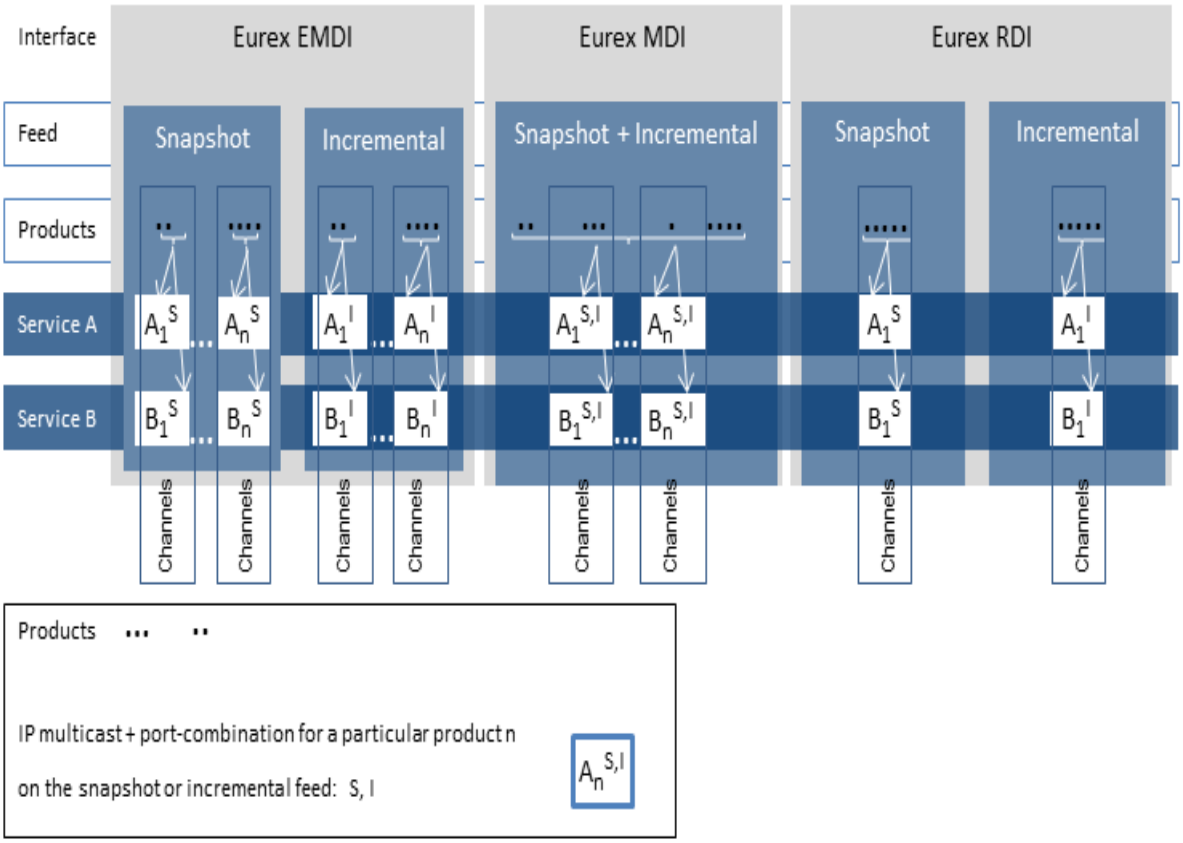


Figure 2: *Overview of the three interfaces*

The Eurex RDI is published on exactly one snapshot channel, indicated by (A_1^S) and one incremental channel (A_1^I). The Eurex EMDI has multiple channels that have either snapshots (A_1^S) to (A_n^S) and multiple incremental channels (A_1^I) to (A_n^I). The Eurex MDI has the snapshots and incrementals combined over multiple channels ($A_1^{S,I}$) to ($A_n^{S,I}$).

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

Several partitions, each with a unique *SenderCompID (49)*, may contribute to the same multicast address as shown in figure 18 on pg. 76. The *SenderCompID (49)* is unique across all partitions. However, it should not be relied upon as under unlikely but possible conditions on the exchange this is not true.

In contrast to the Eurex EMDI, the snapshot and incremental messages for the **Eurex 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 grouped on product basis.

The snapshot and incremental feed for the **Eurex RDI** are delivered via separate channels (out-of band) and need to be synchronized. In contrast to the order book information, the snapshot and incremental feeds are not divided into further channels.

All feeds are sent on two different multicast addresses via different physical connections (Service A and B). Service A and Service B are identical in terms of the information provided, i.e. the packet contents, sequence numbers and sequence in which packets are sent is the same. This is called "live-live" concept.

Product groups are distributed across several partitions on the Eurex backend side. Service A and Service B cannot be published at exactly the same time.

3.1 Distribution sequence for Eurex EMDI

The rule for the **distribution sequence** across partitions is as follows:

Even partitions: Publish on Service A first, then on Service B.
Odd partitions: Publish on Service B first, then on Service A.

The above rule is applied by using the field *PartitionID* (5948). It is available in the *product snapshot* message and in the *packet header* and contains the number of the partition for the product of interest. The *PartitionID* (5948) never changes intraday.

Example: A PartitionID = 8 indicates an even partition and therefore Service A is published before Service B.

Current production data indicate an average time difference of about 10 - 15 µs; the cable length for both Service A and Service B within the co-location is the same, i.e. both services have the same propagation delay.

The multicast addresses for both of these services are disseminated in the product reference information. Due to the inherently unreliable nature of the UDP protocol, data packets may be lost in the transmission network. Therefore members are advised to join both services to reduce the probability of data loss.

3.2 Distribution sequence for Eurex MDI / Eurex RDI

The rule for the **distribution sequence** across partitions is as follows:

Even and odd partitions: Publish on Service A first, then on Service B.

Example: The *PartitionID* (5948) for Eurex MDI and Eurex RDI is not available in the *packet header* but in the *product snapshot* message. However, the *PartitionID* (5948) doesn't need to be considered because Service A is always published first regardless of the partition.

3.3 Choosing between the Eurex EMDI and the Eurex MDI

Both types of interface, un-netted and netted, provide market information via multicast using a price-level aggregated order book (as opposed to, for example, order-by-order feeds) but they have different bandwidth requirements and service levels.

- The **Eurex Enhanced Market Data Interface (un-netted)** disseminates every order book change up to the configured depth and all on-exchange trades without netting. 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 similar products. As the market becomes busier, the number of messages (and therefore bandwidth usage) increases.
- The **Eurex 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.

  This interface aggregates the order book changes over a specified time interval, which is published in the *Product Snapshot* message via field *MarketDepthTimeInterval (2563)*. This interface has less price levels than the Eurex EMDI. Furthermore, only statistical information is provided for on-exchange trades as well as the price and quantity of the last on-exchange trade in the netting interval.

The following table shows the main differences between the Eurex EMDI and the Eurex MDI:

Area	Eurex EMDI	Eurex MDI
In-band/Out-of-band delivery	Incrementals and snapshots are delivered via different channels, i.e. out-of-band delivery. *LastMsgSeqNumProcessed* in the snapshot feed provides a link between incremental and snapshot feed, as it carries the sequence number of the last message sent on the incremental feed. Snapshots are needed only for start-up/recovery.	Incrementals and snapshots are delivered on the same channel, i.e. in-band delivery. Snapshots might contain new information. A flag (*RefreshIndicator*) within the snapshot indicates whether it has to be applied or not. *LastMsgSeqNumProcessed* is not used.
Sequence numbers on message level	Messages on the market data incremental feed have their own sequence number range per product; *MsgSeqNum's* exist on the depth incremental feed only as shown in table 14 on pg. 36.	Messages on the combined market data incrementals + snapshot feed have one sequence number range per product as shown in table 15 on pg. 38.
Trade Volume Reporting	Trade Volume Reporting is provided. Each on-exchange trade is reported individually.	Only statistical information (daily high/low price and total traded quantity) and last trade information is provided.

Area	Eurex EMDI	Eurex MDI
Packet header	A Performance Indicator[4] is provided for incrementals within the Packet Header as shown in figure 19 on pg. 79.	A Performance Indicator does not exist as shown in figure 20 on pg. 80.
Functional beacon message	A *functional beacon* message on a product level including the last valid *MsgSeqNum* is sent if no other message has been sent for a configured time period.	Snapshots act as *functional beacon* message, hence no separate *functional beacon* messages are provided.

Table 3: *Main differences between the Eurex EMDI and the Eurex MDI*

Both interfaces, un-netted and netted, provide different recovery time intervals to offer the participants the opportunity to implement a suitable public market data recovery mechanism. The recovery time interval, *MDRecoveryTimeInterval(2565)*, of a product for each interface is available in *Product Snapshot* message via the Eurex RDI, as well as in file format via the Eurex RDF.

3.4 Choosing between the Eurex RDI and Eurex RDF

Reference data is provided via the **Eurex RDI** and in file form as compressed **Reference Data Files (RDF)** in FIXML-layout, updated approximately every 5 minutes via the Common Report Engine [5] (CRE).

The initial reference data file generated at start-of-day contains the "reference data snapshots" available from the previous day. During the actual trading, multiple incremental files are created as complex and flexible instruments are added. New complex instruments predefined by the exchange are also sent in incremental files before the start of the actual trading.

Please note that the intraday changes to reference data are also published in form of the *complex instrument update* and *flexible instrument update* messages via the market data incremental feed of the Eurex EMDI and the market data feed of the Eurex MDI. During normal operations participants do not need to listen to the incremental feed of the **Eurex RDI**, because the *complex instrument update*[6] and *flexible instrument update* can be received on the market data feed. Furthermore, market data for new complex and flexible instruments is never provided ahead of their reference data on EMDI or MDI but may come ahead of its publication via RDI.

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

The following table shows the main differences between the Eurex RDI (message based) and the Eurex RDF (file based):

[4] Time between arrival of an incoming order/quote transaction on the Eurex matching engine and send time of the corresponding outgoing market data

[5] For more information please see the "Common Report Engine User Guide"

[6] The layout is the same as on the *instrument incremental* message except for the fields *SecurityStatus (965)* and *SenderCompID (49)*

Area	Eurex RDI: Message based	Eurex 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 and apply each Intra-day Reference Data File as they become available. A late starting application can always retrieve the latest picture of the reference data by this method.

Table 4: *Differences between the Eurex RDI and the Eurex RDF*

It is important to note that the Eurex reference data interface, Eurex RDI and Eurex RDF, is intended to be available prior to the daily T7 start up processing. This service will normally be available during non-trading days to support participants with the Eurex public reference data. However, during any kind of infrastructure maintenance this service will not be available on non-trading days.

4 Overview of the Eurex Public Interfaces

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

4.1 Infrastructure requirements

The Eurex market and reference data interfaces disseminate market and reference data over the Eurex 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 Trading states

State changes are disseminated over both the Eurex EMDI and the Eurex MDI market data feeds. Trading state information is not communicated over the Eurex Enhanced Transaction Interface (Eurex ETI) or FIX interface.

The Eurex EMDI and the Eurex MDI market data feeds follow the FIX protocol for the publication of trading state information. The Eurex product and instrument states are displayed by these interfaces as shown in the following tables.

Section 9.9, Trading states for a sample business day illustrates state messages for a typical business day. The hours of operations for the Eurex system is provided in Section 4.8, Hours of operation/availability of messages.

4.2.1 Product State Changes

The product state is published with a *product state change* message (FIX *TradingSessionStatus*, MsgType = h). In this message, the product state can normally be found in the field *TradingSessionSubID (625)*. Only for quiescent product states, the field *TradingSessionID (336)* must be evaluated additionally to determine the actual product state.

product state change message			
Eurex Product State	**FIX TradingSessionID (336)**	**FIX TradingSessionSubID (625)**	**FIX TradeSesStatus (340)**
Start of Day	3 = Morning	7 = Quiescent	3 = Closed
Pre-Trading	3 = Morning	1 = Pre-Trading	2 = Open
Trading	1 = Day	3 = Trading	2 = Open
Closing	1 = Day	4 = Closing	2 = Open
Post-Trading	5 = Evening	5 = Post-Trading	2 = Open
End of Day	5 = Evening	7 = Quiescent	3 = Closed
Post End of Day	6 = After-Hours	7 = Quiescent	3 = Closed
Halt	1 = Day	7 = Quiescent	1 = Halted
Holiday	7 = Holiday	7 = Quiescent	3 = Closed

Table 5: *Product states*

A Halt state is additionally indicated by the FIX field *TradSesStatus (340)* containing the value 1 = Halted. A Fast Market is reported with the same message type using the new FIX field *FastMarketIndicator (2447)* which can take the values 0 = No or 1 = Yes.

The product TES activity status is independent of the on-exchange product state. TES activity status is reported using the new FIX field *TESTradSesStatus (25044)*.

product state change message	
Eurex TES Activity Status	**FIX TESTradSesStatus (25044)**
Off	3 = Closed
On	2 = Open
Ended	5 = Pre-Close
Halted	1 = Halted

Table 6: *TES Activity Status*

4.2.2 Instrument State Changes

The instrument state is published with an *instrument state change message* (FIX *SecurityStatus*, MsgType = f) in case of a single instrument, or with a (FIX *SecurityMassStatus*, MsgType = CO) message in case that all or most of the instruments of a product and of a specific instrument type[7] change their state.

- In the *instrument state change* message (FIX *SecurityStatus*, MsgType = f), the instrument state can be found directly in the field *SecurityTradingStatus* (326).
- In the *mass instrument state change* message (FIX *SecurityMassStatus*, MsgType = CO), the instrument state can be found in the field *SecurityMassTradingStatus* (1679). This message may contain an exception list of instruments that have a different instrument state. The exception list contains the instrument state in the field *SecurityTradingStatus* (326) for each of these instruments.

Eurex Instrument State	***instrument state change* message / *mass instrument state change* message**
	FIX *SecurityTradingStatus (326)* / FIX *SecurityMassTradingStatus (1679)*
Closed	200 = Closed
Restricted	201 = Restricted
Book	202 = Book
Continuous	203 = Continuous
Opening Auction	204 = Opening Auction
Opening Auction Freeze	205 = Opening Auction Freeze
Intraday Auction	206 = Intraday Auction
Intraday Auction Freeze	207 = Intraday Auction Freeze
Volatility Interrupt Auction	208 = Circuit Breaker Auction
Volatility Interrupt Auction Freeze	209 = Circuit Breaker Auction Freeze
Closing Auction	210 = Closing Auction
Closing Auction Freeze	211 = Closing Auction Freeze

Table 7: *Instrument states*

The field *FastMarketIndicator (2447)* is also contained in the *mass instrument state change* message; each instrument state message also contains the information about whether the product that the instrument belongs to is in a Fast Market state. This implies that a *mass instrument state change* message is sent when a product is set to Fast Market (or back) without a change in the instrument states.

[7] Instrument types distinguish simple instruments (option series, futures contracts) and various types of complex instruments

The status of the instrument (as opposed to the instrument state) distinguishes active and published instruments and is contained in the field *SecurityStatus* (965).

4.3 Overview of the various message types

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

4.3.1 Eurex RDI

Service messages:

- *Technical heartbeat message* is sent out periodically by the Eurex system on every multicast address and on a specific port assigned for the *technical heartbeat*; it consists of a FAST reset message only. The purpose of the heartbeat message is network related only[8].
- *Functional beacon message* (Eurex RDI) contains the last valid *MsgSeqNum* and is only sent on the reference data incremental feed when there is no activity for a certain amount of time. No functional beacons are sent for the Eurex RDI because the snapshots act as a functional beacon.

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 snapshot message* contains product specific reference data.
- *Instrument snapshot message* contains a snapshot of instrument specific reference data and also contains the reference data related to complex and flexible instruments that existed at start-of-day.
- *Instrument incremental message* used with intraday added complex and flexible instruments. Identical messages are also sent on the market data incremental feed of the Eurex EMDI as well as on the market data feed of the Eurex MDI.
- *Variance futures status message* used to convey information specific to variance future instruments either at the start of day or intra-day.
- *Total return futures status message* used to convey information specific to total return future instruments either at the start of day or intra-day.

4.3.2 Eurex EMDI/MDI

Service messages:

- *Technical heartbeat message* is sent out on all multicast addresses of the Eurex EMDI/MDI. The description is the same as for Eurex RDI.
- *Functional beacon message* (Eurex EMDI) contains the last valid *MsgSeqNum* of each product and is only sent on the market data incremental feed when there is no activity in a product for a certain amount of time. No functional beacons are sent for the Eurex MDI because the snapshots act as a functional beacon.

Data messages:

- *Depth snapshot message* is used to send a snapshot of all price levels 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 receive updates on the initial order book.
- *Top Of Book Implied message* is used to send top of book information resulting from synthetic Inter Product Spread (IPS) matching opportunities.

[8] It is used to keep the Spanning Tree alive.

- *Product state change message* is used to publish the state of the Eurex products.
- *Mass instrument state change message* provides the state information for all instruments of a product. This message can publish different states for instruments of the same product, e.g. in case of a volatility interruption the front month could be in a different state than the back month.
- *Instrument state change message* provides state information for a single instrument.
- *Quote request message* provides requests to market makers to enter quotes for specified instruments.
- *Cross request message* is sent once a participant announces the intention to enter a cross trade.
- *Complex instrument update message* is used to publish new complex instruments. This message is sent via the market data incremental feed of the Eurex EMDI and the market data feed of the Eurex MDI. A message is sent for each newly created complex instrument.
- *Flexible instrument update message* is used to publish new flexible instruments. This message is sent via the market data incremental feed of the Eurex EMDI and the market data feed of the Eurex MDI. A message is sent for each newly created flexible instrument.

A detailed description of the message types listed above is given in section 11, Detailed data feed description and layout.

4.4 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 **without** a quantity. If a potential auction price is calculated, it is also sent without the quantity. 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 Eurex ETI in recoverable form.
- Prices for external underlyings are **not** provided. These prices will continue to be available in multicast form via the current Eurex system.
- 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.
- Implied prices are only sent for Best Market, they are **not** sent for the order book depth except for top of book.

4.5 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.

Note: FAST 1.2 templates and FAST 1.1 compatible templates are provided.

After the decoding process, the actual FIX message can be built by applying the FIX structure to the decoded message. The detailed process is shown in Part II, FIX/FAST-Implementation.

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

4.6 Freedom of choice

Eurex does not need to provide any software for accessing the services offered. The Eurex 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.

4.7 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 Eurex service fail-over scenarios. For this purpose, special test scenarios are offered in a simulation environment.

4.8 Hours of operation/availability of messages

- The product state "*Pre-Trading*" (*TradingSessionSubID (625)* = 1) for many of the EUREX products begins at **7:30 CET**.
- "*Post-Trading*" (*TradingSessionSubID (625)* = 5) for some Eurex products lasts until **22:30 CET**.

For detailed information on trading hours please refer to:
www.eurexchange.com > Trading > Trading Calendar > Trading Hours.

- Eurex 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 stops while in state "Post-End-Of-Day". During that period *depth snapshots* are sent.

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

Table 8: *Availability of messages per instrument state*

Part II
How to guide

5 FIX/FAST-Implementation

This chapter describes the message structure for the three interfaces. It also provides the basic FAST-rules used by the interface and describes the basic steps from receiving a FAST datagram, decoding it and building FIX-messages out of it.

The FAST 1.2 specification is provided as an extension to the FAST 1.1 specification. The documents can be found under the following links:

FAST Specification (Version 1.1)
www.fixprotocol.org > Technical Specifications > FAST Protocol > FAST Protocol Specifications > FAST Specification Version 1.1

FAST version 1.2 Extension Proposal
www.fixprotocol.org > Technical Specifications > FAST Protocol > FAST Protocol Specifications > FAST Extension Version 1.2

5.1 Structure of Messages

The three public interfaces disseminate data in UDP datagrams in network byte order also known as big endian byte order. This includes vector encoded numbers. A UDP datagram has the following structure:

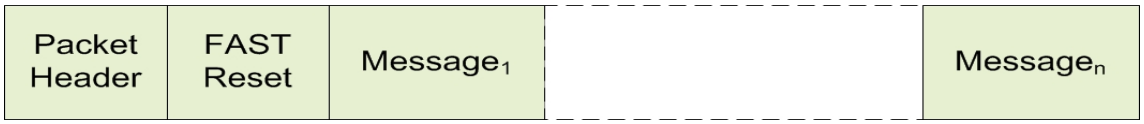


Figure 3: *Structure of a UDP datagram*

- The UDP datagram starts with the packet header message as shown in section 11.1.2.
- Followed by a FAST reset message.
- Followed by the actual message ($Message_1$).
- Possibly followed by one or more messages ($Message_2$ - $Message_n$).

Each message shown in the picture above has the following sub structure:

- PMAP (Presence Map).
- TID (Template ID).
- Data Part.

This is shown in the following diagram:

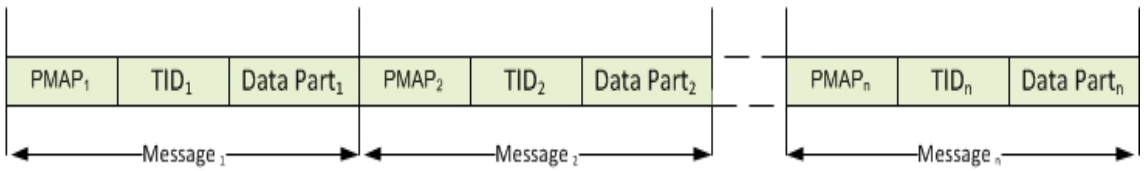


Figure 4: *Structure of consecutive messages within one datagram*

One UDP datagram contains one or more FAST encoded FIX 5.0 SP2 messages. The UDP protocol adds a 28 byte header to every packet (20 byte IP header plus 8 byte UDP protocol header). Due to the unreliable nature of UDP, every UDP datagram is self contained; there is no dependency across datagrams.

5.2 FAST terminology

5.2.1 FAST reset message

The Eurex Market Data Interfaces use **global dictionary** scope for FAST operators[9]. All operators share the same dictionary regardless of the template and application type. The *FAST reset* message is inserted at the start of every datagram to explicitly reset all the dictionaries.

5.2.2 Presence Map (PMAP)

The presence map is a bit combination indicating the presence or absence of a field in the message body, one bit in the PMAP for each field that uses a PMAP bit according to the FAST type. The allocation of a bit for a field in the presence map is governed by the FAST field encoding rules.

5.2.3 Template ID (TID)

The template identifier is represented by a number (integer) and points to a specific FAST template which describes the layout and characteristics of the message to be decoded. The FAST XML files are provided in section 13, FAST templates.

FAST uses templates to reduce redundancies within a message by using the following methods:

- The order of fields within the FAST message is fixed, so the field meaning is defined by its position in the message and there is no need to transfer the field tag to describe the field value.
- The templates specify the order and occurrence of message fields like type, presence and operators.

The following list contains the message types and their corresponding template identifiers used with the three Eurex interfaces:

[9] The dictionary scope should always be derived from the template definition.

Message	TID Eurex RDI	TID Eurex EMDI	TID Eurex MDI
Functional Beacon (aka Functional Heartbeat)	109	109	-
Packet header for Eurex RDI / EMDI / MDI	72	62	67
FAST Reset Message	120	120	120
MarketDataReport	125	-	-
ProductSnapshot	122	-	-
InstrumentSnapshot	123	-	-
InstrumentIncremental	121	-	-
VarianceFuturesStatus	162	-	-
TotalReturnFuturesStatus	514	-	-
ComplexInstrumentUpdate	-	124	105
DepthSnapshot	-	93	101
DepthIncremental	-	94	102
QuoteRequest	-	95	106
CrossRequest	-	96	107
ProductStateChange	-	97	108
MassInstrumentStateChange	-	99	104
InstrumentStateChange	-	98	103
TopOfBookImplied	-	512	513
FlexibleInstrumentUpdate	-	100	110

Table 9: *Template identifiers for Eurex RDI/EMDI/MDI*

Note: The template id for the *packet header* will change in future releases and can be used to identify the software release.

Example: The TID=67 indicates the *packet header* for Eurex MDI in the current release. In the next release the TID for the *packet header* will change to another value.

5.2.4 Dictionaries

A dictionary is a cache in which previous values are stored. FAST operators (-> 5.2.6) make use of the previous values.

5.2.5 Stop bit encoding

Most FAST fields are stop bit encoded, each byte consists of seven *data bits* for data transfer and a *stop bit* to indicate the end of a field value. An exception from this rule are Byte Vectors as they are used in the *packet header* of Eurex EMDI/MDI/RDI.

5.2.6 FAST operators

Field operators are used to remove redundancies in the data values. Message templates are the meta-data for the message and are provided earlier. When the messages arrive, the receiving application has complete knowledge of the message layout via the template definition; it is able to determine the field

values of the incoming message.
The following FAST operators are used in Eurex EMDI/MDI/RDI:

- delta.
- copy.
- constant.
- default.
- increment.

For more information on the new FAST 1.2 features please refer to:
www.fixprotocol.org > Technical Specifications > FAST Protocol > FAST Protocol Specifications > FAST Extension Version 1.2.

5.3 Decoding the FAST-message

The FAST messages need to be decoded by means of the FAST templates. The FAST templates provide all necessary information to decode a message such as data types (e.g. uInt32), field names (e.g. MsgType), FIX tags (e.g. 35) and FAST operators (e.g. increment). The FAST templates also contain information about repeating groups (sequences).

A typical example for a XML FAST template with a repeating group is shown in figure 21 of section 14.1, Example for a XML FAST template.

5.4 Transfer decoding

Transfer decoding describes the process of how the fields are decoded from the FAST format. For further information, please refer to section 10 of the FAST Specification Version 1.1. Transfer encoding describes the opposite process.

5.5 Composing the Actual FIX-Message

A typical FAST decoder would not deliver FIX messages after the decoding process. In order to compose FIX messages, applications need to apply additional rules.

The sequence of FIX-fields after composing the FIX-message on participants' side is not governed by the FIX-layout of the messages, i.e. the fields names of the FIX-message do not need to be in the same sequence. The FIX message, however, needs to fulfill the minimum requirement:

- BeginString(8) in the Standard Header must be the first tag in the message.
- BodyLength(9) in the Standard Header must be the second tag in the message.
- MsgType(35) in the Standard Header must be the third tag in the message.
- CheckSum(10) Standard Trailer must be the last tag in the message.

5.6 New features in FAST version 1.2

The following new features from the FAST 1.2 protocol are used:

- **New Type Definition Syntax**: This allows the separation of the "type definitions" from the "type usage" within template definitions.
- **Enumeration**: This feature can be used when there is a fixed set of valid values for a single field.
- **Set (multi-value field)**: This feature can be used when there is a fixed set of valid values which could be sent together as a bit combination instead of using a repeating group. An example for a set would be the field *TradeCondition (277)* in the Depth incremental message. Sets are used to define the valid values for fields.
- **Timestamp Data Type**: The use of this feature allows native support of time stamp fields which becomes increasingly important for the Eurex market data interface. A time stamp is an integer that represents a number of time units since an epoch.

5.7 Data types

The Eurex implementation of FAST utilizes the following FAST data types:

- Decimal
- Length
- String
- uInt32/uInt64/int64
- Byte vector
- Set
- Enum
- Timestamp

5.8 FAST version 1.1 compatible templates

Participants who choose not to upgrade their FAST 1.2 decoders can use FAST 1.1 compatible files offered by Eurex. The following needs to be considered:

- **Enumerations**: As described in the previous chapter enumerations have a list of codes. Participants receive an integer but not the description (meaning) of the integer. Since FAST 1.1 does not support enumerations this description of codes needs to be taken from the valid values provided with Eurex Market + Reference Data Interfaces - XML FAST Templates. These files could be found on www.eurexchange.com > Technology > Eurex Exchange's T7 > System documentation.
- **Sets**: Similar to enumerations, however, participants receive a bitmap and multiple items from the list. The items need to be taken from the valid values provided with Eurex Market + Reference Data Interfaces - XML FAST Templates. These files could be found on www.eurexchange.com > Technology > Eurex Exchange's T7 > System documentation.

The FAST version 1.2 Extension Proposal available at www.fixprotocol.org > fastspec describes how the encoded field (wire format) value looks.

Example for enumeration: *TradingSessionID* (336) can have one of the following values as defined in the FAST 1.2 XML files:

```
<define name="TradingSessionID">
  <enum>
    <element name="1" id="Day"/>
    <element name="3" id="Morning"/>
    <element name="5" id="Evening"/>
    <element name="6" id="AfterHours"/>
    <element name="7" id="Holiday"/>
    <copy/>
  </enum>
</define>
```

The wire format of the values 1, 3, 5, 7 is 0, 1, 2, 3, i.e. each value is represented by an index. Enumerations are not defined in the FAST 1.1 XML files. When the decoder receives a 3 he needs to know that it means "Holiday".

Example for set: *TradeCondition* (277) can have one or more values as defined in the FAST 1.2 XML files:

```
<define name="TradeConditionSet">
  <set>
    <element name="U" id="ExchangeLast"/>
    <element name="R" id="OpeningPrice"/>
    <element name="AX" id="HighPrice"/>
    <element name="AY" id="LowPrice"/>
    <element name="AJ" id="OfficialClosingPrice"/>
    <element name="AW" id="LastAuctionPrice"/>
    <element name="k" id="OutOfSequenceETH"/>
    <element name="BD" id="PreviousClosingPrice"/>
    <element name="a" id="VolumeOnly"/>
  </set>
</define>
```

The wire format of the values U, R, AX, AY, AJ, AW, k, BD, a is 1, 2, 4, 8, 16, 32, 64, 128, 256, i.e. each value is represented by a different bit. The values can be added together to form combinations of the values. If U, AX are sent then 1 + 4 = 5 are the encoded field values.

Sets are not defined in the FAST 1.1 XML files. When the decoder receives a 5 he needs to know that it is a combination of 1 and 4 which is "ExchangeLast" and "HighPrice".

6 Description of a typical trading day

This chapter describes a typical 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:

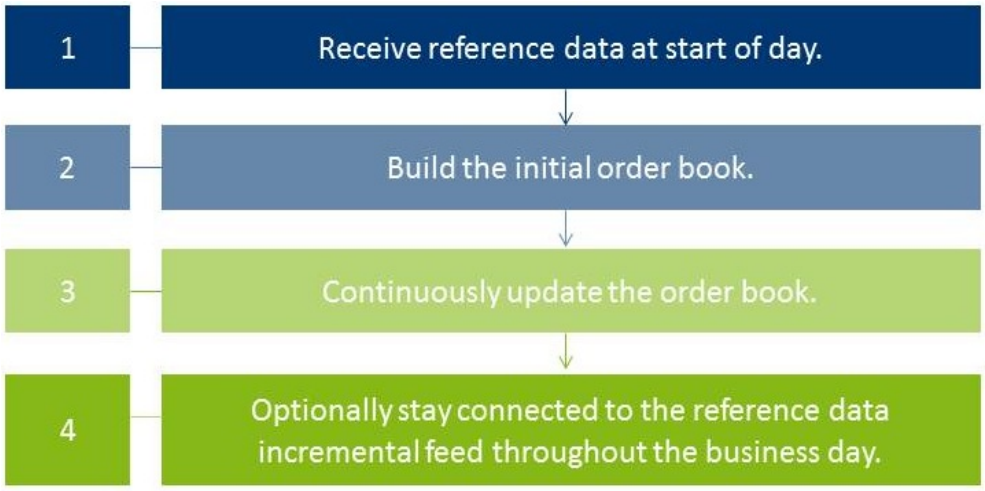


Figure 5: *Typical trading day*

6.1 Start of day operation

Before processing any market data, receiving applications need to retrieve technical and functional information via the Eurex RDI. Alternatively, reference data can be received in file format (Reference Data from File). A detailed description of the reference data feeds and messages is provided in section 9.1, Reference data messages.

At start-up, reference data must be processed to create the initial order book baseline.

6.2 Receiving reference data via Eurex 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 functional and technical parameters.
- **Instrument snapshot**, **variance futures status** and **total return futures status** to receive instrument details.
- **Instrument incremental** to receive intraday creation of complex and flexible instruments.

Port information and multicast addresses for the reference data feeds as well as the address ranges for market data are published in the document "Network Access To Exchange Applications" and in section 12, Multicast addresses. 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 9.2, General reference data rules. However, the basic steps in order to receive reference data are the following:

1. Listen to the reference data incremental feed and start buffering the messages. If an application starts listening to the reference data messages early enough, there are no messages available.
2. Listen to the 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 two values:

- *MDReportCount*, containing the number of reference data snapshot messages in the initial snapshot cycle.
- *LastMsgSeqNumProcessed*, containing the sequence number of the last message at the end of the snapshot cycle; This could be a snapshot or an incremental message.

Process all messages of the snapshot until you encounter the *market data report* message denoting the end of the snapshot cycle.

3. At this point you need to complete the list with the messages received on the incremental feed since you started listening. But only after you have discarded all messages having[10]:
   MsgSeqNum <= *LastMsgSeqNumProcessed* - *MDReportCount*.
4. Store the reference data information for future use.
5. Join the market data incremental feed of EMDI or the market data feed of MDI in order to receive additional reference data changes.
6. Leave the reference data snapshot and incremental feeds.

Note: Applications starting early do not require steps 1 and 3 since no incremental message exists at this time.

New complex instruments predefined by the exchange are also sent in *instrument incrementals* before the start of the actual trading.

Note: Participants interested in complex or flexible instruments should use the *complex instrument update* and *flexible instrument update* messages. These are published on the market data incremental feed of the Eurex EMDI as well as on the market data feed of the Eurex MDI. They are published faster than the instrument incremental message of the reference data incremental feed.

6.3 Receiving reference data file (RDF) at start of day

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

The initial reference data file generated at start-of-day contains the "reference data snapshots" available from the previous day. During the actual trading multiple incremental files are created as complex and flexible instruments are added.

New complex instruments predefined by the exchange are also sent in incremental files before the start of the actual trading.

In case a receiving application starts late, each of the intraday Reference Data Files in addition to the Start-Of-Day Reference Data File must be applied. Start-Of-Day and Intraday Reference Data Files are available via the *Common Report Engine*.

Note: In case a late starting application uses the Start-Of-Day Reference Data File without the intraday files, the intraday created complex instruments remain unknown and hence order book data may be received for unknown instruments.

[10] The snapshot and incremental feeds have a different sequence number range.

6.4 Build the initial order book

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

Note: Sequence numbers contained in the market data messages are incremented per product.

6.4.1 Build the initial order book with the Eurex EMDI

For each instrument within the desired products do the following:



Figure 6: *Eurex EMDI initial order book*

6.4.2 Build the initial order book with the Eurex MDI

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



Figure 7: *Eurex MDI initial order book*

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

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

6.5 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:

6.5.1 Update the order book with the Eurex EMDI

As long as the *MsgSegNum* values for the *depth incremental* message are contiguous per product do the following [11]:

- 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.

[11] The reason is that the unreliable nature of UDP multicast can cause packets to arrive delayed, in incorrect sequence or may be missing.

6.5.2 Update the order book with the Eurex MDI

As long as the *MsgSegNum* values for the *depth incremental* message are contiguous per product do the following[11]:

- Keep applying all *depth incremental* as well as *depth snapshot*[12] messages to the current order book.

Each incremental message can carry different update instructions with the "update action" (New, Change, Delete, Delete From, Delete Thru, Overlay).

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

[12] only if the *RefreshIndicator (1187) = Y*

7 Recovery

Due to the unreliable nature of UDP multicast it is possible that some packets may either be delayed, arrive in the incorrect order or even 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.

7.1 Detecting duplicates and gaps by means of the packet header

The *packet header* allows receiving applications to identify **identical** packets between Service A and Service B. This is achieved by a simple memory comparison on the first 9 bytes for Eurex EMDI or 8 Bytes for Eurex MDI of a datagram containing *SenderCompId* and *PacketSeqNum* as shown in figure 19, Packet header (Eurex EMDI) and figure 20, Packet header (Eurex MDI /Eurex RDI). 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*.

> **Note:** Packets with the same *SenderCompID* (field length: 1 Byte) have contiguous sequence numbers per multicast address / port combination.

This means that field *PacketSeqNum* can be used not only to detect duplicates but also to detect missing packets. *PacketSeqNum* is a Byte vector and therefore not stop bit encoded as per the FAST specification.

The *packet header* itself does not contain any product information. In order to find out which product is missing, the product level sequence number must be used in addition to the packet level sequence number; the packet needs to be decoded further down to the message level. This leaves participants with **two recovery options** when a gap in the *PacketSeqNum's* of the *packet header* is detected.

Example:
A single multicast address carries products FDAX and FGBL, but the participant is only interested in FGBL.

I. Pessimistic approach: The receiving application assumes that FGBL is part of the missing packet: It immediately starts recovery actions [13] 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 FGBL is not part of the message which is inside the lost packet.

II. Optimistic approach: The receiving application assumes that FGBL is not part of the missing packet: It waits for the next message on the same service and decodes the packet up to the message level to find out if a packet for FGBL has been lost before triggering recovery actions.

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

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

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

[13] by means of the other service (live-live concept) or by listening to the *depth snapshot*

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*.

	Service A:			Service B:	
Time	MsgSeqNum	Message	Time	MsgSeqNum	Message
10:30:00	206	New 151@4	10:30:01	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

Table 10: *Recovery via Service B (live-live concept)*

As the above example shows, the same information is delivered on Service A and B. While *MsgSeqNum* = 208 is missing on Service A, it is provided on 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 discardes 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 6.4, Build the initial order book.
- The maximum expected recovery interval for a particular feed can be obtained in the product snapshot message of the Eurex RDI snapshot feed (field: *MDRecoveryTimeInterval (2565)*).

7.3 Delayed packets

The following example indicates 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

Table 11: *Packets arriving in correct sequence*

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

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, Eurex Market Data Interface sends incremental messages in ascending *MsgSeqNum* order, but they might arrive in an incorrect 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

Table 12: *Delayed Packet 207*

In this example, message 207 is delayed within the network, allowing message 208 to arrive first.

A correct communications layer responds as follows:

1. Release message 206 to the application immediately on arrival.
2. On arrival of 208, recognises that 207 is missing.
3. Start an appropriate timed operation to trigger the recovery actions if the out-of-sequence message 207 fails to arrive in a reasonable time.
4. Assuming that 207 arrives within that reasonable time, release 207 and then 208 to the application in that order and cancel the timed recovery action.

7.4 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

Table 13: *Missing seqNum 207*

The correct behaviour in this instance is:

1. Release message 206 immediately on arrival.
2. Hold on to 208 because it is out-of-sequence, and initiate timer-based recovery actions.
3. Hold on to 209 for the same reason. Timer-based recovery actions are already pending for this product, so do not reset the timer.
   (a) Even though message 209 is a "New" operation, it may be unsafe to apply 208 and 209 because we do not know what 207 contains.
4. If the missing message (207) fails to arrive within the allowed time:
   (a) 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*.
   (b) In case the recovery from the respective other service (A or B) fails: initiate recovery via snapshots.

7.4.1 Recovery (Eurex 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 Figure 2, Differences between the interfaces).

Incrementals are sent whenever there is a change of the order book (event-driven); snapshots are sent periodically in 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.

MsgSeqNum	Product	LastMsgSeq-NumProcessed	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	$A_2{}^I$

Table 14: *Snapshots and incrementals within the Eurex EMDI*

The appropriate recovery action for missing *depth incrementals* is the same as the logic described in section 6.4.1, Build the initial order book with the Eurex EMDI.

There are some additional points to be aware of when performing recovery:

- During recovery, applications should be prepared to receive *depth incremental* messages for instruments they didn't know existed. This can occur if a strategy creation event (via a *complex instrument update* on the market data feed) is missed due to packet loss. In this case, applications must consult the reference data snapshot feed to obtain the strategy description.
- *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.
- The *LastMsgSeqNumProcessed (369)* is not necessarily the same for all instruments belonging to a product on the market data snapshot feed.

Note: The market data snapshot feed does not contain any "start" or "end" messages to delineate the cycle.

There are two ways to determine when to leave the snapshot feed during recovery:

Method 1: Process specific products

For each *SenderCompID (49)* contributing to the market data snapshot feed, *depth snapshot* messages are grouped by product as illustrated below:

P_1I_1 | P_1I_2 | P_1I_3 | P_1I_n | P_2I_1 | P_2I_2 | P_2I_3 | P_2I_n | P_3I_1 | P_3I_2 | P_3I_3 | P_3I_q | [...]

with:
P_n: Product n
I_q: Simple, flexible or complex instrument q for product n

Depth snapshots for instruments in the same product will often all appear in the same packet, but this should not be relied upon as it is not true when the amount of data is simply too great to fit into a single packet, and under certain other technical conditions on the exchange.

A change of product *MarketSegmentID (1300)* for a given *SenderCompID (49)* indicates the end of the *depth snapshot* messages for the respective product. This allows applications to easily determine when they've received a snapshot for every instrument in the products they're interested in and leave the snapshot feed.

Method 2: Process an entire depth snapshot cycle

It's also easy for an application to listen to an entire snapshot cycle.

Applications can determine when they've seen an entire snapshot cycle simply by remembering the *SecurityID (48)* of the first *depth snapshot* message they saw from each *SenderCompID (49)*.

When they see the same *SecurityID (48)* again for each *SenderCompID (49)*, they know that a complete depth cycle has been seen and can leave the snapshot feed.

Note: Receiving applications also need to consider *depth snapshot* messages for newly created complex instruments.

Note: If a failover occurs during snapshot processing the *SenderCompID (49)* for the affected partition changes and the snapshot cycle for that partition starts again.

7.4.2 Recovery (Eurex 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:

MsgSeqNum	Product	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	$A_2^{S,I}$
27	B	depth incremental	$A_2^{S,I}$

Table 15: *Snapshots and incrementals within the Eurex MDI*

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

All depth incremental messages for product A with a lower sequence number than the next market data snapshot message for product 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 product A with *MsgSeqNum* ≥ 11 can be applied to the latest snapshot with *MsgSeqNum* = 10.

Note: *LastMsgSeqNumProcessed* is not necessary for recovery purposes in the Eurex MDI.

8 Various time stamps in EUREX and how to use them

The various Eurex Exchange's T7 timestamps mentioned throughout the document, are taken at high-frequency gateways, matching engines and market data servers, both in production and simulation. They are also provided through messages sent on Eurex EMDI, Eurex MDI and Eurex EOBI feeds. These can be used to analyze one way transport times. To reiterate, 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 the following locations:,

Gateway:

- On entry to the Gateway.

MatchingEngine:

- 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 (see Eurex ETI Manual).

Market Data (Eurex EMDI, Eurex MDI and Eurex EOBI):

- *SendingTime* for order book delta and snapshot messages,
- addtionally timestamps from Matching Engine such as *PriorityTimestamp*, *TransactTime*, *GatewayInTimestamp*, etc. are provided on market data messages.

The following picture provides an overview of Eurex Exchange's T7 timestamps:

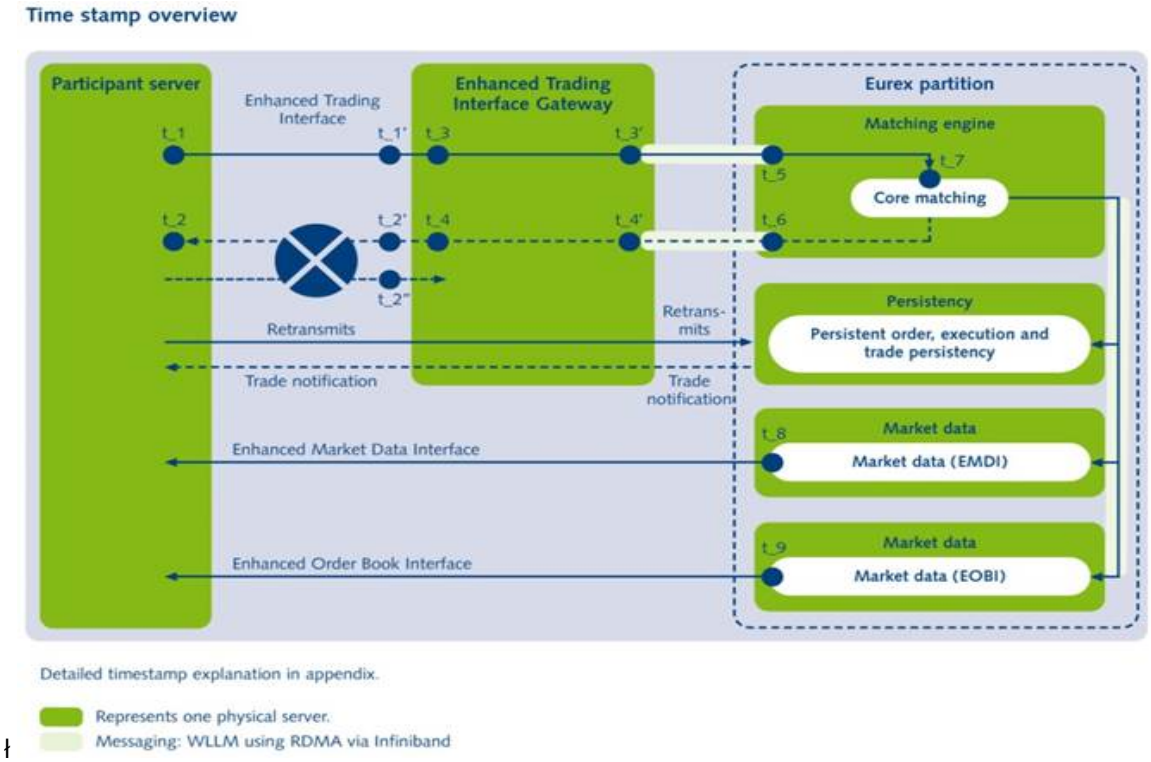


Figure 8: *Timestamp Overview*

The following table lists the mapping of Eurex Exchange's T7 timestamps:

Timestamp	Semantic	FIX fields	Description
t_3	Gateway request in	RequestTime (5979)	Provides the time the Eurex application has read an inbound message on a gateway from the TCP socket.
t_3‘	Gateway request out	RequestOut (7764)	Provides the time the Eurex application has sent an outbound message from a gateway to the matching engine.
t_4‘	Gateway response in	ResponseIn (7765)	Provides the time the Eurex 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 Eurex 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, strategy creation, execution or as transaction timestamp for others.
t_5	Matching engine in	TrdRegTSTimeIn (21002)	Provides the time the Eurex application has received an inbound message on a matching engine.
t_6	Matching engine out	TrdRegTSTimeOut (21003)	Provides the time the Eurex application has sent an outbound message from a matching engine.
t_8	Eurex EMDI out	SendingTime (byte vector)	Provides the sending time when Eurex EMDI has put the datagram on the wire.
t_9	Eurex EOBI out	TransactTime (60)	Provides the sending time when Eurex EOBI has put the datagram on the wire.

Table 16: *Timestamp mapping*

9 Important topics with use cases and examples

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

9.1 Reference data messages

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

- **Snapshot feed of Eurex RDI** provides a snapshot of all products and instruments (simple, complex and flexible) and is sent out on a regular basis throughout the day. Additions of complex and flexible instruments are incorporated into the next snapshot cycle.
- **Incremental feed of Eurex RDI** is event triggered and provides real-time information about complex instruments[14] and flexible instruments that are added intraday and about variance futures and total return futures status updates. Any change is incorporated within the next snapshot cycle.
- **Market data incremental feed of EMDI** is event triggered and provides real-time information about complex and flexible instruments that are added or inactivated intraday on the same channel as market data.
- **Market data feed of MDI** is event triggered and provides real-time information about complex and flexible instruments that are added or inactivated intraday.

The following messages are sent via different feeds:

a) **Snapshot feed of Eurex RDI**:

- ***Product snapshot*** for products available at start of day.
- ***Instrument snapshot*** for simple, complex and flexible instruments available at start of day.
- ***Variance futures status*** for variance futures instruments.
- ***Total return futures status*** for total return futures instruments.
- ***Instrument incremental*** for complex and flexible instruments added intraday.
- ***Market data report*** indicates the start of reference data (*MDReportEvent=1*).
- ***Market data report*** indicates the end of reference data (*MDReportEvent=2*).

b) **Incremental feed of Eurex RDI**:

- ***Instrument incremental*** for complex and flexible instruments added intraday.
- ***Variance Futures Status*** when the conversion parameters have been approved.
- ***Total Return Futures Status*** when there is a change in any of the conversion parameters.

c) **Market Data incremental feed of EMDI**:

- ***Complex instrument update***[15] for complex instruments added intraday.
- ***Flexible instrument update*** for flexible instruments added intraday.
- ***Instrument state change*** or ***mass instrument state change*** for complex instruments inactivated or re-activated intraday.

d) **Market data feed of MDI**:

- ***Complex instrument update*** for complex instruments inactivated or re-activated intraday.
- ***Flexible instrument update*** for flexible instruments added intraday.

[14] No product information is delivered

[15] The layout is the same as on the instrument incremental message except for the field *SecurityExchange (207)*, *SecurityStatus (965)* and *SenderCompID (49)*

9.2 General reference data rules

9.2.1 General structure of the snapshot cycle

A snapshot cycle consists of (see figure 9):

- A ***market data report message*** (*MDReportEvent* = 1 = "**StartOfReferenceData**").
- A sequence of a *product snapshot* followed by the associated *instrument snapshots* (simple, flexible and complex), repeating for all products and instruments.
- A dynamically growing sequence of *instrument incremental* messages.
- Finally ***market data report message*** (*MDReportEvent* = 2 = "**EndOfReferenceData**").

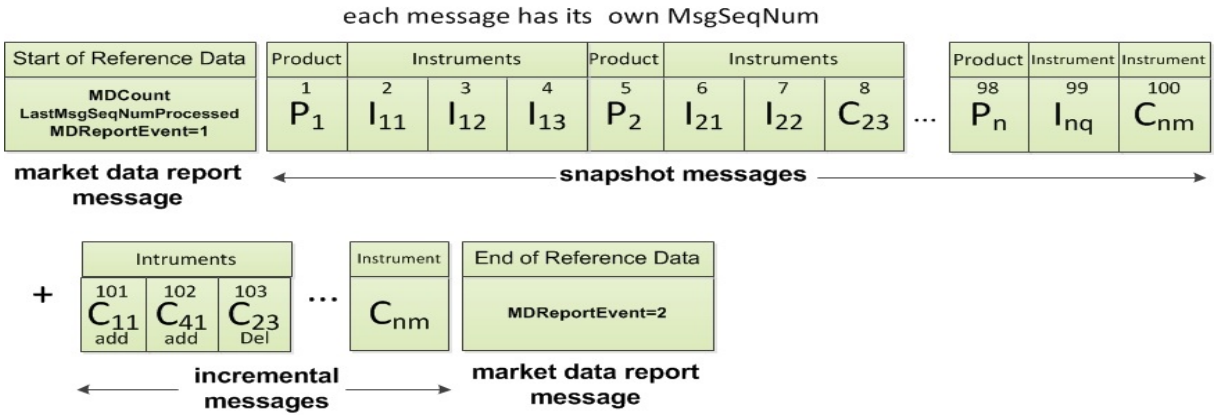


Figure 9: *Entire snapshot cycle on the Eurex RDI snapshot feed*

with:
P_n: Product n
I_{nq}: Simple instrument q for product n
C_{nm}: Flexible or complex instrument m for product n

Product and *instrument snapshot* messages are sent for the initial set of products and instruments. While the snapshots do not change intraday, the number of incremental messages increases if, e.g., complex and flexible instruments are added. Figure 10 illustrates how more *instrument incrementals* are added over the course of n cycles:

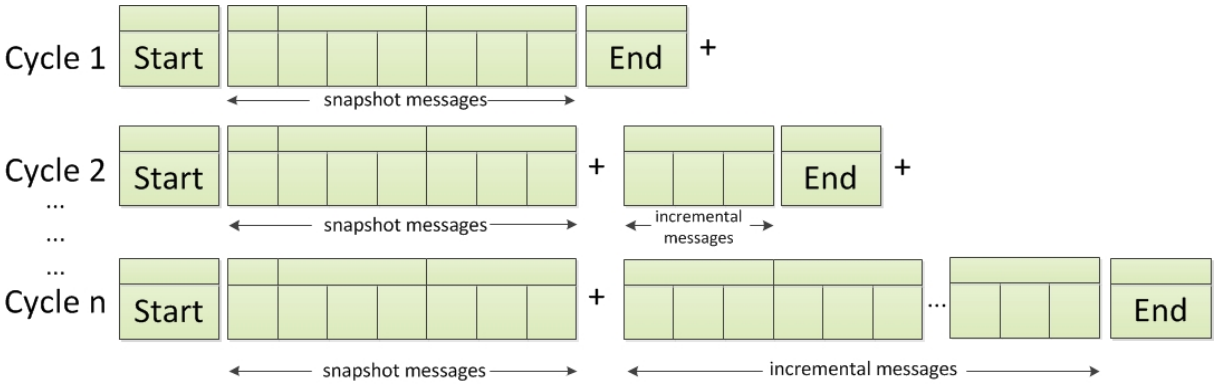


Figure 10: *Reference data with constant snapshots and extending incrementals*

Note: Overnight changes to the products and instruments are reflected in the *reference data snapshot messages* after the technical start on the next business day.

9.2.2 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. The value represents the number of product level messages + the number of instrument level messages (simple instrument, flexible instrument, complex instrument, variance futures status and total return futures status) at start-of-day.

  If a failure of Eurex RDI occurs the number of messages in the reference data snapshot and herewith *MDReportCount* (2536) changes.

- ***LastMsgSeqNumProcessed (369)***: This is the *MsgSeqNum* value of the last reference data message (snapshot or incremental) in the snapshot cycle (products and instruments share a single sequence number).

  Note: The number of incremental updates in a snapshot cycle can be calculated as:
  Number of incremental updates = *LastSeqNumProcessed* - *MDReportCount*.

- ***TotNoMarketSegmentReports (2537)***: Contains the number of product level messages sent in the snapshot cycle. This value remains constant intraday as products are not created or deleted intraday.

- ***TotNoInstrumentReports (2538)***: Contains the number of instrument level messages sent in the snapshot cycle. This value changes as more flexible and complex instruments are created intraday or as variance futures status and total return futures status messages are disseminated.

TotNoMarketSegmentReports (2537) and *TotNoInstrumentReports (2538)* can be used as a sanity check and to pre-allocate the product and instrument containers.

The *market data report* message of type **"EndOfReferenceData"** marks the end of reference data messages and does not contain any counters.

The following examples highlight a few scenarios which require special attention. The focus lies on the reference data snapshot feed which provides constant snapshot messages and a variable part with incrementals for flexible [16] and complex [17] instruments.

[16] Participants interested in flexible instruments can also use the *flexible instrument update message* via the faster depth incremental feed of EMDI or the market data feed of MDI.

[17] Participants interested in complex instruments can also use the *complex instrument update message* via the faster depth incremental feed of EMDI or the market data feed of MDI.

9.2.3 Use case 1: Reference data at the start of the reference data service

At the **start** of the reference data service the *reference data snapshot* is sent. Figure 11 shows how snapshots for simple, complex and flexible instruments, which are already in the system, are sent at start of day:

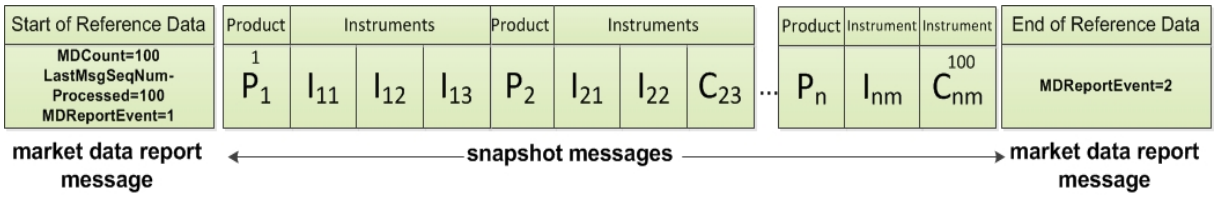


Figure 11: *Reference data snapshot message on the reference data snapshot feed at the start of the reference data service*

The first snapshot cycle can already contain some complex and flexible instruments in the snapshot messages. A *reference data incremental* message does not exist at this time.

9.2.4 Use case 2: Reference data after intraday addition of complex instruments

The next example shows an **intraday** addition of three complex instruments C_{11}, C_{41} and C_{33}. See figure 12). The *reference data incremental messages* for complex instruments C_{11}, C_{41} and C_{33} are appended to the *reference data snapshot messages*:

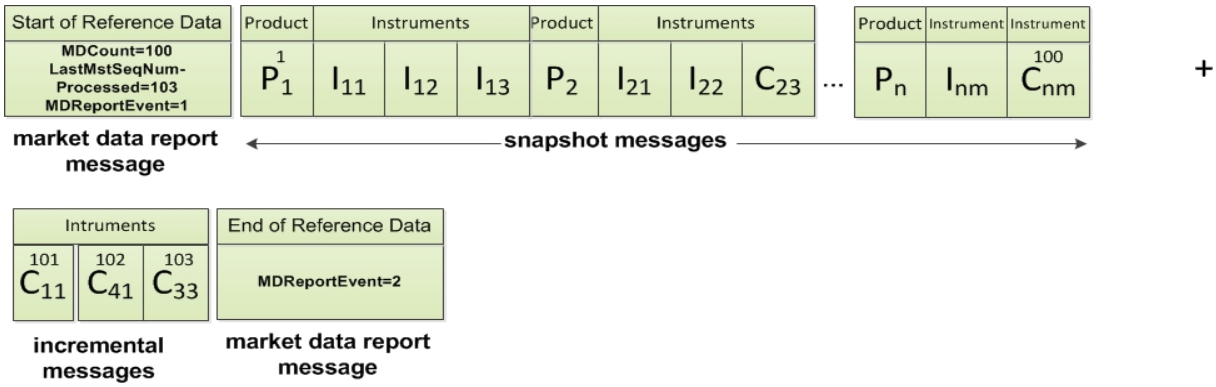


Figure 12: *Reference data snapshot after intraday addition of complex instruments C_{11}, C_{41} and C_{33}*

LastMsgSeqNumProcessed (369) in the *market data report*, type: "Start of Reference Data" increases to 103. The number of incremental messages can be calculated as *LastMsgSeqNumProcessed - MDReport-Count* = 103 - 100 = 3.

New complex instruments predefined by the exchange are sent in *instrument incrementals* and not in *snapshot messages* on the day of creation; the messages are sent before trading starts.

9.2.5 Use case 3: Reference data on the next business day

The complex instruments which still exist on the next business day and which have been sent as *reference data instrument incrementals* on the previous business day, are sent as *instrument snapshot* messages on the next business day, if orders still exist in the respective order books as shown in figure 13:

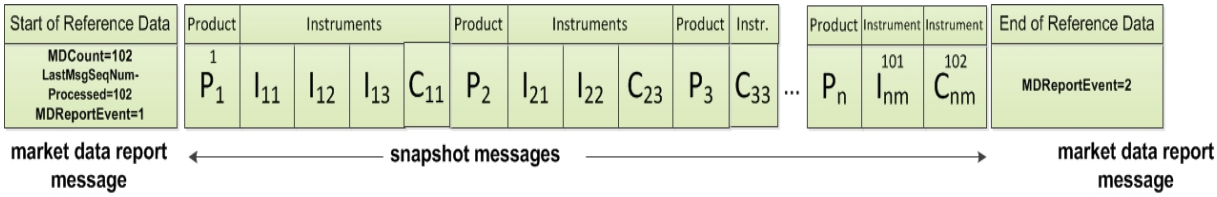


Figure 13: *Reference data snapshot on the next business day*

At the beginning of the next business day[18] *LastMsgSeqNumProcessed (369)* is 102 as this reflects the number of remaining complex instruments C_{11} and C_{33}. C_{41} has already been deleted at the end of the previous business day. At this point in time *MDReportCount (2536)* has increased to 102 as well.

9.2.6 Use case 4: Failover or restart of Eurex RDI

In the event that a Eurex RDI fails, another instance takes over. Receiving applications can detect this by a change of the *SenderCompID* for a marketID and the receipt of a *market data report* message. Applications should respond to this situation as described in section 6.2, Receiving reference data via Eurex RDI at start of day. The same recovery actions apply in case of a complete restart of Eurex RDI.

9.2.7 Use case 5: Chronological order of messages for complex instrument creation

The intraday creation of complex/flexible instruments results in the following sequence of messages:

1. On the **market data incremental feed** of the Eurex EMDI and **market data feed** of the Eurex MDI: A *complex instrument update* (alternatively *flexible instrument update*) message is sent to inform the participant as fast as possible. In case a new complex/flexible instrument has been created the corresponding message is sent prior to the publication of any order book data for the new complex/flexible instrument.
2. On the **reference data incremental feed** of the Eurex RDI: An *instrument incremental* message is also sent with additional fields populated. There is no *product incremental* message.
3. On the **reference data snapshot feed** of the Eurex RDI: An *instrument incremental* message is appended to the end of the current snapshot cycle without removing or changing any of the existing snapshot or incremental messages in the cycle[19]. Therefore the cycle is only extended intraday and never reduced.

9.2.8 Use case 6: Chronological order of messages for complex instrument deletion

For the deletion of a complex instrument, e.g. initiated by Market Supervision, an *instrument state change* or *mass instrument state change* message is sent with *SecurityStatus (965)* set to 2 = Inactive for the specific SecurityID on the **market data incremental feed** of the Eurex EMDI and **market data feed** of the Eurex MDI. No message is sent by the Eurex RDI.

In case the complex instrument is re-created on the same day, an *instrument state change* or *mass instrument state change* message is sent with *SecurityStatus (965)* set to 1 = Active for the specific SecurityID on the **market data incremental feed** of the Eurex EMDI and **market data feed** of the Eurex MDI. No message is sent by the Eurex RDI.

[18] also in case of a feed restart on the exchange side
[19] A complete snapshot cycle is a combination of start, refdata snapshots, refdata incrementals and end message.

9.2.9 Use case 7: Variance Futures Status messages

At the start of day, a variance futures instrument appears on the **reference data snapshot feed** of the Eurex RDI as a regular *instrument snapshot* message immediately followed by a *variance futures status* message. Therefore, for a variance futures instrument, field *TotNoInstrumentReports (2538)* in the *market data report* is incremented by 2.

During the day, a variance futures status message is sent on the **reference data incremental feed** of the Eurex RDI when the conversion parameters have been approved. The same message is then appended to the end of the current snapshot cycle of the **reference data snapshot feed** of the Eurex RDI.

9.2.10 Use case 8: Product pool

A Product Pool is a facility to enable functionalities which depend upon interaction of a set of products. Multiple products can be related by a product pool.

The product pool facility is introduced in the form of an IPS product pool, i.e. a type of product pool, the purpose of which is to support IPS functionality. An IPS product pool plays the role of a product for IPS instruments. I.e. IPS instruments do not belong to a product, but instead IPS instruments belong to an IPS product pool.

The ProductSnapshot message is used to distribute information about product pool parameter settings in the same way as information about product parameter settings. The MarketSegmentID field and the MarketSegment field identify the product pool in the same way as they identify a product. The other fields are filled for product pools, if they are available for product pools. For example, there is no underlying for a product pool, so the fields related to the underlying of a product are not filled, but fields transporting price step tables are filled as they are set up for inter-product spreads on product pool level.

The ProductSnapshot message specifically does not contain information about which products are associated to the pool.

9.2.11 Use case 9: Flexible instruments

The intraday creation of flexible instruments results in the following sequence of messages:

1. On the **market data incremental feed** of the Eurex EMDI and **market data feed** of the Eurex MDI: a *flexible instrument update* message is sent to inform the participant as fast as possible. In case a new flexible instrument has been created the corresponding message is sent prior to the publication of any order book data for the new flexible instrument.
2. On the **reference data incremental feed** of the Eurex RDI: an *instrument incremental* message is also sent with the following fields populated: *SecurityID (48)*, *CFICode (461)*, *SecurityDesc (107)*, *MinPriceIncrement (969) MinPriceIncrementAmount (1146)*, *ContractMultiplier (231)*, *ProductComplex (1227)* , *MaturityDate (541)*, *MaturityMonthYear (200)*, *SecurityExchange (207)*, *SecurityType (167)*, *PutOrCall (201)*, *StrikePrice (202)*, *StrikePricePrecision (2577)*, *OptAttribute (206)*, *ExerciseStyle (1194)*, *InstrumentPricePrecision (2576)*, *SettlMethod (1193)*, *SettlSubMethod (2579)*, *SecurityStatus (965)* and *MarketSegmentID (1300)*.
3. On the **reference data snapshot feed** of the Eurex RDI: an *instrument incremental* message is appended to the end of the current snapshot cycle without removing or changing any of the existing snapshot or incremental messages in the cycle[20]. Therefore the cycle is only extended intraday and never reduced.

9.2.12 Use case 10: Total Return Futures Status messages

At the start of day, a total return futures instrument is published on the **reference data snapshot feed** of the Eurex RDI as a regular *instrument snapshot* message immediately followed by a *total return futures status* message. Therefore, for a total return futures instrument, field *TotNoInstrumentReports (2538)* in the *market data report* is incremented by 2.

During the day, a *total return futures status* message is sent on the **reference data incremental feed** of the Eurex RDI each time there is an update on any field of the message. The same message is then appended at the end of the current snapshot cycle of the **reference data snapshot feed** of the Eurex RDI.

[20] A complete snapshot cycle is a combination of start, refdata snapshots, refdata incrementals and end message.

9.3 General order book rules and mechanics

The Eurex Market Data Interfaces, Eurex EMDI and MDI, provide order book updates from level 1 to the maximum level. The maximum level is provided for each product in the product snapshot records 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 principles apply to order book updates:

- 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 Eurex EMDI.
- Each Eurex EMDI packet relates only to a single product. In other words, although each Eurex EMDI packet may contain multiple messages, those messages will always relate to the same product. This does not apply to Eurex MDI where a single packet may relate to multiple products.
- 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.
- During the product states "Post-Trading" and "End-Of-Day", ToB prices (MDEntryType 0=Bid and/or 1=Offer) are sent for simple and complex instruments using *depth incremental* messages. Furthermore, *depth snapshot* messages continue to disseminate the same during product states "Post-Trading" and "End-Of-Day".
- An implied price is the only element of the group without a price level (for MDEntryType = 0=Bid or 1=Offer). For price levels from 1 to max. price levels, outright prices are distributed. An implied price can either be fully implied or partially implied (for more information please refer to section 9.3.1, Determination of the price sources).
- If two (or more) synthetic prices (with the same price) are created for the Best Market via a different path, then the quantity from the path with maximum quantity is reported for the particular price. An example is provided in section 14.2.4 table 57.
- Top Of Book information resulting from synthetic IPS matching opportunities is disseminated using the Top Of Book Implied message on the Eurex EMDI incremental feed. There is no mechanism to guarantee that the Top Of Book messages are in the same datagram as the *depth incremental* messages. The same information is also sent using depth snapshot messages with the same MDEntry group but with MDBookType=Top Of Book(1) and MDSubBookType=Implied without restrictions(1) / Implied with restrictions(2). For details of the Top Of Book Implied message, see section 11.3.3, Top Of Book Implied message.
- There can be multiple updates in one message. The bid side is updated first followed by the ask side.
- If update instructions "new" or "delete" is sent for an implied price, the order book levels 1-n don't need to be shifted down or up.
- 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 outright price (price level 1)
  - down to the worst price (max. price level configured per product).
  - if the resulting book depth, after each applied individual orderbook 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 a quantity. Receiving applications need to delete a pre existing quantity when an absent value is received during a transition into an auction.

- During an auction, there can either be a crossed or an uncrossed book situation. A crossed book is identified to the user by means of an auction clearing price (*MDEntryType*=Q) (aka indicative or potential auction price). An uncrossed book is identified by means of ToB prices (*MDEntryType* 0=Bid and/or 1=Offer). The visibility of the order book is limited during an auction. Depth information will be explicitly provided again when transitioning from an auction to continuous trading as the user cannot know how much of an order book situation prior to the auction is still valid. Depth information will also be explicitly removed when transitioning from continuous trading to an auction.
- A state transition to Freeze is sent as an *instrument state change* message and does not require any implicit action.
- If the book is crossed, an indicative auction price is calculated and disseminated. The new indicative auction prices are always sent with update action "New".
- Intraday expired instrument information is provided by a *depth incremental* and *instrument state change* message.
- Only the snapshot and incremental messages of the Eurex MDI carry a common and contiguous sequence number per product. The incremental message of Eurex 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.
- Only the best implied price is published. The best implied price will be included in market data only in case it is equal to or better than the best direct price in the respective instrument.
- Whenever the quantity or price of the Best Market changes it is disseminated with update action "New" on the incremental feed. Similarly, the Best Market is removed with update action "Delete".

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

Figure 14 illustrates a typical order book and terminology used in the following chapters.

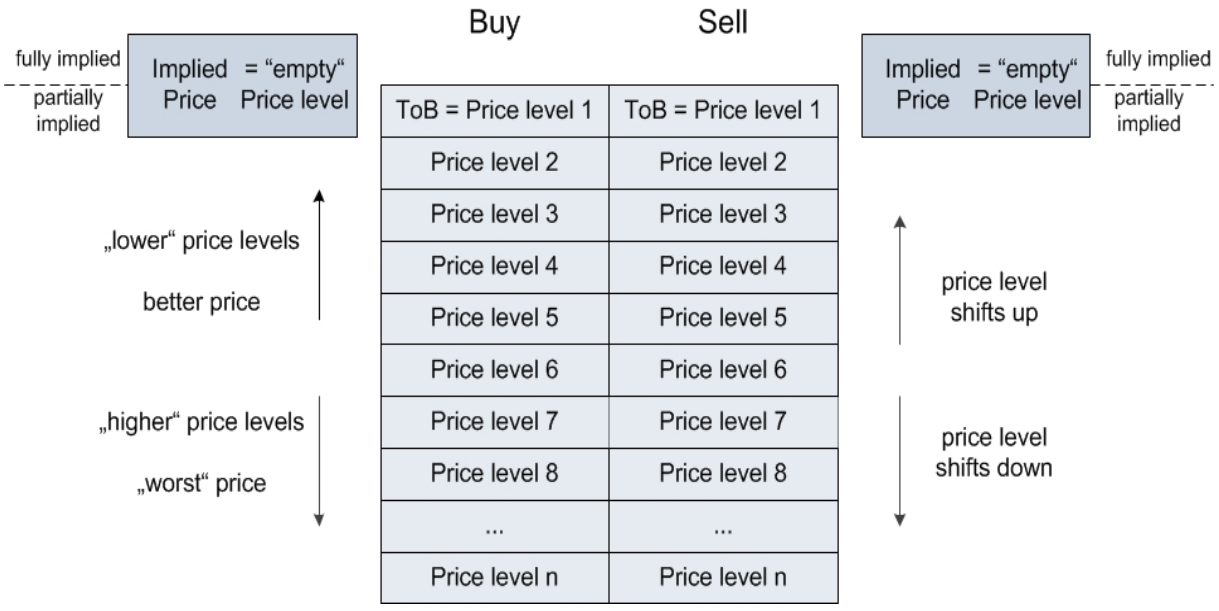


Figure 14: *Typical order book*

An implied price can be either better (fully implied) or the same (partially implied) as price level 1.

9.3.1 Determination of the price sources

Eurex Exchange's T7 supports synthetic matching, where the implied prices from complex instruments can create prices equal or better than the best outright price in the instrument. The implied prices are disseminated in the market data in addition to the prices from outright orders. These prices are shown without a price level. The reported quantities for implied prices and level 1 are not aggregated, i.e. quantities on level 1 are fully outright and do not contain any implied components.

Eurex Exchange's T7 publishes implied prices in market data only in case it is equal to or better than the best outright price in the respective instrument.

In order to find out which situation applies, a price comparison between the implied price (with empty price level) and level 1 (see figure 14) needs to be done:

1. Implied price is better than the outright price at level one -> Fully Implied.
2. Implied price disseminated is equal to the outright price at level 1 -> Partially Implied.
3. Implied price is deleted or absent -> the Best Market price is fully outright and is the same as on level 1.

Examples for all three cases are provided in section 14.2, Example for determination of the price source.

9.3.2 Top Of Book

Simple instruments that are legs of IPS instruments (enabled for synthetic matching) may have synthetic matching opportunities that involve IPS instruments. The corresponding synthetic prices are published with the help of a new Top Of Book Implied data message. Synthetic prices on IPS instruments resulting from their leg instruments are also published via this new Top Of Book Implied data message.

Note: The PriceDepth message continues to contain the order book depth for direct matching and the top-of-book synthetic price derived from synthetic futures spread matching opportunities.

There are two types of synthetic prices due to IPS related matching opportunities that are distributed in the TopOfBookImplied data message:

1. Synthetic prices resulting from synthetic IPS matching opportunities that have no quantity restriction. This price reflects matching opportunities stemming from those IPS instruments that have a leg ratio of 1 in the leg instrument, for which the synthetic price is calculated. Note that the leg ratio condition applies only to that leg. The leg ratios for the other legs of the IPS instrument may have any value.
2. Synthetic prices resulting from synthetic IPS matching opportunities that do have a quantity restriction. This price reflects matching opportunities stemming from those IPS instruments that have a leg ratio greater than 1 in the leg instrument, for which the synthetic price is calculated.

Example: Publish synthetic Top-of-Book without quantity restriction. Bid, Price 106, quantity10.

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1069	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	**0**	Bid
1021	> MDBookType	1	Indicates the book type. Always Top-of-Book = 1.
1173	> MDSubBookType	**1**	Indicates the IPS Implied Volume restriction. 1: Implied volume without quantity restriction. 2: Implied volume with quantity restriction.
48	> SecurityID	8852	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**106**	Price
271	> MDEntrySize	**10**	Quantity
273	> MDEntryTime	t_0	official time of book entry
276	> Quotecondition	F	The quote condition. Set to F if crossed. Otherwise empty.

Table 17: *Top Of Book Implied without quantity restriction*

9.3.3 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 specified price level. [21].
- All existing rows in the order book at the specified and higher levels are to be incremented accordingly. [22].
- Price levels exceeding the maximum specified depth must not be kept in memory.

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

Example: Buy Limit Order, 10@58.22, enters an empty order book:

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1068	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**58.22**	Price
271	> MDEntrySize	**10**	Quantity
346	> NumberOfOrders	1	Number of order/quotes on this level
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_0	official time of book entry

Table 18: *MDUpdateAction "New"*

[21] A *MDUpdateAction* (279) = 0 ("New") is also disseminated whenever the quantity changes for the implied price (empty price level).

[22] This is not the case if the *MDUpdateAction* (279) = 0 ("New") is sent for the implied price (with empty price level).

9.3.4 Change of a price level

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

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

Note: *MDUpdateAction*="Change" is sent only for depth ≥ 1 when the price does not change. A *MDUpdateAction* (279) "Change" contains a price which can be used as a consistency check. However, it never contains a price that is different from the existing one on the current price level.

Example: Quantity changed to 8 for limit order above:

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1069	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	1	Change
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**58.22**	Price
271	> MDEntrySize	**8**	Quantity
346	> NumberOfOrders	1	Number of order/quotes on this level
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_1	official time of book entry

Table 19: *MDUpdateAction "Change"*

9.3.5 Overlay

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

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

Note: *MDUpdateAction*="Overlay" is sent only for depth ≥ 1, i.e. the field *MDPriceLevel (1023)* must be present. In contrast to the *MDUpdateAction*="Change" this instruction contains a price change.

Example: Buy limit order replaces the best buy limit order during instrument state "Auction":

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	205	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	70	Product
268	NoMDEntries	1	
279	> MDUpdateAction	5	
269	> MDEntryType	0	Bid
48	> SecurityID	63743	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**2.48**	Price
271	> MDEntrySize	N/A	Quantity remains the same in this example
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_5	official time of book entry

Table 20: *MDUpdateAction "Overlay"*

9.3.6 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 a consistency check.

Example: Deletion of limit order modified above:

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1070	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	2	Delete
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**58.22**	Price
271	> MDEntrySize	N/A	Quantity not populated for "Delete" !
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_2	official time of book entry

Table 21: *MDUpdateAction "Delete"*

9.3.7 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 $\geq$ specified price level.

Note: All price levels from the specified one and up to the maximum need to be deleted.
Example: Deletion of all orders for SecurityID = 8852, MarketSegmentID = 89 from level 3 and above:

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1068	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	4	Delete From
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Identifier assigned to each instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**58.19**	Price
271	> MDEntrySize	N/A	Quantity not populated !
1023	> MDPriceLevel	**3**	Book level
273	> MDEntryTime	t_3	official time of book entry

Table 22: *MDUpdateAction "Delete From"*

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

A *depth incremental* message with *MDUpdateAction (279)* = 3 ("Delete Thru") 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.
Example: Deletion of all price levels from 1 to price level 3.

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1068	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	3	Delete Thru
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Unique identifier assigned to each instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**58.22**	Price on level 3
271	> MDEntrySize	10	Quantity
346	> NumberOfOrders	1	Number of order/quotes on this level
1023	> MDPriceLevel	**3**	Book level
273	> MDEntryTime	t_4	official time of book entry

Table 23: *MDUpdateAction "Delete Thru"*

9.4 TES Trades

In addition to on-exchange trades Eurex EMDI reports ratified disclosed TES trades. For TES trades the *MDOriginType (1024)* is set to 1 = Off-Book. For all other trades the *MDOriginType (1024)* is set to 0 = Book.
An entry consists of

1. *MDOriginType (1024)* is set to 1 = Off-Book.
2. *TradeCondition (277)* field is set to U = Exchange last.
3. *MDEntrySize (271)* and *MDEntryPx (270)* is filled with quantity and price of the trade.
4. *MDEntryID (278)* and the *MDEntryTime (273)*.
5. *TrdType (828)* is filled with 1 = Off-Book BlockTrade, 2 = Exchange for Physical (EFP), 12 = Exchange for Swap (EFS), 55 = Exchange Basis Facility, 1000 = Vola Trade, 1001 = EFP-Fin Trade, 1002 = EFP-Index-Futures Trade or 1004 = Block Trade at Market

or

1. *MDOriginType (1024)* is set to 1 = Off-Book.
2. *TradeCondition (277)* field is set to a = Volume Only.
3. *MDEntrySize (271)* is filled with quantity of the changed volume.
4. *MDEntryID (278)* and the *MDEntryTime (273)*.
5. *TrdType (828)* is filled with 1 = Off-Book BlockTrade

When the *TESTradSesStatus (25044)* switches to 5 = PreClose the total *NonDisclosedTradeVolume (28873)* is published. An entry consits of

1. *MDOriginType (1024)* is set to 1 = Off-Book.
2. *MDEntryType (269)* is set to B = Trade Volume
3. *MDEntrySize (271)* is filled with the total quantity of the disclosed TES trades.
4. *NonDisclosedTradeVolume (28873)* is filled with the total quantity of the nonDisclosed TES trades.

9.5 Manual Trade Entry and Trade Reversal (Eurex EMDI)

The Eurex EMDI reports all on-exchange trades executed on Eurex Exchange's T7. In addition to order book trades, members receive trade messages for trades or trade reversals manually entered by Market Supervision.

The following fields are not sent for trade entries and trade reversals: *AggressorSide (2446)*, *AggressorTime (2445)*, *RequestTime (5979)*, *NumberOfBuyOrders (2449)*, *NumberOfSellOrders (2450)*.

The Eurex MDI does not report manual trade entries nor trade reversals as only statistical information is provided.

9.5.1 Manual Trade Entry (by Market Supervision) (Eurex EMDI)

The entry consists of

1. *TradeCondition (277)* field are always set to "Out Of Sequence" (=k).
2. *MDEntryType (269)* field is always set to "Trade" (=2).
3. *MDEntrySize (271)* and *MDEntryPx (270)* is filled with quantity and price of the trade.
4. *MDEntryID (278)* and the *MDEntryTime (273)*.

A manually entered trade will not affect the price statistics. Even when the manually entered trade is higher than the daily high price, it does not change the daily high price. For that reason the field *TradeCondition (277)* for a manually entered trade must only contain the "Out Of Sequence" attribute.

9.5.2 Trade Reversal (by Market Supervision) (Eurex EMDI)

A trade reversal is triggered by Market Supervision in order to delete a trade completely. A trade can only be reversed with its complete quantity.

Deleting a trade may affect the Trade Volume Report. Sometimes one or more price statistics are adjusted. An incremental for a trade reversal consists of one entry with *MDUpdateAction* = 2 ("Delete") and potentially one or more entries with *MDUpdateAction* = 1 ("Change") per involved instrument.

The entry consists of

1. *MDEntrySize (271)* and *MDEntryPx (270)* of the reversed trade.
2. *MDEntryType (269)* is set to Trade (= 2).
3. *MDEntryID (278)* (match event identifier) of the reversed trade.
4. *MDEntryTime (273)* of when the trade reversal request was processed.

The incremental entry with the *MDUpdateAction* = 1 ("Change") provides information about what was affected by the reversal. The entry consists of

- *MDEntrySize (271)* and *MDEntryPx (270)* if a new last price is set or *MDEntryPx* only if a other price statistic is affected (High, Low, Opening, Closing). If no price statistic is effected, *MDEntrySize* and *MDEntryPx* are empty.
- *TradeCondition (277)*, if *MDEntryPx* is not empty.
- *MDEntryType (269)* is set to Trade (=2).
- *MDEntryTime* (273) of the updated last trade if *TradeCondition* (277) contains "Exchange Last".

9.6 Trade Volume Reporting (Eurex EMDI)

All on-exchange trades executed on Eurex Exchange's 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).

The Eurex EMDI disseminates information about book and Off-Book trades. The *MDOriginType (1024)* is set to Book or Off-Book accordingly.

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 and origin type for each business day. A synthetic match can result in more than one trade volume record with the same *MDEntryID (278)* as shown in use case 3, section 9.6.3 and use case 4, section 9.6.4. The trade volume record for the future leg of a volatility option strategy is reported without *MDEntryID (278)*.

Every match step occurring in the exchange has an identifier in Eurex 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 Eurex ETI with the market data incremental feed of the Eurex 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 (see paragraph 9.8.4, Market data feed restart (Eurex EMDI)). 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 traded size on simple instruments not involving any simple instrument orders (e.g. direct match of complex instruments) is published via an additional *depth incremental* message having *Trade Condition (277)* set to 'a' (Volume only).

The following 4 use cases illustrate the *MDEntryID (278)* and how Trade Volume Reporting works. Please note that, the leg ratio is assumed always 1 for the following uses cases.

9.6.1 Use case 1: Direct match of simple instruments

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

Incoming buy order, 150@2885, FESX Sep

Existing Order book:

Bid	Ask
	50@2884 FESX Sep
	100@2885 FESX Sep

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.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell
FESX Sep	1	NEW	50@2884	U,R,AX,AY	BUY	1	1
FESX Sep	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"

9.6.2 Use case 2: Direct match of complex instruments

An incoming complex order [23] involving 2 legs is matching against the opposite side of the complex instrument order book.

Incoming buy order, 100@8, FESX Sep/Dec

Existing Order book:

Bid	Ask
	50@7 Sep/Dec
	50@8 Sep/Dec

Trade Volume Reporting: The entire strategy trade is reported followed by volume only information for each individual instrument leg for which there was no order in the book.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell
Sep/Dec	3	NEW	50@7	U,R,AX,AY	BUY	1	1
Sep/Dec	4	NEW	50@8	U,AX	BUY	1	1
Sep	3	NEW	50@-	a			
Sep	4	NEW	50@-	a			
Dec	3	NEW	50@-	a			
Dec	4	NEW	50@-	a			

with:
a = "Volume only"

[23] In general, a complex order is any combination of simple instruments. This example is based on a Future Time Spread. However, it is also valid for non-standard options strategy, e.g. 7 C BMW DEC12 54 Buy, 2 P BMW DEC12 53 Buy, 5 C BMW DEC12 52 Buy.

9.6.3 Use case 3: Complex versus simple order match

A buy spread order as an incoming complex order (Time Spread) matches (synthetically) against several simple instrument leg orders (outright orders).

Incoming buy order, 200@8.0 FESX Sep/Dec

Existing Order book:

Bid	Ask
120@2878 Dec	
30@2878 Dec	

Bid	Ask
	60@2886.0 Sep
	50@2886.0 Sep
	40@2886.0 Sep

This results in the following implied price:

Bid	Ask
	Sep12/Dec12 150@8.0

Trade Volume Reporting: The incoming spread order matches against the implied-in order of the order book which is a composition of all 5 outright orders in the order book. Again, the trades are aggregated per price level. The fields *NumberOfBuyOrders (2449)* and *NumberOfSellOrders (2450)* show how many orders are involved.

In case of a synthetically matched complex order either the buy or sell side contains an empty value. In case of a direct matched complex instrument, both sides are filled.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell
Sep/Dec	5	NEW	150@8	U	BUY	1	
Sep	5	NEW	150@2886	U,AX			3
Dec	5	NEW	150@2878	U,R,AX,AY		2	

9.6.4 Use case 4: Complex versus simple/complex match

Incoming buy order, 250@8, FESX Sep/Dec

Existing Order book:

Bid	Ask
120@2878 Dec 30@2878 Dec	

Bid	Ask
	150@2886 Sep

Bid	Ask
	100@8 Sep/Dec

Trade Volume Reporting: Incoming complex order is matching directly against the opposite side of a complex order; another part is matching against an implied-in order which was created by two existing outright orders for the Sep and Dec contracts. The direct match of the complex orders can be identified by existing entries for *NumberOfBuyOrders (2449)*, *NumberOfSellOrders (2450)*. The synthetic match can be identified by the missing entry for *NumberOfSellOrders (2450)*.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell
Sep/Dec	6	NEW	100@8	U	BUY	1	1
Sep/Dec	6	NEW	150@8	U	BUY	1	
Sep	6	NEW	150@2886	U			1
Dec	6	NEW	150@2878	U,AY		2	
Sep	6	NEW	100@-	a			
Dec	6	NEW	100@-	a			

9.6.5 Use case 5: Opening auction

After the uncrossing of the order book in a simple 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 Sep	60@24.39 Sep
25@24.39 Sep	57@24.39 Sep
20@24.39 Sep	18@24.39 Sep
55@24.39 Sep	
5@24.39 Sep	

Trade Volume Reporting: All orders are matching on the same price level. Therefore they are reported only once but with different *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:

Instr.	MD-EntryID	MDUpdate-Action	size@prc	TradeCond.	TrdType	AggrSide	#Buy	#Sell
Sep	1	NEW	135@24.39	U,R,AX,AY,AW	OPENING		5	3

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

Instr.	MDUpdateAction	size@prc	TradeCondition	TrdType
Sep	NEW	135@24.39	U,R,AX,AY	
Sep	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.

9.6.6 Use case 6: Self match prevention (order is totally canceled)

An incoming order is canceled due to self match prevention.

Incoming buy order, 150@2885, FESX Mar, MatchInstCrossID=1, member A

Existing Order book:

Bid	Ask
	50@2884 FESX Mar (MatchInstCrossID=1, Member A)

Trade Volume Reporting: A trade is reported: 0@2884, AggressorSide BUY. MDEntryID, TradeCondition, number of Buy and number of Sell orders are not filled. The resting canceled quantity is 50. The incoming canceled quantity (150) is not reported.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell	#RestingCxlQty
FESX Mar		NEW	0@2884		BUY			50

9.6.7 Use case 7: Self match prevention (order is partially canceled)

An incoming order is partially canceled due to self match prevention.

Incoming buy order, 150@2885, FESX Mar, MatchInstCrossID=1, Member A

Existing Order book:

Bid	Ask
	20@2884 FESX Mar (MatchInstCrossID=1, Member A)
	30@2884 FESX Mar (MatchInstCrossID=0)

Trade Volume Reporting: A trade is reported: 30@2884, AggressorSide BUY. The resting canceled quantity is 20.The incoming canceled quantity (120) is not reported.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell	#RestingCxlQty
FESX Mar	1	NEW	30@2884	U	BUY	1	1	20

9.7 Trade Volume Reporting (Eurex MDI)

The Eurex 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)*, *NumberOfSellOrders (2450)* are not provided.

For each simple instrument participating in a trade, Eurex MDI reports the total traded volume even when there are no simple instrument orders involved in the trade (e.g. direct match of complex instruments).

9.8 Failure of the market data feed/ matching engine

The following chapters explain fail-over scenarios and how receiving applications need to process them.

9.8.1 Normal processing

At start-up, the system assigns a unique sender identifier, the *SenderCompID (49)* to each market data feed. Afterwards the *SenderCompID (49)* remains constant for a given product during the entire business day. The *SenderCompID (49)* as shown in section 7.1 is available in the *packet header* and in the data message[24], e.g. *depth incremental* or *depth snapshot* itself.

For each incremental and snapshot message sent by market and reference data feeds:

- the field content for *SenderCompID (49)* in the *packet header* and in each data message is always the same.

For each incremental and snapshot message sent by the market data feeds:

- the *PacketSeqNum's* in the *packet header* are contiguous per *SenderCompID*, multicast address and port combination.
- the *MsgSeqNum's* in the data message are contiguous per product on the incremental feed of the Eurex EMDI.
- the *MsgSeqNum's* in the data message are contiguous per product on the market data feed of the Eurex MDI[25].

Figure 15 provides an example for constant *SenderCompID's* and increasing sequence numbers:

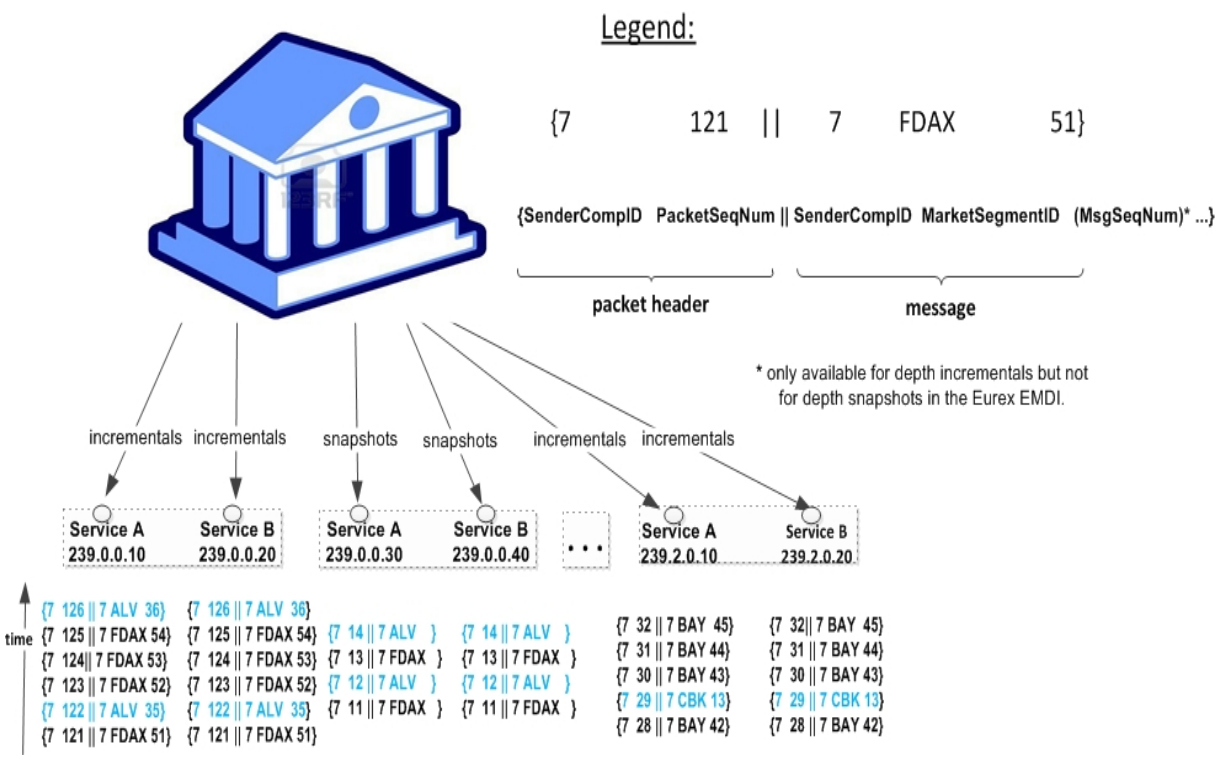


Figure 15: *Normal processing of sequence numbers for the Eurex EMDI*

[24] the content is the same.
[25] because the Eurex MDI delivers incrementals and snapshots on the same channel.

9.8.2 Market data feed fail-over (Eurex EMDI)

A new *SenderCompID*, available in the *packet header* and in each data message for incrementals and snapshots, indicates a fail-over of the market data feed.

Incrementals:

- the *PacketSeqNum's* in the *packet header* are reset to 1 and are contiguous per *SenderCompID (80)*, multicast address and port combination.
- the *MsgSeqNum's* in the data message remain contiguous per product.

Snapshots:

- the *PacketSeqNum's* in the *packet header* are reset to 1 and are contiguous per *SenderCompID* (80), multicast address and port combination.

Figure 16 illustrates the different behaviour for incremental and snapshot messages:

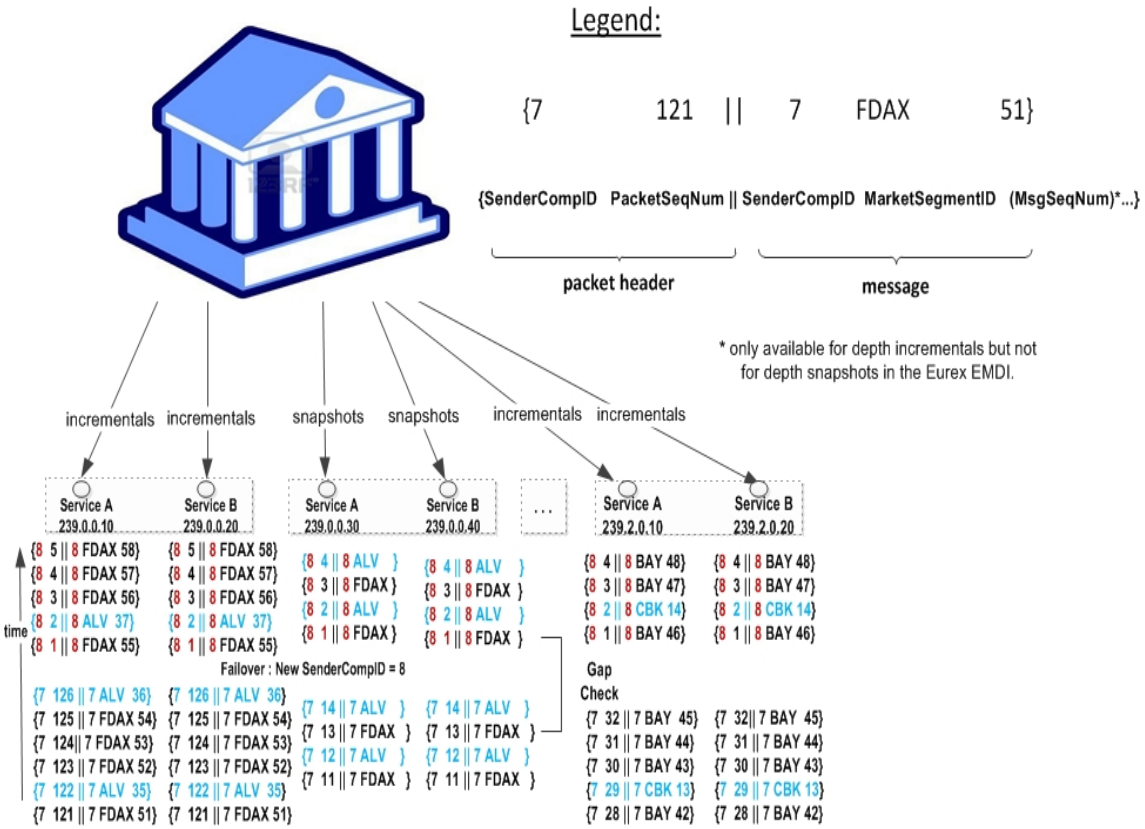


Figure 16: *Data feed fail-over for Eurex EMDI*

9.8.3 Market data feed fail-over (Eurex MDI)

Receiving applications are able to identify a failure as follows:

- the *PacketSeqNum's* in the *packet header* are reset to 1 and are contiguous per *SenderCompID* (80), multicast address and port combination.
- by a change of the *SenderCompID (80)* in the *packet header* and in all subsequent messages.
- by a reset of the *MsgSegNum's* for effected products to 1.

The snapshots are sent for all instruments before the incrementals are generated.
Figure 17 illustrates the different behaviour for incremental and snapshot messages:

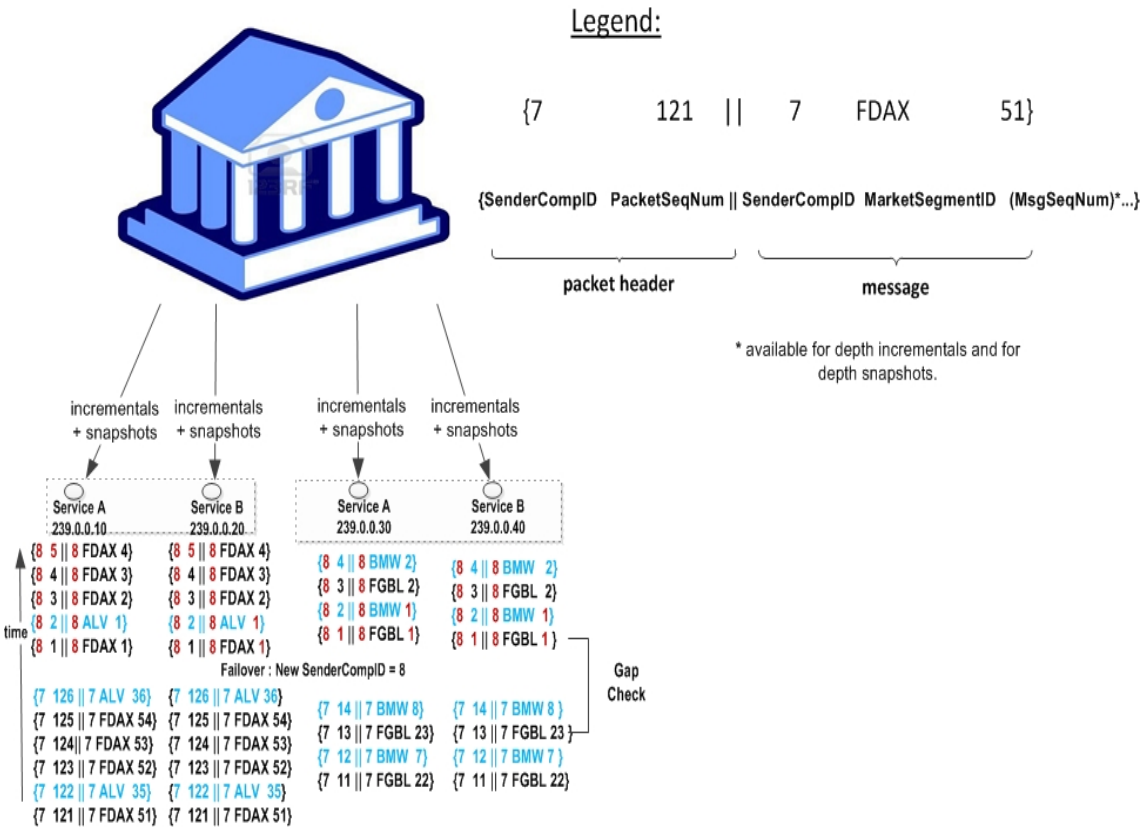


Figure 17: *Data feed fail-over for Eurex MDI*

Participants can identify this failover scenario by decoding the *packet header* of UDP datagram and comparing the *SenderCompID* value with the previous value.

9.8.4 Market data feed restart (Eurex EMDI)

A new *SenderCompID*, available in the *packet header* and in each data message for incrementals and snapshots, indicates a failure.

Incrementals:

- the *PacketSeqNum's* in the *packet header* are reset to 1 and are contiguous per *SenderCompID*, multicast address and port combination.
- the *MsgSeqNum's* in the data message is reset to 1 and are contiguous per product for incrementals. There is also a *Depth Incremental* message sent on the incremental feed that contains a full refresh of the Trade Statistics equivalent to the Trade Statistics that are also sent on the snapshot feed after the restart. Therefore member applications do not need to listen to the snapshot feed for synchronizing incrementals and snapshots. Please note that these Trade Statistics do not contain any *MDEntryID (278)* as for a regular trade event. See paragraph 9.6, Trade Volume Reporting (Eurex EMDI).

Snapshots:

- the *PacketSeqNum's* in the *packet header* are reset to 1 and are contiguous per *SenderCompID*, multicast address and port combination.

Once this condition is observed it is safe to assume that a fail-over scenario took place and the only correct action is to rebuild the order book. The receiving application needs 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 *depth snapshots* or from *depth incremental* messages, as described in section 6.4.1, Build the initial order book with the Eurex EMDI.

9.8.5 Market data feed restart (Eurex MDI)

Receiving applications are able to identify a failure as follows:

- by a change of the *SenderCompID (80)* in the *packet header* and in all subsequent messages.
- by a reset of the *MsgSegNum's* for all products to 1.

The *snapshots* are sent for all instruments before the incrementals are generated.

Once this condition is observed it is safe to assume that a fail-over scenario took place and the only correct action is to rebuild the order book. The receiving application needs 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 *depth snapshots* or from *depth incremental* messages, as described in section 6.4.2, Build the initial order book with the Eurex MDI.

9.8.6 Failure of the matching engine

All non-persistent orders and quotes are deleted. Participants can see a product state change as a result of the market reset. No special processing is necessary for market data applications.

In addition, participants receive a market reset event from their ETI-interface. The *service availability* message indicates the unavailability of the matcher by the ETI-field *MatchingEngineStatus (25005)*.

9.9 Trading states for a sample business day

Section 4.2, Trading states introduced the trading state information. This section describes a typical day with Eurex Exchange's T7. The example refers to the FDAX future on an expiry day. The times for each trading phase are valid for FDAX.

Participants should not rely on any specific order or sequence of messages as described in the following chapters. For instance, the system could send an *instrument state change* message instead of a *mass instrument state change* message resulting in the same trading state at the participants' side.

Unless participants rely on the message-specific fields (TradingSessionID (336), SessionSubID (625) and TradSesStatus (340)), the product state change messages don't have to be processed in order to receive the correct order book state; it is sufficient to process the *instrument state* and *mass instrument state* change messages.

9.9.1 Start-Of-Day

The system startup occurs in the morning. Note that with Eurex Exchange's T7, business days are not technically linked to the local calendar. Under normal circumstances a business date is equal to the local calendar date. Nevertheless it is possible that the system startup and with it the new business day starts before midnight on the previous calendar day.

At startup, the FDAX product goes into the product state "Start-of-Day", while all its instruments are in the state Closed. Traders have no access to the order book.

The system sends a *product state change* message (FIX *TradingSessionStatus* (MsgType = h)) with the field *TradingSessionID (336)* set to 3 = Morning and the field *TradingSessionSubID (625)* set to 7 = Quiescent. This indicates the product state "Start-of-Day".

The system furthermore sends *mass instrument state change* message (FIX *SecurityMassStatus* (MsgType = CO)) with the field *SecurityMassTradingStatus (1679)* containing 200 = Closed, which indicates that all instruments are in the state Closed. This message is sent once for the futures contracts (specified in the field *InstrumentScopeProductComplex (1544)* containing 1 = Simple Instrument) and once for futures spreads (specified in the field *InstrumentScopeProductComplex (1544)* containing 5 = Futures Spread) which is the only complex instrument type supported for futures.

The reference data feed begins with the system startup. Instruments that are scheduled to expire during the day are included in the reference data, but instruments that have already expired on a previous business day are no longer included in the reference data.

9.9.2 Pre-Trading

At **7:30 CET**, the FDAX product goes into the product state Pre-Trading while all its instruments change their instrument state to Book. Traders are now able to perform full order and quote maintenance.

The system sends a *product state change* message with the field *TradingSessionID (336)* set to 3 = Morning and the field *TradingSessionSubID (625)* set to 1 = Pre-Trading. This indicates the product state Pre-Trading.

The system furthermore sends *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 202 = Book, which says that all instruments are in the state Book. This message is sent once for simple instruments and once for futures spreads.

9.9.3 Opening Auction

At **7:50 CET**, the FDAX product goes into the product state Trading. At the same time, all its simple instruments (futures contracts) change their instrument state to Opening Auction. The complex instruments (futures spreads) remain in the instrument state Book. Traders can do full order and quote maintenance. For the simple instruments, the system publishes the best bid and ask prices if the order book is not crossed, or an indicative auction price if the order book is crossed.

The system sends a *product state change* message with the field *TradingSessionID (336)* set to 1 = Day and the field *TradingSessionSubID (625)* set to 3 = Trading. This indicates the product state Trading.

The system furthermore sends one *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 204 = Opening Auction, which says that all instruments are in the

state opening auction. This message is sent only for simple instruments. There is no message sent for futures spreads as they do not change their state.

9.9.4 Continuous Trading

At **8:00 CET**, the opening auction period of the FDAX product ends and continuous trading starts. There is no product state change involved, but all the instruments transition to the instrument state Continuous. The change of the instrument state implies an auction trade if the order book was crossed. This applies also to the complex instruments (futures spreads), even though they had no formal auction call phase before.

In the instrument state Continuous, traders can maintain their orders and quotes. Incoming orders and quotes are continuously matched. The system publishes order book and trade data.

The system sends two *mass instrument state change* messages with the field *SecurityMassTradingStatus (1679)* containing 203 = Continuous, which means that all instruments are in the state Continuous. This message is sent once for simple instruments and once for futures spreads.

9.9.5 Intraday Expiry

At **13:00 CET**, the front month contract of the FDAX future expires on an expiration day. The affected simple instrument goes to the instrument state Restricted. The same happens to all complex instruments (futures spreads) that have the affected simple instrument as a leg. For these instruments, all quotes are deleted automatically. Traders may delete their orders but not modify them or add new orders.

For the expired simple instrument, the system sends a *instrument state change* message (FIX *SecurityStatus* (MsgType = f)) with the field *SecurityTradingStatus (326)* containing 201 = Restricted, which says that this particular instrument is in the state Restricted. Furthermore, the field *SecurityStatus (965)* contains the value 4 = Expired.

For each impacted complex instrument, the system sends a *instrument state change* message with the field *SecurityTradingStatus (326)* containing 201 = Restricted. However, the field *SecurityStatus (965)* still contains the value 1 = Active.

9.9.6 Closing Auction

At **22:00 CET**, the FDAX product is set into the product state Closing. At the same time, all its simple instruments (futures contracts) change their instrument state to Closing Auction. The complex instruments (futures spreads) change to the instrument state Book. Traders can do full order and quote maintenance. For simple instruments, the system publishes the best bid and ask prices if the orderbook is not crossed, or an indicative auction price if the order book is crossed.

The expired front month contract and the related futures spread instruments are not affected. They remain in the state Restricted.

The system sends a *product state change* message with the field *TradingSessionID (336)* set to 1 = Day and the field TradingSessionSubID (625) set to 4 = Closing. This indicates the product state Closing.

For simple instruments, the system sends one *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 210 = Closing Auction. The message carries an exception list which contains the expired instrument as the only list item. For this instrument, the list item field *SecurityTradingStatus (326)* contains 201 = Restricted.

For the futures spreads, the system sends one *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 202 = Book. The message carries an exception list which contains all the futures spreads that are in the state Restricted. For these instruments, the list item field *SecurityTradingStatus (326)* contains 201 = Restricted.

9.9.7 Post-Trading

At **22:03 CET**, the closing auction period of the FDAX product ends. The product FDAX goes into the product state Post-Trading. The simple instruments that have been in the instrument state Closing Auction now change to the state Book. The other instruments do not change their state.

The expired front month contract and the related futures spread instruments are not affected. They remain in the state Restricted.

For the instruments that are in the instrument state Book, traders can do full order and quote maintenance.

The system sends a *product state change* message with the field *TradingSessionID (336)* set to 5 = Evening and the field *TradingSessionSubID (625)* set to 5 = Post-Trading. This indicates the product state: Post-Trading.

For simple instruments, the system sends one *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 202 = Book. The message carries an exception list which contains the expired instrument as the only list item. For this instrument, the list item field *SecurityTradingStatus (326)* contains 201 = Restricted.

For the futures spreads, the system sends one *mass instrument state change* message with the field *SecurityMassTradingStatus (1679)* containing 202 = Book. The message carries an exception list which contains all the futures spreads that are in the state Restricted. For these instruments, the list item field *SecurityTradingStatus (326)* contains 201 = Restricted.

9.9.8 End-Of-Day

After **22:30 CET**, the FDAX product goes into the product state End-Of-Day. All its instruments change into the instrument state Closed. Traders can no longer access the order book. The exchange system will start the end-of-day processing.

The system sends a *product state change* message with the field *TradingSessionID (336)* set to 5 = Evening and the field *TradingSessionSubID (625)* set to 7 = Quiescent. This indicates the product state End Of Day

The system also sends two *mass instrument state change* messages with the field *SecurityMassTradingStatus (1679)* containing 200 = Closed, which means that all instruments are in the state Closed. This message is sent once for simple instruments and once for future spreads.

10 Fine tuning client applications

This chapter covers some aspects of application tuning which should be considered during the design process of receiving applications.

10.1 Buffer size

Messages need to be buffered and sorted in order to deal with datagrams arriving in reversed order. A bigger buffer size usually slows down the processing of messages and should therefore be avoided. Conversely, receiving applications might falsely declare a message as lost if the buffer size is too small. As you can see from this example, a bigger buffer size works contrary to the speed of an application but reduces the chances of "lost" messages.

Another factor which effects the ideal buffer size is the ratio of joined multicast streams to available bandwidth of an Eurex Market Data connection. A connection which operates at high network utilization levels might more often cause multicast drops or packets arriving in an incorrect sequence.

Last but not least, the location of the receiving application also matters. For instance, an application running in co-location has very few out-of-order multicast packets (none in most cases) while an application which is located at a far distance from the Eurex host receives a few packets out-of-order.

Therefore a general recommendation concerning the buffer size cannot be made. Developers need to determine the ideal buffer size during internal testing. Please take into account that the message rate for the public broadcast is usually much lower in the simulation environment than it is in production.

10.2 Packet and message processing

It is important that messages are removed from the network in a timely fashion to prevent them from being dropped by the client machine due to "receive buffer" overflow in the IP stack. In addition to the removal of messages from the network stack (as might be performed in response to a select() operation, for example), this design requires a time-based component to determine when a missing packet is declared lost (as opposed to simply delayed).

The mechanism behind this is an implementation detail, and is platform-specific, but in its simplest form a timed select() and polling of an internal list of overdue packets would suffice.

The actual time out value applied is very implementation-specific, and may be either determined dynamically (with a knowledge of when the first overdue packet is declared lost) or a simple static value.

Note: *Depth incremental* messages must not be applied to the order book unless they are in sequence.

For each network packet received, decode it into the constituent FIX message and then for each message:

The market data feeds may contain information about multiple products. If it is not for a product that the clients application is interested in:

- Throw it away.

If the message is already in the cache:

- The clients application already received this message from the mirror channel, or it has been duplicated in the network.
- Throw it away.

Otherwise:

- Add it to the cache.

10.3 Application level

Various approaches can lead to faster processing on application level. The approaches depend primarily on the purpose and algorithm of the application.

10.3.1 Discarding duplicate packets witin the live-live environment

It is expected that receiving applications process packets from Service A and B simultaneously. The concept of the *packet header* allows receiving applications to detect duplicates based on the *PacketSeqNum*. It is recommended to discard a packet after decoding the *packet header* once it has been identified as duplicate. The actual message following the *packet header* no longer has to be decoded, allowing a faster processing speed.

10.3.2 Order book processing

Depth incremental messages deliver changes of the order book from ToB to worse price levels. Trading algorithms which are based on fast matching without the knowledge of the order book could process ToB only before making a decision and process the order book afterwards.

Conversely, trading algorithms with a matching logic based on the knowledge of the order book need to process the order book before sending orders/quotes.

10.3.3 Optimal processing of desired products (Eurex EMDI)

Receiving applications interested in certain products need to join a multicast address which contains the desired products according to the mapping table provided in the reference data. Packets may arrive from different partitions on the same multicast address as shown in figure 18. The PartitionID in the *packet header* for the Eurex EMDI can be used to identify packets arriving from partitions which carry the desired products. All other packets can be easily discarded without decoding the entire message.

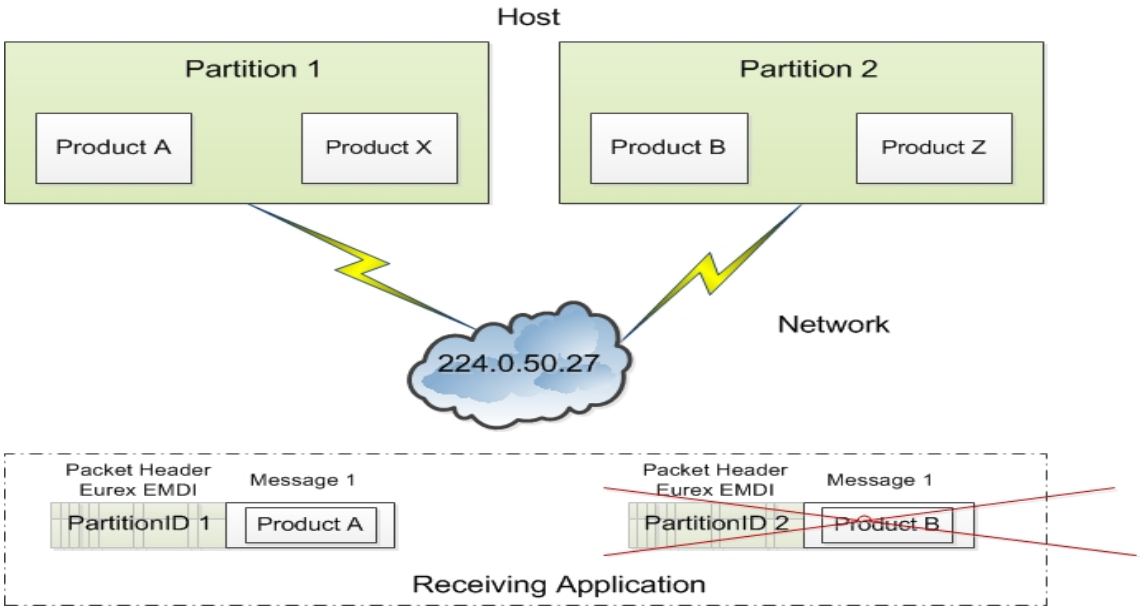


Figure 18: *Discarding packets with unwanted products based on the PartitionID of the Eurex EMDI packet header*

The example provided in figure 18 shows two products arriving on multicast address 224.0.50.27. The participant is only interested in product A. Packets containing product A or product X can be identified by the field *PartitionID* in the *packet header*. As product X is not one of the desired products it can be discarded after decoding the message.

Based on the reference data, the receiving application knows that packets coming from PartitionID 2 contain only undesired products. It discards all packets with *PartitionID* = 2 in the *packet header* without decoding message 1.

Part III
Reference

11 Detailed data feed description and layout

This chapter provides message layouts and field information. It is structured by service messages, data messages and data files.

Please consider, that the following tables will only list valid values for enum and set data types, which are used within that specfic context. The complete list and order of all valid values supported by a specific enum or set datatype could be found within Eurex Market + Reference Data Interfaces - XML FAST Templates. These files could be found on www.eurexchange.com > Technology > Eurex Exchange's T7 > System documentation. Specifically the actual wire values for Fast 1.1 decoders need to be derived from the XML Fast Templates.

11.1 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.

11.1.1 FAST reset message

The template with ID = 120 is not included in the "FAST Message Templates" file. This TID is reserved in the main FAST specification and allocated by the FAST Session Control Protocol specification (SCP 1.1[26])

Note: A conforming decoder must be able to deal with the FAST reset message even though it is not mentioned in the template file. Once the *FAST reset* message is sent out, the dictionary needs to be initialized.

11.1.2 Packet header (Eurex EMDI)

Delivered in: Every UDP-datagram

The *packet header* is a technical header used for identification of datagrams and is sent on a channel basis. Every partition stamps outgoing datagrams with a sequence number (field: *PacketSeqNum*).

One method to identify duplicates between Service A and B is by the use of the field *PacketSeqNum* which is unique per *senderCompID*; a faster way is to perform a memory comparison on the first 9 bytes of the *packet header*.

This method eliminates the need to even decode the header in order to determine, if it has already been processed. This is especially useful to applications using both Service A and Service B feeds, allowing them to determine that a packet has already been processed without incurring any decoding overhead at all.

[26] www.fixprotocol.org/fast > FAST Session Control Protocol.

As the *packet header* message is not defined in the FIX standard, the FIX Tags for *PacketSeqNumber*, *SendingTime* and *PerformanceIndicator* are not shown in the table below. The following layout is available after FAST decoding of the *packet header*:

Field Name	Data Type	Description
PartitionID	uInt32	Sending partition.
SenderCompID	uInt32	Unique id for a sender.
PacketSeqNumber	byte vector	Datagram sequence number.
SendingTime	byte vector	Time when market data feed handler writes packet on the wire.
PerformanceIndicator	byte vector	Current load of system. Time difference between the incoming ETI-order/quote and the time the market data is written to the socket. This information is provided for the incremental feed of Eurex EMDI only and is not provided for the Eurex MDI. This field should be interpreted as a signed 32-bit integer having a minimum value of 0x80000000 (in case of time synchronisation anomalies the value can be negative).

The following picture shows the structure of the *packet header* before FAST-decoding :

1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	4 Bytes	1 Byte	8 bytes	1 Byte	4 Bytes
PMAP	TID	Parti-tionID	Sender Comp ID	Length	PacketSeqNum	Length	SendingTime	Length	PerformanceIndicator (only for un-netted feed)
1	2	3	4	5	9	10	18	19	23

Figure 19: *Structure of the packet header for Eurex EMDI*

The last three fields are byte vectors with fixed length. Byte vectors are not stop bit encoded according to the FAST standard. Each of them are preceded by a FAST encoded 1 Byte length field as per the FAST specification for byte vector fields.

Note: The field *PerformanceIndicator* including the length field is only available in messages on the Eurex EMDI incremental feed. The *PartitionID* is available in messages on both incremental and snapshot feed of the Eurex EMDI.

11.1.3 Packet header (Eurex MDI /Eurex RDI)

Delivered in: every UDP-datagram

The *packet header* of Eurex MDI and Eurex RDI doesn't contain the *PerformanceIndicator* with length field and the *PartitionID*. The rest of the *packet header* is identical to Eurex EMDI. Duplicates between Service A and Service B can be detected by a memory comparison on the first 8 bytes of the *packet header*.

Field Name	Data Type	Description
SenderCompID	uInt32	Unique id for a sender
PacketSeqNumber	byte vector	Datagram sequence number
SendingTime	byte vector	Time when market data feed handler writes packet on the wire.

Wire representation:

1 Byte	1 Byte	1 Byte	1 Byte	4 Bytes	1 Byte	8 bytes
PMAP	TID	Sender Comp ID	Length	PacketSeqNum	Length	SendingTime
1	2	3	4	8	9	17

Figure 20: *Structure of the packet header for Eurex MDI and Eurex RDI*

11.1.4 Functional beacon message

Delivered on: Eurex EMDI incremental and Eurex RDI incremental

The *functional beacon* message is sent as a "line active" indicator whenever there are no messages generated on the EMDI incremental feed for the respective product within the last 10 seconds in production. On the RDI incremental feed it is sent every two minutes whenever there are no messages generated.

Functional beacons are sent once the market data service becomes available. If no messages have been sent on the incremental feed for a product (or market for RDI) then *LastMsgSeqNumProcessed (369)* is set to zero.

US-customers receive a *functional beacon* on the EMDI incremental for US-tradable products only.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: 0 – Beacon
49	SenderCompID	Y	uInt32	Unique id of a sender.
50	SenderSubID	Y	uInt32	Product Identifier, e.g. 89, for EMDI or Market Identifier, e.g. 1 (EUREX), for RDI.
369	LastMsgSeqNum-Processed	Y	uInt32	Last sequence number on the incremental feed for this SenderSubID.

11.1.5 Technical heartbeat message

Delivered on: every channel for Eurex EMDI, Eurex MDI and Eurex RDI

The *technical heartbeat* (also called *technical beacon*) message is sent out periodically on every multicast address and consists of a *FAST reset message* (TID=120) only. The sole purpose of the *technical heartbeat* message is to keep routing trees alive, i.e. this message prevents routers from dropping multicast packages.

11.1.6 Market data report message

Delivered on: Eurex RDI snapshot feed

The number of reference data could be large, and it might take some time to complete one snapshot cycle. In order to mark the beginning of the reference data messages, the market data report message is sent with a start- and end-flag indicated by the field *MDReportEvent (2535)*. The fields *MDReportCount (2536)* and *LastMsgSeqNumProcessed (369)* allow to determine the number of snapshots and incrementals within the reference data snapshot feed. For further details see section 9.2.2, Counters as part of the market data report message, on page 43.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: DR – Market Data Report
2536	MDReportCount	N	uInt32	Number of reference data snapshot messages in the snapshot cycle of the current business day. Only sent for MDReportEvent = 1, i.e. at the start of the cycle.
1301	MarketID	Y	string	Market Identifier Code as specified in ISO 10383.
50	SenderSubID	Y	uInt32	Product Identifier, e.g. 89, for EMDI or Market Identifier, e.g. 1 (EUREX), for RDI.
369	LastMsgSeqNumProcessed	N	uInt32	*MDReportCount* value of the last reference data message (snapshot or incremental) at the end of the snapshot cycle (products and instruments share a single sequence). Number of incremental updates in snapshot cycle can be calculated as *LastSeqNumProcessed - MDReportCount*. Only sent for MDReportEvent = 1, i.e. at the start of the cycle.
2535	MDReportEvent	Y	MDReportEvent (enum)	Defines start/end of reference data. Other values have been added due to exchange wide FAST template alignments. **Value** / **Description**: 1 – Start of reference data; 2 – End of reference data
60	TransactTime	Y	timestamp	Creation time of the snapshot.
2537	TotNoMarketSegmentReports	N	uInt32	Contains the number of product level messages sent in the snapshot. Only sent for MDReportEvent = 1.
2538	TotNoInstrumentReports	N	uInt32	Contains the number of instrument level messages sent in the snapshot. Only sent for MDReportEvent = 1.

11.2 Reference data messages

Reference data is delivered via the snapshot and incremental feeds with the Eurex RDI. A *product incremental* message does not exist in this release.

11.2.1 Product snapshot message

Delivered on: Eurex RDI snapshot feed

The *product snapshot* message sends the collapsed view of the market, market segment and product. It provides the dynamic multicast addresses/ports for each type (field: *MDFeedType* (1022)) of the market data feed.

The maximum order book depth is specified by the field *MarketDepth* (264).

One message per product is sent.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	Reference data snapshot (one per product). **Value** / **Description**: BU – Market Definition
34	MsgSeqNum	Y	uInt32	Contiguous across all messages on the reference data snapshot feed regardless of message type. See figure 9 chapter 9.2.1, General structure of the snapshot cycle.
1301	MarketID	Y	string	Market Identifier Code as specified in ISO 10383.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. "89". Can also be a product pool id.
2400	EffectiveBusinessDate	Y	uInt32	Current business date, e.g. "20131128".
28871	NextEffectiveBusiness-Date	Y	uInt32	Next business date.
7703	MarketSegment	Y	string	Product name, e.g. "FDAX".
15	Currency	Y	string	Currency as published in ISO 4217
2542	MarketSegmentStatus	Y	MDStatus (enum)	Defines if on-exchange trading on Eurex Exchange's T7 is available for a product (=Active) or not (=Published). Reason for a published product can be for the following reasons: a) The product is still traded on the current "Eurex platform prior to migration, b) First trading day of the new product has not yet been reached, or c) the product is setup on the current "Eurex platform for Off-Book trading with no intention for on-exchange trading. **Value** / **Description**: 1 – Active; 10 – Published
5948	PartitionID	Y	uInt32	Partition of the product.
<DerivativesDescriptor> (optional) group starts				
1396	MarketSegmentDesc	Y	string	Product Description, e.g. "OPT ON THE DAX INDEX".
7177	MarketSegmentSymbol	Y	string	Product ISIN, e.g. "DE0009653147".
1325	ParentMktSegmID	Y	string	Standard Eurex product types, e.g. "FINX". For Product Pools the default value "FIPS" will be provided.
9543	USFirmFlag	Y	USFirm-Flag (enum)	Flag to identify whether product may be traded from the USA. **Value** / **Description**: N – No; Y – Yes
2543	MarketSegmentType	N	Market-Segment-Type (enum)	**Value** / **Description**: 1 – Pool
2544	MarketSegmentSubType	N	Market-Segment-Sub-Type (enum)	**Value** / **Description**: 1 – Inter Product Spread
<DerivativesDescriptor> (optional) group ends				

Tag	Field Name	Req'd	Data Type	Description
<UnderlyingDescriptor> (optional) group starts				
30308	UnderlyingSecurityExchange	N	string	Market Identifier Code of the underlying as specified in ISO 10383.
30311	UnderlyingSymbol	N	string	Underlying security symbol, e.g. "DAX".
30309	UnderlyingSecurityID	N	string	ISIN code of the underlying, e.g. "DE0008469008".
30305	UnderlyingSecurityIDSource	N	string	**Value** / **Description**: 4 – ISIN
28843	UnderlyingPrevClosePx	N	decimal	Closing price of the underlying on the previous day.
<UnderlyingDescriptor> (optional) group ends				
<InstrumentScopes> sequence starts				
1656	NoInstrumentScopes	N	length	Number of instrument scope in the segment. Always 1.
1535	> InstrumentScopeOperator	Y	InstrumentScopeOperator (enum)	Instrument scope operator. **Value** / **Description**: 1 – Include
1547	> InstrumentScopeSecurityType	Y	InstrumentScopeSecurityType (enum)	Type of security. **Value** / **Description**: MLEG – Complex Instrument; VAR – Variance Futures; TRF – Total Return Futures
1548	> InstrumentScopeSecuritySubType	N	uInt32	Standard strategy type for complex instruments. The mapping of integer values to the corresponding strategy types is published on the Eurex website at www.eurexchange.com > Technology > Eurex Exchange's T7 > Documents
<InstrumentScopes> sequence ends				
<RelatedMarketSegments> sequence starts				
2545	NoRelatedMarketSegments	N	length	Number of market segments having a relationship with the current entry.
2546	> RelatedMarketSegmentID	Y	uInt32	Market segment identifier.
2547	> MarketSegmentsRelationship	Y	MarketSegmentRelationship (enum)	Defines the type of the relationship. **Value** / **Description**: 100 – Cash leg for volatility strategies.
<RelatedMarketSegments> sequence ends				
<BaseTradingRules> group starts				
<TickRules> sequence starts				
1205	NoTickRules	Y	length	Number of tick rules defining the price step table for the product. See Functional Reference Manual, section "Limit Price".
28887	> TickRuleID	N	uInt32	Table identifier, only for cash. Used in field RefTickTableID(1787) from the Instrument Snapshot message.
2571	> TickRuleProductComplex	N	ProductComplex (enum)	Defines the instrument type for the tick rule. **Value** / **Description**: 1 – Simple Instrument; 2 – Standard Option Strategy; 3 – Non-Standard Option Strategy; 4 – Volatility Strategy; 5 – Futures Spread; 6 – Inter-Product Spread; 7 – Standard Futures Strategy; 8 – Pack and Bundle; 9 – Strip
1206	> StartTickPriceRange	Y	decimal	Starting price range for specified tick increment (inclusive).
1207	> EndTickPriceRange	Y	decimal	Ending price range for the specified tick increment (non-inclusive).

Tag	Field Name	Req'd	Data Type	Description
1208	> TickIncrement	Y	decimal	Tick increment for stated price range. Specifies the valid price increments at which quotes or trades for an instrument of the product can be entered.
<TickRules> sequence ends				
<PriceRangeRules> sequence starts				
2550	NoPriceRangeRules	Y	length	Price Range Tables. See Functional Reference Manual, section "Price Range Table Basics".
2556	> PriceRangeRuleID	Y	uInt32	Table identifier.
2555	> PriceRangeProduct-Complex	N	Product-Complex (enum)	Not filled for cash. **Value** / **Description** 1 – Simple Instrument 2 – Standard Option Strategy 3 – Non-Standard Option Strategy 4 – Volatility Strategy 5 – Futures Spread 6 – Inter-Product Spread 7 – Standard Futures Strategy 8 – Pack and Bundle 9 – Strip
2551	> StartPriceRange	Y	decimal	Start of price range (inclusive).
2552	> EndPriceRange	Y	decimal	End of price range (non-inclusive).
2553	> PriceRangeValue	N	decimal	Maximum allowable quote spread (absolute value). Conditionally required if PriceRangePercentage is absent.
2554	> PriceRangePercentage	N	decimal	Maximum allowable quote spread (percentage value). Conditionally required if PriceRangeValue is absent.
<PriceRangeRules> sequence ends				
2559	QuoteSideIndicator	Y	QuoteSide-Indicator (enum)	Defines if one sided quotes are allowed. **Value** / **Description** 0 – One-sided quote not allowed 1 – One-sided quote allowed
2557	FastMarketPercentage	N	decimal	Percentage by which range resulting from *PriceRangeValue (2553)* and *PriceRangePercentage (2554)* has to be extended to obtain the valid price range during Fast Market: 0 = no change to normal range, e.g. 80->80 50 = increase by half, e.g. 80->120 100 = range to be doubled, e.g. 80->160 More details are available in the Functional Reference Manual.
<BaseTradingRules> group ends				
<BaseTradingRulesDerivatives> (optional) group starts				
<QuoteSizeRules> (optional) sequence starts				
2558	NoQuoteSizeRules	N	length	
647	> MinBidSize	N	uInt32	Minimum bid quantity (identical to minimum offer quantity).
648	> MinOfferSize	N	uInt32	Minimum offer quantity (identical to minimum bid quantity).
2447	> FastMarketIndicator	N	FastMarket-Indicator (enum)	Indicates if product is in the state Fast Market. **Value** / **Description** 0 – No 1 – Yes
<QuoteSizeRules> (optional) sequence ends				
<FlexRules> sequence starts				
2560	NoFlexProductEligibilities	N	length	

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>2561</td><td>> FlexProductEligibility-Complex</td><td>Y</td><td>Product-Complex (enum)</td><td>Defines the strategy types available for user defined strategies.<table><tr><th>Value</th><th>Description</th></tr><tr><td>2</td><td>Standard Option Strategy</td></tr><tr><td>3</td><td>Non-Standard Option Strategy</td></tr><tr><td>4</td><td>Volatility Strategy</td></tr><tr><td>5</td><td>Futures Spread</td></tr><tr><td>6</td><td>Inter-Product Spread</td></tr><tr><td>7</td><td>Standard Futures Strategy</td></tr><tr><td>8</td><td>Pack and Bundle</td></tr><tr><td>9</td><td>Strip</td></tr></table></td></tr>
<tr><td>1242</td><td>> FlexProductEligibilityIndicator</td><td>Y</td><td>Flex-Product-Eligibility-Indicator (enum)</td><td>Defines if user defined strategies are allowed.<table><tr><th>Value</th><th>Description</th></tr><tr><td>N</td><td>No</td></tr><tr><td>Y</td><td>Yes</td></tr></table></td></tr>
<tr><td colspan="5"><FlexRules> sequence ends</td></tr>
<tr><td colspan="5"><BaseTradingRulesDerivatives> (optional) group ends</td></tr>
<tr><td colspan="5"><BaseTradingRulesCash> (optional) group begins</td></tr>
<tr><td colspan="5"><AuctionTypeRuleGrp> sequence starts</td></tr>
<tr><td>2548</td><td>NoAuctionTypeRules</td><td>Y</td><td>length</td><td>Number of rules. Always 1.</td></tr>
<tr><td>1803</td><td>> AuctionType</td><td>Y</td><td>AuctionType (enum)</td><td>Defines the type of auction.<table><tr><th>Value</th><th>Description</th></tr><tr><td>100</td><td>Any Auction</td></tr></table></td></tr>
<tr><td>28874</td><td>> ClosedBookIndicator</td><td>Y</td><td>Closed-Book-Indicator (enum)</td><td>Indicates whether the order book is closed during auction trading.
N = No reserved for future releases.<table><tr><th>Value</th><th>Description</th></tr><tr><td>N</td><td>No</td></tr><tr><td>Y</td><td>Yes</td></tr></table></td></tr>
<tr><td>28875</td><td>> MarketImbalanceIndicator</td><td>Y</td><td>Market-Imbalance-Indicator (enum)</td><td>Controls if during auction call/volatility interruption/extended volatility interruption/market order interruption phase a surplus (side and volume) at the indicative price (if crossed order book) or the best bid/best ask limit and quantity (if uncrossed order book) is displayed to the market. Only makes sense when ClosedBookIndicator is Y = Yes.
Y = Yes reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>N</td><td>No</td></tr><tr><td>Y</td><td>Yes</td></tr></table></td></tr>
<tr><td colspan="5"><AuctionTypeRuleGrp> sequence ends</td></tr>
<tr><td>28878</td><td>SettlBusinessDays</td><td>N</td><td>uInt32</td><td>Indicates the number of business days from trade execution after which settlement is to be effected.</td></tr>
<tr><td colspan="5"><BaseTradingRulesCash> (optional) group ends</td></tr>
<tr><td colspan="5"><MatchRules> sequence starts</td></tr>
<tr><td>1235</td><td>NoMatchRules</td><td>Y</td><td>length</td><td>Used to convey allocation rules for matching</td></tr>
</table>

Tag	Field Name	Req'd	Data Type	Description
2569	> MatchRuleProduct-Complex	Y	Product-Complex (enum)	Indicates the instrument type. **Value** / **Description** 1 – Simple Instrument 2 – Standard Option Strategy 3 – Non-Standard Option Strategy 4 – Volatility Strategy 5 – Futures Spread 6 – Inter-Product Spread 7 – Standard Futures Strategy 8 – Pack and Bundle 9 – Strip
1142	> MatchAlgorithm	Y	Match-Algorithm (enum)	Defines the order allocation method. **Value** / **Description** PT – Price-time PR – Pro-rata TPR – Time pro-rata
574	> MatchType	N	MatchType (enum)	If *MatchType* is not filled the same matching algorithm is used for continuous trading and auction trades. AutoMatch stands for continuous trading and CallAuction for auction trades. **Value** / **Description** 4 – Auto Match 7 – Call Auction
<MatchRules> sequence ends				
<Feeds> sequence starts				
1141	NoMDFeedTypes	Y	length	Number of feeds.
1022	> MDFeedType	Y	MDFeed-Type (enum)	Type of feed. **Value** / **Description** HI – High Incremental (unnetted feed) HS – High Snapshot (unnetted feed) L – Low (netted feed)
1021	> MDBookType	Y	MDBook-Type (enum)	Type of book data. **Value** / **Description** 2 – Price Depth 3 – Order Depth
264	> MarketDepth	N	uInt32	Order book (price level) depth.
2563	> MarketDepthTimeInterval	N	uInt32	Netting interval for low bandwidth feeds (0=no netting).
2565	> MDRecoveryTimeInterval	N	uInt32	Recovery interval (duration of one cycle).
2567	> PrimaryServiceLocationID	Y	string	IP Address for Service A.
28591	> PrimaryServiceLocationSubID	Y	uInt32	Port number for IP address Service A.
2568	> SecondaryServiceLocationID	N	string	IP Address Service B.
28593	> SecondaryServiceLocationSubID	N	uInt32	Port number for IP address Service B.
<Feeds> sequence ends				

11.2.2 Instrument snapshot message

Delivered on: Eurex RDI snapshot feed

The *instrument snapshot* message is used to describe simple, complex and flexible instruments. For a complex instrument, the *LegSecurityID (602)* is the *SecurityID (48)* of the instrument that is part (leg) of the complex instrument.

A link to the reference information of the Eurex System is provided with the field *SecurityAltID (455)*.

One message per instrument is sent.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: d – Security Definition
34	MsgSeqNum	Y	uInt32	Contiguous across all messages on the reference data snapshot feed regardless of message type. See figure 9 chapter 9.2.1, General structure of the snapshot cycle.
48	SecurityID	Y	int64	The instrument identifier uniquely identifies an instrument, e.g. 72057800196358145.
22	SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
<SecurityAlt> (optional) sequence starts				
454	NoSecurityAltID	Y	length	Number of alternate identifiers.
455	> SecurityAltID	Y	string	Alternate instrument identifier.
456	> SecurityAltIDSource	Y	string	4 always present. B only for cash. M only for derivatives. **Value** / **Description**: 4 – ISIN; B – Wertpapier (WKN = Wertpapierkennnummer); M – Marketplace-assigned identifier
<SecurityAlt> (optional) sequence ends				
167	SecurityType	Y	Security-Type (enum)	Type of security. **Value** / **Description**: OPT – Option; FUT – Future; MLEG – Complex instrument; CS – Common stock; BOND – Bond
965	SecurityStatus	Y	MDStatus (enum)	Instruments may be published prior to becoming active for trading. Status is identical to the product level, i.e. all instruments of a product are either published or active with the exception of those that expire one or more business days after they stop trading. These will still be distributed until they expire but as inactive. Active instruments can also expire intra-day or be suspended which is conveyed with the same field but only within intraday state change messages (EMDI or MDI). **Value** / **Description**: 1 – Active; 2 – Inactive; 10 – Published

Tag	Field Name	Req'd	Data Type	Description
107	SecurityDesc	N	string	The format depends on the type of instrument: Standard options strategies: PROD.O.YYMMDD.IST.SEQ-NO Non-standard options strategies: PROD.N.YYMMDD.SEQ-NO Options volatility strategies: PROD.V.YYMMDD.IST.SEQ-NO Future spreads: PROD.V.YYMMDD.IST.SEQ-NO Flexible options: PROD FI YYYYMMDD CS AM P 101.82 0 Flexible futures: PROD FI YYYYMMDD CS
207	SecurityExchange	N	string	MIC (ISO 10383). Used, for derivatives, to identify an instrument of a co-operation partner. For cash this is the exchange where the IPO took place.
<DerivativesDescriptor> (optional) group starts				
461	CFICode	Y	string	Indicates the type of security using ISO 10962 standard.
762	SecuritySubType	N	uInt32	Standard strategy type for complex instruments. The mapping of integer values to the corresponding strategy types is published on the Eurex website at: www.eurexchange.com > Technology > Eurex Exchange's T7 > Documents
1227	ProductComplex	Y	Product-Complex (enum)	Type of instrument. **Value** / **Description** 1 – Simple Instrument 2 – Standard Option Strategy 3 – Non-Standard Option Strategy 4 – Volatility Strategy 5 – Futures Spread 6 – Inter-Product Spread 7 – Standard Futures Strategy 8 – Pack and Bundle 9 – Strip 10 – Flexible Instrument
<SimpleInstrumentDescriptor> (optional) group starts				
541	MaturityDate	N	uInt32	Actual expiration day of the instrument (YYYYMMDD).
200	MaturityMonthYear	N	uInt32	Expiration month (YYYYMM). Note that the actual expiration day provided by MaturityDate (541) may be in a different month.
202	StrikePrice	N	decimal	Strike price, e.g. 52.00.
2577	StrikePricePrecision	N	uInt32	Maximum number of decimal points for the strike price. Only relevant for display purposes.
2576	InstrumentPricePrecision	N	uInt32	Display decimals.
231	ContractMultiplier	N	decimal	Contract size, e.g. 100.
201	PutOrCall	N	PutOrCall (enum)	Defines if instrument is a put or call. **Value** / **Description** 0 – Put 1 – Call
206	OptAttribute	N	uInt32	Version of an option. The version changes as a result of corporate actions or events. The "new" option gets version zero, the "old" version gets a higher number and may exist temporarily in parallel until it is delisted. Delisting is done if no open interest exists for the Call and Put. For a new version zero a unique SecurityID is introduced.

Tag	Field Name	Req'd	Data Type	Description
1194	ExerciseStyle	N	ExerciseStyle (enum)	Style family of an option. **Value** / **Description**: 0 European; 1 American
2578	OrigStrikePrice	N	decimal	Original strike price prior to corporate action, e.g. 5.20.
25034	ContractGenerationNumber	N	uInt32	Contract generation.
2574	LowExercisePriceOptionIndicator	N	LepoFlag (enum)	LEPO Flag. **Value** / **Description**: N No; Y Yes
2575	BlockTradeEligibilityIndicator	N	BlkTrdFlag (enum)	Block trade flag. Always NULL for a Eurex product. **Value** / **Description**: N No; Y Yes
1197	ValuationMethod	Y	ValuationMethod (enum)	Traditional or futures margin style. **Value** / **Description**: EQTY Premium Style; FUT Futures Style Mark-to-Market
1193	SettlMethod	Y	SettlMethod (enum)	Settlement type. **Value** / **Description**: C Cash; P Physical
2579	SettlSubMethod	N	SettlSubMethod (enum)	Physical settlement type. **Value** / **Description**: 1 Shares; 2 Derivatives; 3 Payment Versus Payment; 4 Notional; 5 Cascade; 99 Other
734	PriorSettlPrice	N	decimal	Previous day's settlement price. Provided in clearing notation for variance futures and total return futures instruments.
55	Symbol	N	string	FlipId/FlexId, only for flexible instruments.
<SimpleInstrumentDescriptor> (optional) group ends				
<ComplexInstrumentDescriptorGroup> (optional) group starts				
<InstrmtLegGrp> sequence starts				
555	NoLegs	N	length	
600	> LegSymbol	N	uInt32	Product identifier of the leg security.
602	> LegSecurityID	Y	int64	Instrument identifier of the leg security.
603	> LegSecurityIDSource	Y	string	**Value** / **Description**: M Marketplace-assigned identifier
609	> LegSecurityType	Y	LegSecurityType (enum)	**Value** / **Description**: MLEG Part of a multi-leg instrument; ULEG Underlying leg
624	> LegSide	Y	LegSide (enum)	**Value** / **Description**: 1 Buy; 2 Sell

Tag	Field Name	Req'd	Data Type	Description
623	> LegRatioQty	N	uInt32	The ratio of quantity for this individual leg relative to the entire multileg security. For details on the leg ratio rules, please refer to the Functional Reference document on the Eurex website at www.eurexchange.com > Technology > T7 > System documentation > Release 4.0 > Overview and Functionality. The quantity for the leg of an order can be calculated as LegRatioQty(623) X OrderQty(38, ETI) = the Leg quantity.
566	> LegPrice	N	decimal	Price for the leg instrument. Used for the underlying leg of a Volatility Strategy.
<InstrmtLegGrp> sequence ends				
<ComplexInstrumentDescriptor> (optional) group ends				
60	TransactTime	N	timestamp	Creation time of flexible or complex instruments.
<DerivativesDescriptor> (optional) group ends				
<CashDescriptor> (optional) group starts				
1787	RefTickTableID	Y	uInt32	Reference to tick size table identifier from product level message.
120	SettlCurrency	Y	string	Settlement currency.
<InstrumentParties> (optional) sequence starts				
1018	NoInstrumentParties	Y	length	Number of parties.
1019	> InstrumentPartyID	Y	string	Identifies a party associated with an instrument.
1050	> InstrumentPartyIDSource	Y	InstrumentPartyIdSourceType (enum)	**Value** / **Description**: D – Proprietary
1051	> InstrumentPartyRole	Y	InstrumentPartyRoleType (enum)	**Value** / **Description**: 21 – Clearing Organization; 66 – Market Maker
2378	> InstrumentPartyRoleQualifier	Y	InstrumentPartyRoleQualifierType (enum)	**Value** / **Description**: 20 – Designated Sponsor
<InstrumentParties> (optional) sequence ends				
<InstrumentAttributes> (optional) sequence starts				
870	NoInstrAttrib	Y	length	Number of instrument attributes.
871	> InstrAttribType	Y	InstrumentAttributeType (enum)	**Value** / **Description**: 100 – Minimum Reserve Order Volume; 101 – Minimum Display Volume; 102 – Issuer Name; 103 – Issuer Number; 104 – Market Type; 105 – Market Type Supplement; 106 – Reporting Market
872	> InstrAttribValue	Y	string	
<InstrumentAttributes> (optional) sequence ends				
<CashDescriptor> (optional) group ends				
<Events> (optional) group starts				
864	NoEvents	N	length	Number of events.
865	> EventType	Y	EventType (enum)	**Value** / **Description**: 7 – Last Eligible Trade Date; 100 – First Eligible Trade Date
866	> EventDate	Y	uInt32	
<Events> (optional) group ends				

Tag	Field Name	Req'd	Data Type	Description
969	MinPriceIncrement	N	decimal	Defines the minimum increment for trade prices (tick size). This value is identical for all simple instruments of a product. Trading and clearing tick sizes, i.e. increments for trade prices in trading and clearing notation are the same unless MinPriceIncrementClearing(28888) is present.
28888	MinPriceIncrementClearing	N	decimal	Defines the minimum increment for trade prices in clearing notation (clearing tick size). Used for products to distinguish between tick sizes for trading and clearing notation, e.g. Total Return Futures.
1146	MinPriceIncrementAmount	N	decimal	Defines the minimum price movement in the respective currency (tick value). This value is identical for all instruments of a product. Amounts are only relevant for clearing tick sizes, i.e. to be multiplied with MinPriceIncrement(969) unless MinPriceIncrementClearing(28888) is present.
<MarketSegmentGrp> sequence starts				
1310	NoMarketSegments	Y	length	always "1".
1300	> MarketSegmentID	Y	uInt32	Product identifier or product pool identifier, e.g. 89.
1144	> ImpliedMarketIndicator	N	Implied-Market-Indicator (enum)	Defines the matching method. Only for derivatives. **Value** / **Description** 0 – Not implied 3 – Both implied-in and implied-out
1377	> MultilegModel	N	Multileg-Model (enum)	Defines if instrument is predefined by the exchange (=0) or user defined (=1). User defined instruments are deleted at the end of the day if the order book is empty. Only for derivatives. **Value** / **Description** 0 – Pre-defined multileg security 1 – User-defined multileg security
423	> PriceType	N	PriceType (enum)	The unit in which an instrument is quoted/stated when buying or selling. Only for cash. **Value** / **Description** 1 – Percent 2 – Shares 22 – Points
28876	> PostTradeAnonymity	N	PostTrade-Anonymity-Type (enum)	Only for cash. **Value** / **Description** 0 – No 1 – Yes 2 – Central Counterparty
><QuoteSizeRules> (optional) sequence starts				
2558	> NoQuoteSizeRules	N	length	Only one table. Only for cash.
647	> > MinBidSize	Y	uInt32	Bid side minimum quote quantity.
648	> > MinOfferSize	Y	uInt32	Offer side minimum quote quantity.
><QuoteSizeRules> (optional) sequence ends				
><PriceRangeRules> (optional) sequence starts				
2550	> NoPriceRangeRules	Y	length	Only one table.
2556	> > PriceRangeRuleID	Y	uInt32	Reference to table identifier from product level messages.
><PriceRangeRules> (optional) sequence ends				
<MarketSegmentGrp> sequence ends				

11.2.3 Instrument incremental message

Delivered on: Eurex RDI incremental feed, Eurex RDI snapshot feed

The *instrument incremental* message is used to report additions of complex and flexible instruments to the reference data.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: BP – Security Definition Update Report
980	SecurityUpdateAction	Y	string	Intraday creation of complex or flexible instruments. **Value** / **Description**: A – Add

Note: All fields from the Instrument snapshot message are also part of the message.

11.2.4 Variance futures status message

Delivered on: Eurex RDI incremental feed, Eurex RDI snapshot feed

The *Variance Futures Status* message is used to convey information specific to variance futures instruments.

During a normal day, a first initial snapshot message with the "Preliminary" attributes is published at the start of day, completed later by only one new message with the "Final" values.

In case the current business day is a holiday for the product, the status message will contain the information of the previous trading day.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: f – Security Status
34	MsgSeqNum	Y	uInt32	
1300	MarketSegmentID	Y	uInt32	Product identifier.
48	SecurityID	Y	int64	The instrument identifier uniquely identifies an instrument.
22	SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
167	SecurityType	Y	string	**Value** / **Description**: VAR – Variance Futures
><ClearingPriceParameters> sequence starts				
2580	NoClearingPriceParameters	Y	length	One table for each type of business day. Always "2".
2581	> BusinessDayType	Y	Business-Day-Type (enum)	**Value** / **Description**: 5 – Preceding Day; 1 – Current Day
2582	> ClearingPriceOffset	N	decimal	Price constant defined on the instrument level and used for the clearing price conversion.
2583	> VegaMultiplier	N	uInt64	Constant multiplier of the Notional Vega defined on the product level and used for the clearing quantity conversion.
2584	> AnnualTradingBusinessDays	N	uInt32	Approximate number of trading days during one year defined as a constant on the product level and used for the calculation of RealizedVariance (2587).
2585	> TotalTradingBusinessDays	N	uInt32	Total number of trading days of the instrument, including the first and the last trading day, which is one day before the expiration.
2586	> TradingBusinessDays	N	uInt32	Total number of trading days already passed since the introduction of the instrument.
2588	> StandardVariance	N	decimal	Used as a variance reference for the trading price conversion and the settlement price calculation. The standard variance is set to the implied volatility at the end of the first trading day. It is provided during the first trading day when parameters are final and then remains unchanged for the rest of the trading days.
2589	> RelatedClosePrice	N	decimal	Closing price of the underlying on the product level and used to calculate the realised variance. Also provided for previous day.
2587	> RealizedVariance	N	decimal	Calculated from all underlying closing prices since the introduction of the instrument adjusted by AnnualTradingBusinessDays (2584). Also provided for previous day.
1190	> RiskFreeRate	N	decimal	Interest rate till the instrument expiration interpolated from an interest curve. It is used to calculate the corresponding DiscountFactor (1592) till expiration. Also provided for previous day.
1592	> DiscountFactor	N	decimal	Calculated from the corresponding RiskFreeRate (1190) till expiration interpolated. Also provided for previous day.
2591	> AccumulatedReturnModifiedVariationMargin	N	decimal	Accumulated return on modified variation margin represents the economic cost of the variation margin from one trading day to the next. Also provided for previous day.

Tag	Field Name	Req'd	Data Type	Description
2590	> OvernightInterestRate	N	decimal	Short term interest rate used for the calculation of the next day ARMVM (2591).
1188	> Volatility	N	decimal	Implied volatility that has been used to calculate ClearingSettlPrice (2528). On the first trading day it is the base for StandardVariance (2588) of the instrument. Represents trading notation of ClearingSettlPrice (2528) and is only provided for previous day.
2528	> ClearingSettlPrice	N	decimal	Settlement price in clearing notation. Only provided for previous day.
2592	> CalculationMethod	N	CalculationMethodType (enum)	Indicates whether the automatic calculation of the parameters has been disabled and parameters have been manually defined. Intraday update is possible at any time.
><ClearingPriceParameters> sequence ends				

Value	Description
0	Automatic
1	Manual

Notes:
On the Eurex RDI incremental feed: The *MsgSeqNum* (34) starts with 1 and increments by 1 for each message.
On the Eurex RDI snapshot feed: The *MsgSeqNum* (34) is contiguous across all messages regardless of message type. See figure 9 chapter 9.2.1, General structure of the snapshot cycle.

11.2.5 Total return futures status message

Delivered on: Eurex RDI incremental feed, Eurex RDI snapshot feed

The *Total Return Futures Status* message is used to convey information specific to total return futures instruments.

During a normal day, a first message is published at the start of day with the following initial attributes: *AnnualCalendarDays* (28881), *RemainingCalendarDays* (28880) for the current day (*BusinessDayType* (2581) 1=Current Day) and *PriorSettlPrice* (734).

Later, still at the beginning of the day, a new message is sent containing the initial attributes completed with the following preliminary attributes: *OvernightInterestRate* (2590), *CurrentPaymentParameter* (28883), *AccruedPaymentParameter* (28884), *RelatedIndexValue* (28882), *CurrentCollectionParameter* (28885), *AccruedCollectionParameter* (28886) for the current day and *RelatedClosePrice* (2589) for the previous day (*BusinessDayType* (2581) 5=Preceding Day).

Finally, at the end of the day, a final message is sent with the final attributes, *RelatedClosePrice* (2589) for the current day and *SettlPrice* (730).

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** = f (Security Status)
34	MsgSeqNum	Y	uInt32	
1300	MarketSegmentID	Y	uInt32	Product identifier.
48	SecurityID	Y	int64	The instrument identifier uniquely identifies an instrument.
22	SecurityIDSource	Y	string	**Value** = M (Marketplace-assigned identifier)
167	SecurityType	Y	string	**Value** = TRF (Total Return Futures)
734	PriorSettlPrice	Y	decimal	Daily settlement price of the previous day in trading notation (TRF spread).
730	SettlPrice	N	decimal	Daily settlement price of the current day in trading notation (TRF spread).
<ClearingPriceParameters> sequence starts				
2580	NoClearingPriceParameters	Y	length	Two tables for previous and current day. Always "2".
2581	> BusinessDayType	Y	Business-Day-Type (enum)	**Value** = 5 (Preceding Day); 1 (Current Day)
28881	> AnnualCalendarDays	N	uInt32	Annualisation Factor, a constant value that represents the number of calendar days during one year as applied in the calculations.
28880	> RemainingCalendarDays	N	uInt32	Days to maturity, number of calendar days between the current day settlement date and the instrument expiration settlement date. Used for the trade price conversion from trading to clearing notation.
2590	> OvernightInterestRate	N	decimal	Funding rate, used for the funding calculation of the current day. Represents the overnight or the periodic interest rate established on the previous day. Becomes available in the morning before trading starts.
28883	> CurrentPaymentParameter	N	decimal	Daily funding, calculated from the funding rate applied to the previous day underlying index.
28884	> AccruedPaymentParameter	N	decimal	Accrued funding, accumulated from the daily fundings since the product launch. Used for the trade price conversion from trading to clearing notation.
28882	> RelatedIndexValue	N	decimal	Distribution index, defined for the current day. Becomes available in the morning before trading starts.
28885	> CurrentCollectionParameter	N	decimal	Daily distribution, calculated from the difference between the current and the previous day distribution index.

Tag	Field Name	Req'd	Data Type	Description
28886	> AccruedCollectionParameter	N	decimal	Accrued distribution, accumulated from the daily distributions since the product launch. Used for the trade price conversion from trading to clearing notation.
2589	> RelatedClosePrice	N	decimal	For BusinessDayType (2581) 5=Preceding Day: The previous day's underlying index close value. It is used as preliminary index value for the trade price conversion from trading to clearing notation, before the current day's index close value is known. For BusinessDayType (2581) 1=Current Day: The current day's underlying index close value. Once it is known, it is used as final index value for the trade price conversion from trading to clearing notation.
<ClearingPriceParameters> sequence ends				

Notes:
On the Eurex RDI incremental feed: The *MsgSeqNum* (34) starts with 1 and increments by 1 for each message.
On the Eurex RDI snapshot feed: The *MsgSeqNum* (34) is contiguous across all messages regardless of message type. See figure 9 chapter 9.2.1, General structure of the snapshot cycle.

11.3 Market data messages

The market data feeds disperse public market data via the Eurex EMDI and the Eurex MDI.

Public market data for all instruments are distributed over preconfigured multicast addresses. It is possible to configure multiple instruments over one multicast address and the depth of information to be disseminated can be configured on a per product basis. The multicast address and port combinations are different for the Eurex EMDI and the Eurex MDI.

Two different messages are used for order book updates: The *depth incremental* is sent if the order book changes (driven by an order book event). Conversely, the *depth snapshot* is sent in certain intervals independent from any change in the order book (time driven).

The message layout for the Eurex EMDI and Eurex MDI is the same.

11.3.1 Depth snapshot message

Delivered on: Eurex EMDI snapshot feed, Eurex MDI data feed

This message provides periodic updates for orders and trades independent from any change of the order book. Updates are available up to the maximum depth defined by the exchange in the field *MarketDepth (264)*. The Snapshot can be synchronized with the incremental message as described in chapter 6.5, Update the order book. One message per instrument with pre- and post trade data is sent. An empty book is disseminated during the product states as indicated in chapter 9.3, General order book rules and mechanics, bullet 5.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: W – Market Data Snapshot Full Refresh
34	MsgSeqNum	N	uInt32	Not used by unnetted feed (EMDI) where field is never present. The sequence number of the message is incremented per product across all message types.
49	SenderCompID	Y	uInt32	Unique id of a sender.
369	LastMsgSeqNumProcessed	N	uInt32	Not used by netted feed (MDI) where field is never present. Last message sequence number sent regardless of message type.
1187	RefreshIndicator	N	Refresh-Indicator (enum)	Used by netted feed (MDI) only. If set then the depth snapshot information has not been sent with the depth incremental before. **Value** / **Description**: Y – Mandatory Refresh; N – Optional Refresh
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. "89".
48	SecurityID	Y	int64	Instrument identifier, e.g. "8852".
22	SecurityIDSource	Y	string	Source Identification. **Value** / **Description**: M – Marketplace-assigned Identifier
1227	ProductComplex	Y	Product-Complex (enum)	Type of instrument **Value** / **Description**: 1 – Simple Instrument; 2 – Standard Option Strategy; 3 – Non-Standard Option Strategy; 4 – Volatility Strategy; 5 – Futures Spread; 6 – Inter-Product Spread; 7 – Standard Futures Strategy; 8 – Pack and Bundle; 9 – Strip; 10 – Flexible Instrument

Tag	Field Name	Req'd	Data Type	Description
965	SecurityStatus	Y	MDStatus (enum)	Status of the instrument. **Value** / **Description** 1 Active 2 Inactive 4 Expired 9 Suspended
25045	TESSecurityStatus	N	MDStatus (enum)	Status of the instrument for TES trading. **Value** / **Description** 1 Active 2 Inactive 4 Expired 9 Suspended
779	LastUpdateTime	Y	timestamp	Time of last change for SecurityID (nanoseconds). This can be any trade, change of the orderbook on any price level, or also a product or instrument state change information conveyed in this message.
<MDSshGrp> sequence starts				
268	NoMDEntries	Y	length	
1024	> MDOriginType	Y	MDOrigin-Type (enum)	**0 = Book** is for on-exchange trading. **1 = Off-Book** is for TES trading only. Only applicable for MDEntryType 2 = Trade or B = Trade Volume. **Value** / **Description** 0 Book 1 Off-Book
269	> MDEntryType	Y	MDEntry-Type (enum)	**Q = Auction Clearing Price** is sent as indicative information during the auction. **J = Empty Book** is sent during product states "Start-Of-Day" and "Pre-Trading" or when no price levels exist. During "Post-Trading" and "End-Of-Day" ToB prices are distributed. **B = Trade Volume** The total traded volume of units traded during the day can be found in the MDEntrySize field. Please note that the total traded volume may include coherent volume (from direct matching of complex instruments) as well. **b = Market Bid, c = Market Offer** Reserved for future release. **Value** / **Description** 0 Bid 1 Offer 2 Trade J Empty Book Q Auction Clearing Price B Trade Volume b Market Bid c Market Offer
1021	MDBookType	N	int	Price depth information or top-of-book information. Top-of-book is only applicable to IPS, i.e. requires the presence of MDSubBookType. **Value** / **Description** 1 Top of Book 2 Price Depth

Tag	Field Name	Req'd	Data Type	Description
1173	MDSubBookType	N	MDSubBook-Type (enum)	Optional qualifier to MDBookType. For IPS only. **1 = IPS implied volume without quantity restriction** Synthetically matchable quantity resulting from IPS instruments that is not subject to quantity restrictions. **2 = IPS implied volume with quantity restriction** Synthetically matchable quantity resulting from IPS instruments that is subject to quantity restrictions. **Value** / **Description**: 1 – IPS implied volume without quantity restriction; 2 – IPS implied volume with quantity restriction
828	> TrdType	N	TrdType [27] (enum)	For MDOriginType 0 = Book defines when the trade happens. Only present for MDEntryType 2 = Trade and TradeCondition AW = Last Auction Price. **1**, **2**, **12**, **55**, **1000**, **1001**, **1002** and **1004** only present for MDEntryType 2 = Trade and MDOriginType 1 = Off-Book. **Value** / **Description**: 1 – Off-Book Block Trade; 2 – Exchange For Physical (EFP); 12 – Exchange For Swap (EFS); 55 – Exchange Basis Facility; 1000 – Vola Trade; 1001 – EFP-Fin Trade; 1002 – EFP-Index-Futures Trade; 1004 – Block Trade at Market; 1100 – Opening Auction Trade; 1101 – Intraday Auction Trade; 1102 – Volatility Auction Trade; 1103 – Closing Auction Trade; 1107 – IPO Auction Trade
336	> TradingSessionID	N	Trading-SessionID (enum)	Always attached to the first MDEntry. **Value** / **Description**: 1 – Day; 3 – Morning; 5 – Evening; 6 – After-Hours; 7 – Holiday
625	> TradingSessionSubID	N	Trading-Session-SubID (enum)	See description for *TradingSessionID (336)*. **Value** / **Description**: 1 – Pre-Trading; 3 – Trading; 4 – Closing; 5 – Post-Trading; 7 – Quiescent
25044	> TESTradSesStatus	N	MDStatus (enum)	See description for *TradingSessionID (336)*. **Value** / **Description**: 1 – Halted; 2 – Open; 3 – Closed; 5 – Pre-Close

[27] Cross Auction Trade is never disseminated in the TrdType field of the Depth Snapshot message. However, it is disseminated in the TrdType field of the Depth Incremental message

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>2447</td><td>> FastMarketIndicator</td><td>N</td><td>Fast-Market-Indicator (enum)</td><td>See description for TradingSessionID (336).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>No</td></tr><tr><td>1</td><td>Yes</td></tr></table></td></tr>
<tr><td>326</td><td>> SecurityTradingStatus</td><td>N</td><td>Security-Trading-Status (enum)</td><td>See description for TradingSessionID (336).<table><tr><th>Value</th><th>Description</th></tr><tr><td>200</td><td>Closed</td></tr><tr><td>201</td><td>Restricted</td></tr><tr><td>202</td><td>Book</td></tr><tr><td>203</td><td>Continuous</td></tr><tr><td>204</td><td>Opening Auction</td></tr><tr><td>205</td><td>Opening Auction Freeze</td></tr><tr><td>206</td><td>Intraday Auction</td></tr><tr><td>207</td><td>Intraday Auction Freeze</td></tr><tr><td>208</td><td>Circuit Breaker Auction</td></tr><tr><td>209</td><td>Circuit Breaker Auction Freeze</td></tr><tr><td>210</td><td>Closing Auction</td></tr><tr><td>211</td><td>Closing Auction Freeze</td></tr><tr><td>212</td><td>IPO Auction</td></tr><tr><td>213</td><td>IPO Auction Freeze</td></tr></table></td></tr>
<tr><td>1174</td><td>> SecurityTradingEvent</td><td>N</td><td>Security-Trading-Event (enum)</td><td>Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>10</td><td>Price volatility, auction is extended</td></tr><tr><td>11</td><td>Price volatility, auction is extended again</td></tr></table></td></tr>
<tr><td>28872</td><td>> PotentialSecurity-TradingEvent</td><td>N</td><td>Security-Trading-Event (enum)</td><td>Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>10</td><td>Price volatility, auction is extended</td></tr></table></td></tr>
<tr><td>277</td><td>> TradeCondition</td><td>N</td><td>Trade-Condition (set)</td><td>AW cannot be combined with any other value and has its own entry in order to convey the auction type through TrdType. Instrument state already changed to continuous when the auction trade is reported.<table><tr><th>Value</th><th>Description</th></tr><tr><td>U</td><td>Exchange Last</td></tr><tr><td>R</td><td>Opening Price</td></tr><tr><td>AX</td><td>High Price</td></tr><tr><td>AY</td><td>Low Price</td></tr><tr><td>AJ</td><td>Official Closing Price</td></tr><tr><td>AW</td><td>Last Auction Price</td></tr><tr><td>BD</td><td>Previous Closing Price</td></tr></table></td></tr>
<tr><td>276</td><td>> QuoteCondition</td><td>N</td><td>Quote-Condition (enum)</td><td>"F" only present for IPS (prices without matching restrictions related to quantities that lead to a crossed book).
"Z" together with MDEntryType 0 or 1 to define a surplus (reserved for future release).<table><tr><th>Value</th><th>Description</th></tr><tr><td>F</td><td>Crossed book</td></tr><tr><td>Z</td><td>Order imbalance</td></tr></table></td></tr>
<tr><td>270</td><td>> MDEntryPx</td><td>N</td><td>decimal</td><td>Price.</td></tr>
<tr><td>271</td><td>> MDEntrySize</td><td>N</td><td>uInt32</td><td>Quantity or trade volume when MDEntryType is 2 = Trade or B = Trade Volume. TES disclosed quantity when MDOriginType is 1 = Off-Book.</td></tr>
<tr><td>346</td><td>> NumberOfOrders</td><td>N</td><td>uInt32</td><td></td></tr>
<tr><td>1023</td><td>> MDPriceLevel</td><td>N</td><td>uInt32</td><td>Book level. Absent for implied bid/offer prices.</td></tr>
</table>

Tag	Field Name	Req'd	Data Type	Description
273	> MDEntryTime	N	timestamp	Time of entry (nanoseconds) for last trade entry only (TradeCondition="U"). Statistics do not have an official timestamp in the snapshot, even if they happen to be identical to the last trade and be part of the same entry.
28873	> NonDisclosedTradeVolume	N	uInt32	Contains the TES trade volume that is not displayed during the day. Only present for MDEntryType B = Trade Volume. Used when trade volume is finally disclosed and also for recovery.
<MDSshGrp> sequence ends				

11.3.2 Depth incremental message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

This message provides order book updates and trades. Order book updates are available during Trading and Fast Trading states.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: X – Market Data Incremental Refresh
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. "89".
<MDIncGrp> sequence starts				
268	NoMDEntries	Y	length	
1024	> MDOriginType	Y	MDOrigin-Type (enum)	**0 = Book** is for on-exchange trading. **1 = Off-Book** is for TES trading only. Only applicable for MDEntryType 2 = Trade or B = Trade Volume. **Value** / **Description**: 0 – Book; 1 – Off-Book
279	> MDUpdateAction	Y	MDUpdate-Action (enum)	**Value** / **Description**: 0 – New; 1 – Change; 2 – Delete; 3 – Delete Thru; 4 – Delete From; 5 – Overlay
269	> MDEntryType	Y	MDEntry-Type (enum)	**B = Trade Volume** Trade volume entry for MDI, to provide new total trade volume from the last netting interval. Also used in EMDI for recovery purposes after a failover on the exchange side. In this case, the total traded volume of units traded during the day can be found in the MDEntrySize field. For MDOriginType 1 = Off-Book, "B" is also send together with NonDisclosedTradeVolume and MDEntrySize when TESTrad-SesStatus changes to 5 = Pre-Close. **b = Market Bid, c = Market Offer** Reserved for future release. **Value** / **Description**: 0 – Bid; 1 – Offer; 2 – Trade; Q – Auction Clearing Price; B – Trade Volume; b – Market Bid; c – Market Offer
48	> SecurityID	Y	int64	Instrument identifier, e.g. "8852".
22	> SecurityIDSource	Y	string	Source Identification. **Value** / **Description**: M – Marketplace-assigned Identifier
270	> MDEntryPx	N	decimal	Price of market data (trade or order).
271	> MDEntrySize	N	uInt32	Quantity or trade volume when MDEntryType = 2 or "B". TES disclosed quantity when MDOriginType 1 = Off-Book
346	> NumberOfOrders	N	uInt32	
1023	> MDPriceLevel	N	uInt32	Book level. Absent for implied bid/offer prices.
273	> MDEntryTime	N	timestamp	For bids and offers the official time of book entry, for trades official time of execution (all in nanoseconds).

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>28872</td><td>> PotentialSecurity-TradingEvent</td><td>N</td><td>Security-Trading-Event (enum)</td><td>Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>10</td><td>Price volatility, auction is extended</td></tr></table></td></tr>
<tr><td>276</td><td>> QuoteCondition</td><td>N</td><td>Quote-Condition (enum)</td><td>F only present for IPS (prices without matching restrictions related to quantities that lead to a crossed book).
Z together with MDEntryType 0 or 1 to define a surplus (reserved for future release).<table><tr><th>Value</th><th>Description</th></tr><tr><td>F</td><td>Crossed book</td></tr><tr><td>Z</td><td>Order imbalance</td></tr></table></td></tr>
<tr><td colspan="5"><TradeEntryGrp> (optional) group starts</td></tr>
<tr><td>828</td><td>> TrdType</td><td>N</td><td>TrdType (enum)</td><td>For MDOriginType 0 = Book defines when the trade happens. Only present for MDEntryType 2 = Trade and TradeCondition AW = Last Auction Price, with the exception of cross auction trades which do not establish a last auction price, i.e. TradeCondition AW is not set for these trades.
For trades outside the auctions, this field is not set.
1, 2, 12, 55, 1000, 1001, 1002 and 1004 only present for MDEntryType 2 = Trade and MDOriginType 1 = Off-Book.
1107 reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Off-Book Block Trade</td></tr><tr><td>2</td><td>Exchange For Physical (EFP)</td></tr><tr><td>12</td><td>Exchange For Swap (EFS)</td></tr><tr><td>55</td><td>Exchange Basis Facility</td></tr><tr><td>1000</td><td>Vola Trade</td></tr><tr><td>1001</td><td>EFP-Fin Trade</td></tr><tr><td>1002</td><td>EFP-Index-Futures Trade</td></tr><tr><td>1004</td><td>Block Trade at Market</td></tr><tr><td>1100</td><td>Opening Auction Trade</td></tr><tr><td>1101</td><td>Intraday Auction Trade</td></tr><tr><td>1102</td><td>Volatility Auction Trade</td></tr><tr><td>1103</td><td>Closing Auction Trade</td></tr><tr><td>1104</td><td>Cross Auction Trade</td></tr><tr><td>1107</td><td>IPO Auction Trade</td></tr></table></td></tr>
<tr><td>277</td><td>> TradeCondition</td><td>N</td><td>Trade-Condition (set)</td><td>Defines the type of price for MDEntryPx. Only present for MDEntryType 2 = Trade.
a = Volume Only used for coherent entries from direct matching of complex instruments (mutually exclusive with U).
k = Out of sequence used for trades entered manually by Market Supervision (mutually exclusive with U).<table><tr><th>Value</th><th>Description</th></tr><tr><td>U</td><td>Exchange Last</td></tr><tr><td>R</td><td>Opening Price</td></tr><tr><td>AX</td><td>High Price</td></tr><tr><td>AY</td><td>Low Price</td></tr><tr><td>AJ</td><td>Official Closing Price</td></tr><tr><td>AW</td><td>Last Auction Price</td></tr><tr><td>k</td><td>Out of sequence</td></tr><tr><td>BD</td><td>Previous Closing Price</td></tr><tr><td>a</td><td>Volume Only</td></tr></table></td></tr>
<tr><td>5979</td><td>> RequestTime</td><td>N</td><td>timestamp</td><td>Gateway-In timestamp.</td></tr>
<tr><td>2445</td><td>> AggressorTime</td><td>N</td><td>timestamp</td><td>Entry time of the incoming order that triggered the trade. Only present for MDEntryType=2.</td></tr>
</table>

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>2446</td><td>> AggressorSide</td><td>N</td><td>Aggressor-Side (enum)</td><td>Side of the incoming order that triggered the trade.
Only present for MDEntryType=2.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Buy</td></tr><tr><td>2</td><td>Sell</td></tr></table></td></tr>
<tr><td>2449</td><td>> NumberOfBuyOrders</td><td>N</td><td>uInt32</td><td>Number of buy orders involved in the trade. Only present for MDEntryType=2 and Trade Condition other than "a" (Volume Only).</td></tr>
<tr><td>2450</td><td>> NumberOfSellOrders</td><td>N</td><td>uInt32</td><td>Number of sell orders involved in the trade. Only present for MDEntryType=2 and Trade Condition other than "a" (Volume Only).</td></tr>
<tr><td>28869</td><td>> RestingCxlQty</td><td>N</td><td>uInt32</td><td>Quantity that was cancelled due to SMP. Only present for MDEntryType=2.</td></tr>
<tr><td>278</td><td>> MDEntryID</td><td>N</td><td>uInt32</td><td>Represents the match step ID. This field is unique together with MarketSegmentID. Only present for MDEntryType = 2.</td></tr>
<tr><td>28873</td><td>> NonDisclosedTradeVolume</td><td>N</td><td>uInt32</td><td>Contains the TES trade volume that is not displayed during the day. Only present for MDEntryType B = Trade Volume. Used when trade volume is finally disclosed and also for recovery.</td></tr>
<tr><td colspan="5"><TradeEntryGrp> (optional) group ends</td></tr>
<tr><td colspan="5"><MDIncGrp> sequence ends</td></tr>
</table>

11.3.3 Top Of Book Implied message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

This data message is used to convey top of book information resulting from synthetic IPS matching opportunities.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: X – Market Data Incremental Refresh
34	MsgSeqNum	N	uInt32	The sequence number of the message is incremented per product across different message types.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product id or pool id.
<MDIncGrp> sequence starts				
268	NoMDEntries	Y	length	
279	> MDUpdateAction	Y	MDUpdate-ActionType (enum)	**Value** / **Description**: 0 – To establish a new implied price.; 2 – To remove an existing price.
269	> MDEntryType	Y	MDEntry-Type (enum)	**Value** / **Description**: 0 – Bid; 1 – Offer
1021	> MDBookType	Y	int	Always Top of Book. **Value** / **Description**: 1 – Top of Book
1173	> MDSubBookType	Y	MDSub-Book-Type (enum)	Price / Quantity pairs. **Value** / **Description**: 1 – IPS implied volume without quantity restriction; 2 – IPS implied volume with quantity restriction
48	> SecurityID	Y	int64	Internal identifier assigned to each instrument.
22	> SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
270	> MDEntryPx	Y	decimal	Best implied price with or without a restriction.
271	> MDEntrySize	N	uInt32	Quantity.
273	> MDEntryTime	N	timestamp	For bids and offers the official time of book entry.
276	> QuoteCondition	N	Quote-Condition (enum)	Only present for prices without matching restrictions related to quantities that lead to a crossed book if displayed. Not used for prices with quantity restrictions (see *MDSubBook-Type (1173)*. **Value** / **Description**: F – Crossed book.
<MDIncGrp> sequence ends				

11.3.4 Product state change message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

The *product state change* message provides permanent updates on the trading state for a particular product.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: h – Trading Session Status
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. "89".
336	TradingSessionID	Y	Trading-SessionID (enum)	**Value** / **Description**: 1 – Day; 3 – Morning; 5 – Evening; 6 – After-Hours; 7 – Holiday
625	TradingSessionSubID	Y	Trading-Session-SubID (enum)	**Value** / **Description**: 1 – Pre-Trading; 3 – Trading; 4 – Closing; 5 – Post-Trading; 7 – Quiescent
340	TradSesStatus	Y	TradSes-Status (enum)	**Value** / **Description**: 1 – Halted; 2 – Open; 3 – Closed
2447	FastMarketIndicator	Y	Fast-Market-Indicator (enum)	Indicates if product is in state "Fast Market". **Value** / **Description**: 0 – No; 1 – Yes
60	TransactTime	Y	timestamp	
25044	TESTradSesStatus	N	TradSes-Status (enum)	**Value** / **Description**: 1 – Halted; 2 – Open; 3 – Closed; 5 – Pre-Close

11.3.5 Mass instrument state change message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

The *mass instrument state change* message provides the state information for all instruments of a certain instrument type within a product. Where not all indicated instruments are affected by the new state, the exception list (*SecurityTradingStatus (326)*) is populated with one entry for each such instrument.

Under Fast Market conditions, this message is sent with the *FastMarketIndicator (2447)* set but the actual state information may not have changed and is simply a re-statement of the previous information.

A state change affecting a single instrument (such as an intraday expiration) does not trigger a *mass instrument state change*.

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>35</td><td>MsgType</td><td>Y</td><td>string</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>CO</td><td>Security Mass Status</td></tr></table></td></tr>
<tr><td>34</td><td>MsgSeqNum</td><td>Y</td><td>uInt32</td><td>The sequence number is incremented per product across all message types on a particular feed.</td></tr>
<tr><td>49</td><td>SenderCompID</td><td>Y</td><td>uInt32</td><td>Unique id of a sender.</td></tr>
<tr><td>1300</td><td>MarketSegmentID</td><td>Y</td><td>uInt32</td><td>Product identifier, e.g. "89".</td></tr>
<tr><td>1544</td><td>InstrumentScopeProduct-Complex</td><td>Y</td><td>Instrument-Scope-Product-Complex (enum)</td><td>Instrument type of affected instruments.<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Simple Instrument</td></tr><tr><td>2</td><td>Standard Option Strategy</td></tr><tr><td>3</td><td>Non-Standard Option Strategy</td></tr><tr><td>4</td><td>Volatility Strategy</td></tr><tr><td>5</td><td>Futures Spread</td></tr><tr><td>6</td><td>Inter-Product Spread</td></tr><tr><td>7</td><td>Standard Futures Strategy</td></tr><tr><td>8</td><td>Pack and Bundle</td></tr><tr><td>9</td><td>Strip</td></tr><tr><td>10</td><td>Flexible Instrument</td></tr></table></td></tr>
<tr><td>1679</td><td>SecurityMassTradingStatus</td><td>N</td><td>Security-Trading-Status (enum)</td><td>212 = IPO Auction, 213 = IPO Auction Freeze Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>200</td><td>Closed</td></tr><tr><td>201</td><td>Restricted</td></tr><tr><td>202</td><td>Book</td></tr><tr><td>203</td><td>Continuous</td></tr><tr><td>204</td><td>Opening Auction</td></tr><tr><td>205</td><td>Opening Auction Freeze</td></tr><tr><td>206</td><td>Intraday Auction</td></tr><tr><td>207</td><td>Intraday Auction Freeze</td></tr><tr><td>208</td><td>Circuit Breaker Auction</td></tr><tr><td>209</td><td>Circuit Breaker Auction Freeze</td></tr><tr><td>210</td><td>Closing Auction</td></tr><tr><td>211</td><td>Closing Auction Freeze</td></tr><tr><td>212</td><td>IPO Auction</td></tr><tr><td>213</td><td>IPO Auction Freeze</td></tr></table></td></tr>
<tr><td>1680</td><td>Security-MassTradingEvent</td><td>N</td><td>Security-Trading-Event (enum)</td><td>Identifies an event related to a SecurityMassTradingStatus(1679). Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>10</td><td>Price volatility, auction is extended</td></tr><tr><td>11</td><td>Price volatility, auction is extended again</td></tr></table></td></tr>
<tr><td>2447</td><td>FastMarketIndicator</td><td>Y</td><td>Fast-Market-Indicator (enum)</td><td>Indicates if product is in state "Fast Market". This indicator refers to a product but is provided on instrument level.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>No</td></tr><tr><td>1</td><td>Yes</td></tr></table></td></tr>
<tr><td>60</td><td>TransactTime</td><td>Y</td><td>timestamp</td><td>Time when request was processed by the matcher (nanoseconds).</td></tr>
<tr><td colspan="5"><SecMassStatGrp> sequence starts</td></tr>
<tr><td>146</td><td>NoRelatedSym</td><td>N</td><td>length</td><td></td></tr>
<tr><td>48</td><td>> SecurityID</td><td>Y</td><td>int64</td><td>Instrument identifier, e.g. "8852".</td></tr>
<tr><td>22</td><td>> SecurityIDSource</td><td>Y</td><td>string</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>M</td><td>Marketplace-assigned Identifier</td></tr></table></td></tr>
</table>

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>965</td><td>> SecurityStatus</td><td>Y</td><td>Security-Status (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Active</td></tr><tr><td>2</td><td>Inactive</td></tr><tr><td>4</td><td>Expired</td></tr><tr><td>9</td><td>Suspended</td></tr></table></td></tr>
<tr><td>326</td><td>> SecurityTradingStatus</td><td>N</td><td>Security-Trading-Status (enum)</td><td>212 = IPO Auction, 213 = IPO Auction Freeze Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>200</td><td>Closed</td></tr><tr><td>201</td><td>Restricted</td></tr><tr><td>202</td><td>Book</td></tr><tr><td>203</td><td>Continuous</td></tr><tr><td>204</td><td>Opening Auction</td></tr><tr><td>205</td><td>Opening Auction Freeze</td></tr><tr><td>206</td><td>Intraday Auction</td></tr><tr><td>207</td><td>Intraday Auction Freeze</td></tr><tr><td>208</td><td>Circuit Breaker Auction</td></tr><tr><td>209</td><td>Circuit Breaker Auction Freeze</td></tr><tr><td>210</td><td>Closing Auction</td></tr><tr><td>211</td><td>Closing Auction Freeze</td></tr><tr><td>212</td><td>IPO Auction</td></tr><tr><td>213</td><td>IPO Auction Freeze</td></tr></table></td></tr>
<tr><td>1174</td><td>> SecurityTradingEvent</td><td>N</td><td>Security-Trading-Event (enum)</td><td>Identifies an event related to a SecurityTradingStatus(326). Reserved for future release.<table><tr><th>Value</th><th>Description</th></tr><tr><td>10</td><td>Price volatility, auction is extended</td></tr><tr><td>11</td><td>Price volatility, auction is extended again</td></tr></table></td></tr>
<tr><td>25045</td><td>> TESSecurityStatus</td><td>N</td><td>MDStatus (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Active</td></tr><tr><td>2</td><td>Inactive</td></tr><tr><td>4</td><td>Expired</td></tr><tr><td>9</td><td>Suspended</td></tr></table></td></tr>
<tr><td colspan="5"><SecMassStatGrp> sequence ends</td></tr>
</table>

11.3.6 Instrument state change message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

The *instrument state change* message provides state information for a single instrument. It also informs participants about intraday expirations of instruments. In that case the field *SecurityStatus (965)* is set to "Expired".

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: f – Security Status
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. "89".
48	SecurityID	Y	int64	Instrument identifier, e.g. "8852".
22	SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
965	SecurityStatus	Y	MDStatus (enum)	**Value** / **Description**: 1 – Active; 2 – Inactive; 4 – Expired; 9 – Suspended
326	SecurityTradingStatus	N	Security-Trading-Status (enum)	Trading status of an instrument. **Value** / **Description**: 200 – Closed; 201 – Restricted; 202 – Book; 203 – Continuous; 204 – Opening Auction; 205 – Opening Auction Freeze; 206 – Intraday Auction; 207 – Intraday Auction Freeze; 208 – Circuit Breaker Auction; 209 – Circuit Breaker Auction Freeze; 210 – Closing Auction; 211 – Closing Auction Freeze; 212 – IPO Auction; 213 – IPO Auction Freeze
2447	FastMarketIndicator	Y	Fast-Market-Indicator (enum)	Indicates if product is in state "Fast Market". This indicator refers to a product but is provided on instrument level. **Value** / **Description**: 0 – No; 1 – Yes
1174	SecurityTradingEvent	N	Security-Trading-Event (enum)	Identifies an event related to a SecurityTradingStatus(326). Reserved for future release. **Value** / **Description**: 10 – Price volatility, auction is extended; 11 – Price volatility, auction is extended again
60	TransactTime	Y	timestamp	Time when request was processed by the matcher (nanoseconds).

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>25045</td><td>TESSecurityStatus</td><td>N</td><td>MDStatus
(enum)</td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Active</td></tr>
<tr><td>2</td><td>Inactive</td></tr>
<tr><td>4</td><td>Expired</td></tr>
<tr><td>9</td><td>Suspended</td></tr>
</table>
</td></tr>
</table>

11.3.7 Quote request message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

Market participants can enter a quote request (Trading Interest) that needs to be answered with a quote by Market Makers. The quote request message shows such requests from traders.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: R – Quote Request
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. 89"."
<QuotReqGrp> sequence starts				
146	NoRelatedSym	Y	length	always "1".
48	> SecurityID	Y	int64	Instrument identifier, e.g. 8852"."
22	> SecurityIDSource	Y	string	Identifies class or source of the SecurityID (48) value. Required if SecurityID is specified. Will be sent as a constant. **Value** / **Description**: M – Marketplace-assigned Identifier.
54	> Side	N	Side (enum)	**Value** / **Description**: 1 – Buy; 2 – Sell
38	> OrderQty	N	uInt32	Defines the requested quantity which can be zero in a quote request.
60	> TransactTime	Y	timestamp	Time when request was processed by the matcher (nanoseconds).
<QuotReqGrp> sequence ends				

11.3.8 Cross request message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

A *crossing* is defined as intentional or unintentional execution of orders and quotes against a preselected member or in-house. Using the Cross Request, all Members are informed of a crossing or a pre-arranged trade that shall be executed in the Eurex order book (on-exchange). Other market participants see the order and can also match against them. Eurex expects the orders to be entered within a certain time frame.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: U16 – Cross Request
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender.
1300	MarketSegmentID	Y	uInt32	Product identifier, e.g. 89"."
48	SecurityID	Y	int64	Instrument identifier, e.g. 8852"."
22	SecurityIDSource	Y	string	Identifies class or source of the SecurityID (48) value. Required if SecurityID is specified. Will be sent as a constant "M".
38	OrderQty	Y	uInt32	Defines the requested quantity which cannot be zero in a cross request.
60	TransactTime	Y	timestamp	Time when request was processed by the matcher (nanoseconds).

11.3.9 Complex instrument update message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

This message provides information for new or inactivated complex instruments. This message has a similar structure as the *instrument incremental message* described in section 11.2.3.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: BP – Security Definition Update Report
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	N	uInt32	Unique id of a sender
980	SecurityUpdateAction	Y	string	Intraday creation of complex instruments. **Value** / **Description**: A – Add
48	SecurityID	Y	int64	The instrument identifier uniquely identifies an instrument, e.g. 72057800196358145.
22	SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
107	SecurityDesc	Y	string	The format depends on the type of instrument: Standard options strategies: PROD.O.YYMMDD.IST.SEQ-NO Non-standard options strategies: PROD.N.YYMMDD.SEQ-NO Options volatility strategies: PROD.V.YYMMDD.IST.SEQ-NO Future spreads: PROD.S.MONYY.MONYY
167	SecurityType	Y	Security-Type (enum)	Type of security. **Value** / **Description**: MLEG – complex instrument
762	SecuritySubType	N	uInt32	Standard strategy type for complex instruments. The mapping of integer values to the corresponding strategy types is published on the Eurex website at: www.eurexchange.com > Technology > Eurex Exchange's T7 > Documents
1227	ProductComplex	Y	Product-Complex (enum)	Type of instrument. **Value** / **Description**: 1 – Simple Instrument; 2 – Standard Option Strategy; 3 – Non-Standard Option Strategy; 4 – Volatility Strategy; 5 – Futures Spread; 6 – Inter-Product Spread; 7 – Standard Futures Strategy; 8 – Pack and Bundle; 9 – Strip; 10 – Flexible Instrument
<InstrmtLegGrp> sequence starts				
555	NoLegs	N	length	
600	> LegSymbol	N	uInt32	Product identifier of the leg security.
602	> LegSecurityID	Y	int64	Instrument identifier of the leg security.
603	> LegSecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>609</td><td>> LegSecurityType</td><td>Y</td><td>LegSecurity-Type (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>MLEG</td><td>Part of a multi-leg instrument</td></tr><tr><td>ULEG</td><td>Underlying leg</td></tr></table></td></tr>
<tr><td>624</td><td>> LegSide</td><td>Y</td><td>LegSide (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Buy</td></tr><tr><td>2</td><td>Sell</td></tr></table></td></tr>
<tr><td>623</td><td>> LegRatioQty</td><td>N</td><td>uInt32</td><td>The ratio of quantity for this individual leg relative to the entire multileg security.</td></tr>
<tr><td>566</td><td>> LegPrice</td><td>N</td><td>decimal</td><td>Price for the leg instrument. Used for the underlying leg of a Volatility Strategy.</td></tr>
<tr><td colspan="5"><InstrmtLegGrp> sequence ends</td></tr>
<tr><td colspan="5"><MarketSegmentGrp> sequence starts</td></tr>
<tr><td>1310</td><td>> NoMarketSegments</td><td>N</td><td>length</td><td>always “1”.</td></tr>
<tr><td>1300</td><td>> MarketSegmentID</td><td>Y</td><td>uInt32</td><td>Product identifier, e.g. 89.</td></tr>
<tr><td>1144</td><td>> ImpliedMarketIndicator</td><td>N</td><td>Implied-Market-Indicator (enum)</td><td>Matching method.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Not implied</td></tr><tr><td>3</td><td>Both implied-in and implied-out</td></tr></table></td></tr>
<tr><td colspan="5"><MarketSegmentGrp> sequence ends</td></tr>
<tr><td>60</td><td>TransactTime</td><td>N</td><td>timestamp</td><td>Creation time of complex instruments.</td></tr>
</table>

11.3.10 Flexible instrument update message

Delivered on: Eurex EMDI incremental feed, Eurex MDI data feed

This message provides information for new flexible instruments.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	**Value** / **Description**: BP – Security Definition Update Report
34	MsgSeqNum	Y	uInt32	The sequence number is incremented per product across all message types on a particular feed.
49	SenderCompID	Y	uInt32	Unique id of a sender
980	SecurityUpdateAction	Y	string	Intraday creation of flexible instruments. **Value** / **Description**: A – Add
55	Symbol	Y	string	FlipId / FlexId.
48	SecurityID	Y	int64	The instrument identifier uniquely identifies a flexible instrument.
22	SecurityIDSource	Y	string	**Value** / **Description**: M – Marketplace-assigned identifier
107	SecurityDesc	Y	string	The format depends on the type of instrument. Options: OGBM FI 20140516 CS AM P 101.82 0 Futures: FGBM FI 20140620 CS
167	SecurityType	Y	SecurityType (enum)	Type of security. **Value** / **Description**: OPT – Option; FUT – Future
1227	ProductComplex	Y	ProductComplex (enum)	Type of instrument. **Value** / **Description**: 10 – Flexible Instrument
541	MaturityDate	Y	uInt32	Actual expiration day of the instrument (YYYYMMDD).
202	StrikePrice	N	decimal	Strike price.
201	PutOrCall	N	PutOrCall (enum)	Defines if instrument is a put or call. **Value** / **Description**: 0 – Put; 1 – Call
206	OptAttribute	Y	uInt32	Version of an option.
1194	ExerciseStyle	N	ExerciseStyle (enum)	Style family of an option. **Value** / **Description**: 0 – European; 1 – American
1193	SettlMethod	Y	SettlMethod (enum)	Settlement type. **Value** / **Description**: C – Cash; P – Physical
<MarketSegmentGrp> sequence starts				
1310	NoMarketSegments	Y	length	always "1".
1300	> MarketSegmentID	Y	uInt32	Product identifier or product pool identifier, e.g. 89.
<MarketSegmentGrp> sequence ends				
60	TransactTime	N	timestamp	Creation time of flexible instruments.

11.4 Data files

11.4.1 Reference data from file (Eurex RDF)

As an alternative to the reference data feed, participants can receive reference data in file format. This provision is made especially for bandwidth conscious users who want to prevent an overload of their line capacity caused by joining the reference data feed.

For details regarding FIXML Schema design rules please refer to the document *Version 5.0 Service Pack 2 (with Errata 20110818)* available under www.fixprotocol.org > Technical / Specifications.

Eurex provides a customized FIXML schema via a set of xsd files. They can be found in zipped format at this address: www.eurexchange.com > Technology > T7 > System documentation > Release 4.0 > Market and Reference Data Interfaces.

As per the FIXML specification, Eurex RDF field names are different from the Eurex RDI field names. A mapping table between the fields can be found in section 14.3, FIXML mapping table.

The reference data files can be received via the Common Report Engine as described in the "Common Report Engine User Guide".

Similar to the Eurex RDI, the Eurex RDF mechanism operates on a snapshot plus incremental basis.

The initial file ("snapshot") is created around 6:00 CET and contains all of the instruments defined on the exchange at the beginning of the business day.

An incremental file is created up to every five minutes thereafter and contains the creation events for complex and flexible instruments along with the variance futures and total return futures status messages disseminated in that period. Each of these incremental files must be applied to the initial file (the "snapshot").

If there have been no changes in a given five minute interval, no file is created (i.e. empty files are not created).

Other options to receive intra day created complex instruments are described in section 9.2.7, Use case 5: Chronological order of messages for complex instrument creation.

11.4.2 File name format of the reference data files

The Reference Data files are provided in FIXML format. Product reference data and instrument reference data are provided in one file. The file format of the Start-Of-Day and intraday files looks as follows:

Content	File Name
Start-Of-Day file	[@@][report name][member][business date][mic][file set identifier][nnn].XML
Intraday updates	[@@][report name][member][business date][mic][file set identifier][nnn].XML

Table 51: *Format of the reference data files*

with:
@@ (length 2): environment number, i.e. '90' for production and 95 for simulation
report name (length 8): always 'FILRDF01'
member id (length 5): always 'PUBLI'
business date (length 8): format 'YYYYMMDD'
mic (length 4): MIC code, XEUR for Eurex XEEE for EEX
file set identifier (length 5): remains constant for all files belonging to the same set
nnn (length 3): sequence number 000-999

Example for reference data files in simulation:
Start-Of-Day file for Eurex: 95FILRDF01PUBLI20120815XEURCZA80000.XML
First intraday updated file for Eurex: 95FILRDF01PUBLI20120815XEURCZA80001.XML
Second intraday updated file for Eurex: 95FILRDF01PUBLI20120815XEURCZA80002.XML
...

Note: In release 3.0 the file name always contains 2 MIC codes because it holds products from two markets. In release 4.0 each market will have its own file, therefore the file name will contain only one MIC code.

11.4.3 Reference data file on the next business day

Complex instruments which still exist on the next business day and which have been sent as intraday file updates on the previous day are incorporated into the Start-Of-Day file on the next day.

11.4.4 Reference data file after a failover or restart of Eurex RDI

The file set identifier changes after a restart or failover of the Eurex RDI. The file set identifier always lexically increments within a day, allowing for easy identification of the most recently created set. During a failover or restart, the application must detect the creation of a new file set and rebuild its reference data based on the new file set.

Example:

Start-Of-Day file **after** failover : 95FILRDF01PUBLI20120815XEUR**CZLC0**000.XML
First intraday updated file **after** failover : 95FILRDF01PUBLI20120815XEUR**CZLC0**001.XML
Second intraday updated file **after** failover: 95FILRDF01PUBLI20120815XEUR**CZLC0**002.XML
...

If Eurex RDI fails over, a new initial reference file is generated with a new file set identifier. This file contains all incremental messages disseminated during the day, i.e. the entire history.

If Eurex RDI needs to be restarted by the exchange, a new initial reference file is generated with a new file set identifier. This file contains the currently existing complex instruments but not the entire history of incremental messages.

11.4.5 What receiving applications need to do

Applications using a sftp process need to process the following loop:

1. Find the newest Start-Of-Day file.
2. Apply all updates newer than the Start-Of-Day file.
3. Detect the creation of a new file set.

12 Multicast addresses

The reference information provided by Eurex RDI contains the respective multicast channel information (i.e. multicast addresses and port numbers) for all available products.

12.1 Reference data snapshot feed

Environment	Service A	Service B
EUREX Production	224.0.50.0:59098	224.0.50.128:59098
EEX Production	224.0.29.0:59098	224.0.30.0:59098
EUREX Simulation	224.0.50.96:59598	224.0.50.224:59598
EEX Simulation	224.0.29.96:59598	224.0.30.96:59598

Table 52: *Multicast addresses and ports for the reference data snapshot feeds*

12.2 Reference data incremental feed

Environment	Service A	Service B
EUREX Production	224.0.50.1:59099	224.0.50.129:59099
EEX Production	224.0.29.1:59099	224.0.30.1:59099
EUREX Simulation	224.0.50.97:59599	224.0.50.225:59599
EEX Simulation	224.0.29.97:59599	224.0.30.97:59599

Table 53: *Multicast addresses and ports for the reference data incremental feed*

12.3 Reference data for Eurex Enhanced Order Book Interface

The reference data information such as multicast addresses and port numbers of corresponding products, which are needed to receive public market data via Eurex Enhanced Order Book Interface is available via the Eurex Reference Data Interface (Eurex RDI) and the Eurex Reference Data Files (Eurex RDF).

The *Product Snapshot* message contains the following information for products configured for Eurex Enhanced Order Book Interface:

- Book Type, *MDBookType (1021)*, has value 3 (Order Depth).
- Feed Type, *MDFeedType (1022)*, has value HI (HighIncremental) for Order-By-Order incremental messages and HS (HighSnapshot) for Order-By-Order snapshot messages.
- The IP multicast address and port number of the primary Eurex Enhanced Order Book feed can be obtained via fields *PrimaryServiceLocationID (2567)* and *PrimaryServiceLocationSubID (28591)*.
- The IP multicast address and port number of the secondary Eurex Enhanced Order Book feed can be obtained via fields *SecondaryServiceLocationID (2568)* and *SecondaryServiceLocationSubID (28593)*.

The same information is also available via the Eurex Reference Data Files (Eurex RDF).

13 FAST templates

Two versions for FAST templates are offered:

- FAST templates based on version 1.2
- FAST templates compatible with version 1.1

Participants can either use a decoder which has the new FAST 1.2 feature implemented or use their existing decoder based on FAST 1.1.

The FAST XML files are provided by Eurex in separate files:

- EMDIFastTemplates*.xml for all messages on the EMDI snapshot and incremental feeds.
- MDIFastTemplates*.xml for all messages on the MDI feed.
- RDDFastTemplates*.xml for all messages on the RDI snapshot and incremental feeds.

The FAST templates can be downloaded at the member section of the Eurex website at:

www.eurexchange.com > Technology > T7 > System documentation > Release 4.0 > Market and Reference Data Interfaces > FAST 1.2 Templates.

14 Appendix

14.1 Example for a XML FAST template

This example refers to chapter 5.3, Decoding the FAST-message.

```
- <template id="86" name="DepthIncremental">
- <string name="MsgType" id="35">
  <constant value="X" />
    </string>
- <uInt32 name="MsgSeqNum" id="34">
  <increment />
    </uInt32>
- <uInt32 name="SenderCompID" id="49">
  <copy />
    </uInt32>
- <uInt32 name="MarketSegmentID" id="1300">
  <copy />
    </uInt32>
- <sequence name="MDEntriesGrp">
  <length name="NoMDEntries" id="268" />
- <field name="MDUpdateAction" id="279">
  <type name="MDUpdateAction" />
    </field>
- <field name="MDEntryType" id="269">
  <type name="MDEntryType" />
    </field>
  <uInt64 name="SecurityID" id="48" />
- <string name="SecurityIDSource" id="22">
  <constant value="M" />
    </string>
- <decimal name="MDEntryPx" id="270" presence="optional">
  <delta />
    </decimal>
  <uInt32 name="MDEntrySize" id="271" presence="optional" />
  <uInt32 name="NumberOfOrders" id="346" presence="optional" />
- <uInt32 name="MDPriceLevel" id="1023" presence="optional">
  <delta />
    </uInt32>
- <timestamp name="MDEntryTime" unit="nanosecond" id="273">
  <copy />
    </timestamp>
- <group name="TradeEntryGrp" presence="optional">
  <uInt32 name="TradeVolume" id="1020" presence="optional" />
- <field name="TradeCondition" id="277" presence="optional">
  <type name="TradeConditionSet" />
    </field>
  <uInt32 name="GapIndicator" id="8719" presence="optional" />
  <timestamp name="AggressorTimestamp" unit="nanosecond" id="8720" presence="optional" />
<field name="AggressorSide" id="5797" presence="optional">
  <type name="Side" />
    </field>
  <uInt32 name="NumberOfBuyOrders" id="8721" presence="optional" />
  <uInt32 name="NumberOfSellOrders" id="8722" presence="optional" />
  <uInt32 name="MDEntryID" id="278" />
    </group>
    </sequence>
    </template>
```

Figure 21: *Example for a FAST template with repeating group*

14.2 Example for determination of the price source

14.2.1 Fully implied

Example for 9.3.1, Determination of the price sources

Precondition: Empty order book.

Action: Buy FDAX Mar 10@100, Sell FDAX Jun 10@98

Result: Fully implied price in future spread, Buy FDAX Mar/Jun 10@2 gets created (implied price is sent). Rule 1 in section 9.3.1, Determination of the price sources applies.

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1000	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	0	Bid
48	> SecurityID	8875	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	2	Price
271	> MDEntrySize	10	Quantity
346	> NumberOfOrders	N/A	Number of order/quotes on this level
1023	> MDPriceLevel	N/A	empty indicates implied price
273	> MDEntryTime	t_0	official time of book entry

Table 54: *Fully implied*

14.2.2 Fully outright on level 1

Example for 9.3.1, Determination of the price sources

Precondition: Fully implied price in future spread created above.

Action: Buy FDAX Mar/Jun 8@1.5

Result: Fully outright price gets created on price level 1 (Implied price is absent). No update for implied is sent as it has already been conveyed (see section 14.2.1). Rule 1 in section 9.3.1, Determination of the price sources applies.

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1001	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	0	Bid
48	> SecurityID	8875	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	1.5	Price
271	> MDEntrySize	8	Quantity
346	> NumberOfOrders	1	Number of order/quotes on this level
1023	> MDPriceLevel	1	Price level 1
273	> MDEntryTime	t_1	official time of book entry

Table 55: *Fully outright sent for level 1*

14.2.3 Partially implied

Example for 9.3.1, Determination of the price sources

Precondition: Fully implied price in future spread created before (see section 14.2.1):

- fully implied Buy FDAX (Mar/Jun) 10@2
- outright 8@1.5 (see section 14.2.2)

Action: Buy FDAX Mar/Jun 5@2 (new outright price in future spread)

Result: The initially fully implied price becomes partially implied. Rule 2 in section 9.3.1, Determination of the price sources) applies. The implied price (without price level) is not sent again as it has already been conveyed (see section 14.2.1).

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum	1002	
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	0	Bid
48	> SecurityID	8875	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	2	Price
271	> MDEntrySize	5	Quantity
346	> NumberOfOrders	1	Number of order/quotes on this level
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_2	official time of book entry

Table 56: *Partially implied*

14.2.4 Several fully implied orders at Best Market

Eample for 9.3, General order book rules and mechanics

Precondition: Fully implied Buy FDAX Sep 100@117

Action: Buy FDAX Jun 200@118, Sell FDAX Jun/Sep 200@1.
This creates an implied for Buy FDAX Sep at 200@(118-1) = 200@117.

Result: The reported quantity is the largest of the quantities from the implied orders at the same price, i.e. 200@117 is reported.

Rule 1 in section 9.3.1, Determination of the price sources applies.

Tag number	Tag name	Value	Description
35	MsgType	X	MarketDataIncrementalRefresh
34	MsgSeqNum		
49	SenderCompID	75	Unique id of a sender
1300	MarketSegmentID	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	0	New
269	> MDEntryType	0	Bid
48	> SecurityID	7510	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	117	Price
271	> MDEntrySize	200	Quantity
346	> NumberOfOrders	N/A	Number of order/quotes on this level
1023	> MDPriceLevel	N/A	empty indicates implied price
273	> MDEntryTime	t_3	official time of book entry

Table 57: *Quantities are added for two or more fully implied orders at the same price.*

14.3 FIXML mapping table

The following table provides a mapping between FIXML attributes and FIX fields. The mapping table belongs to chapter 11.4.1, Reference data from file (Eurex RDF). *SecDef* refers to the *instrument snapshot* message, *SecDefUpd* refers to the *instrument incremental* message, *MktDef* refers to the *product snapshot* message and *SecStat* refers to both the *Variance Futures Status* and the *Total Return Futures* messages.

The initial file ("snapshot") includes all four message types while the incremental file contains only the *SecDefUpd* and *SecStat* messages.

FIXML attributes of the Eurex RDF occur in the same sequence as FIX fields of the Eurex RDI.

The *market data report* message is not used in the Eurex RDF because the start and end of the files coincides with the MDReportEvent values.

Example:
MDReportEvent = 1 = "StartOfReferenceData" = Begin of file
MDReportEvent = 2 = "EndOfReferenceData" = End of file

FIXML Attribute	FIX Field Name	FIX Tag / Group	Repeating	FIX Messages
AcrdColtnPrm	AccruedCollectionParameter	28885		SecStat
AcrdPmtPrm	AccruedPaymentParameter	28884		SecStat
AID	SecurityAlt	Group	Y	SecDef, SecDefUpd
AltID	SecurityAltID	455		SecDef, SecDefUpd
AltIDSrc	SecurityAltIDSource	456		SecDef, SecDefUpd
AnnlClndrDays	AnnualCalendarDays	28881		SecStat
AnnlTrdgBizDays	AnnualTradingBusinessDays	2584		SecStat
ARMVM	ARMVM	2591		SecStat
AuctTyp	AuctionType	1803		MktDef
AuctTypRule	AuctionTypeRules	Group	Y	MktDef
BaseTrdgRules	BaseTradingRules	Group	N	MktDef, SecDef, SecDefUpd
BizDayTyp	BusinessDayType	2581		SecStat
BlckTrdEligInd	BlockTradeEligibilityIndicator	2575		SecDef, SecDefUpd
CalcMeth	CalculationMethod	2592		SecStat
Ccy	Currency	15		MktDef
CFI	CFICode	461		SecDef, SecDefUpd
ClrPxOfst	ClearingPriceOffset	2582		SecStat
ClrPxPrm	ClearingPriceParameters	Group	Y	SecStat
ClsdBkInd	ClosedBookIndicator	28874		MktDef
CntrctGenNum	ContractGenerationNumber	25034		SecDef, SecDefUpd
CurColtnPrm	CurrentCollectionParameter	28885		SecStat
CurPmtPrm	CurrentPaymentParameter	28883		SecStat
Desc	SecurityDesc	107		SecDef, SecDefUpd
DiscFctr	DiscountFactor	1592		SecStat
Dt	EventDate	866		SecDef, SecDefUpd
EfctvBizDt	EffectiveBusinessDate	2400		MktDef
EndPxRng	EndPriceRange	2552		MktDef
EndTickPxRng	EndTickPriceRange	1207		MktDef
EventTyp	EventType	865		SecDef, SecDefUpd
Evnt	Events	Group	Y	SecDef, SecDefUpd
Exch	SecurityExchange	207		SecDef, SecDefUpd

FIXML Attribute	FIX Field Name	FIX Tag / Group	Repeating	FIX Messages
Exch	UnderlyingSecurityExchange	30308		MktDef
ExerStyle	ExerciseStyle	1194		SecDef, SecDefUpd
FastMktInd	FastMarketIndicator	2447		MktDef
FastMktPctg	FastMarketPercentage	2557		MktDef
FlexProdElig	FlexProductEligibilityIndicator	1242		MktDef
FlexProdEligCmplx	FlexProductEligibilityComplex	2561		MktDef
FlexProdEligs	FlexRules	Group	Y	MktDef
ID	InstrumentPartyID	1019		SecDef, SecDefUpd
ID	LegSecurityID	602		SecDef, SecDefUpd
ID	RelatedMarketSegmentID	2546		MktDef
ID	SecurityID	48		SecDef, SecDefUpd, SecStat
ID	UnderlyingSecurityID	30309		MktDef
ImpldMktInd	ImpliedMarketIndicator	1144		SecDef, SecDefUpd
InstrAttrib	InstrumentAttributes	Group	Y	SecDef, SecDefUpd
Instrmt	Instrument	Group	N	SecDef, SecDefUpd
InstrmtExt	InstrumentExtension	Group	N	SecDef, SecDefUpd
InstrmtScope	InstrumentScopes	Group	Y	MktDef
Leg	InstrmtLegGrp	Group	Y	SecDef, SecDefUpd
LowExerPxOptnInd	LowExercisePriceOptionIndicator	2574		SecDef,SecDefUpd
MarketSegmentDesc	MarketSegmentDesc	1396		MktDef
MarketSegmentDesc	MarketSegmentDesc	1396		MktDef
MatDt	MaturityDate	541		SecDef, SecDefUpd
MDBkTyp	MDBookType	1021		MktDef
MDFeedTyp	MDFeedType	1022		MktDef
MDFeedTyps	Feeds	Group	Y	MktDef
MDRcvryTmIntvl	MDRecoveryTimeInterval	2565		MktDef
MinBidSz	MinBidSize	647		MktDef, SecDef, SecDefUpd
MinOfrSz	MinOfferSize	648		MktDef, SecDef, SecDefUpd
MinPxIncr	MinPriceIncrement	969		SecDef, SecDefUpd
MinPxIncrClr	MinPriceIncrementClearing	28888		SecDef, SecDefUpd
MinPxIncrAmt	MinPriceIncrementAmount	1146		SecDef, SecDefUpd
MktDepth	MarketDepth	264		MktDef
MktDepthTmIntvl	MarketDepthTimeInterval	2563		MktDef
MktID	MarketID	1301		MktDef
MktImbalInd	MarketImbalanceIndicator	28875		MktDef
MktSeg	MarketSegment	7703		MktDef
MktSegGrp	MarketSegmentGrp	Group	Y	SecDef, SecDefUpd
MktSegID	MarketSegmentID	1300		MktDef, SecDef, SecDefUpd, SecStat
MktSegStat	MarketSegmentStatus	2542		MktDef
MktSegSubTyp	MarketSegmentSubType	2544		MktDef
MktSegTyp	MarketSegmentType	2543		MktDef
MlegModel	MultilegModel	1377		SecDef, SecDefUpd
MMY	MaturityMonthYear	200		SecDef, SecDefUpd
MtchAlgo	MatchAlgorithm	1142		MktDef

FIXML Attribute	FIX Field Name	FIX Tag / Group	Repeating	FIX Messages
MtchRules	MatchRules	Group	Y	MktDef
MtchRuleProdCmplx	MatchRuleProductComplex	2569		MktDef
MtchTyp	MatchType	574		MktDef
Mult	ContractMultiplier	231		SecDef, SecDefUpd
NxtEfctvBizDt	NextEffectiveBusinessDate	28871		MktDef
Oper	InstrumentScopeOperator	1535		MktDef
OptAt	OptAttribute	206		SecDef, SecDefUpd
OrigStrkPx	OrigStrikePrice	2578		SecDef, SecDefUpd
OvrNiteIntRt	OvernightInterestRate	2590		SecStat
ParentMktSegmID	ParentMktSegmID	1325		MktDef
PartID	PartitionID	5948		MktDef
PostTrdAnon	PostTradeAnonymity	28876		SecDef, SecDefUpd
PrevClsPx	UnderlyingPrevClosePx	28843		MktDef
PriSetPx	PriorSettlPrice	734		SecDef, SecDefUpd, SecStat
ProdCmplx	ProductComplex	1227		SecDef, SecDefUpd
Pty	InstrumentParties	Group	Y	SecDef, SecDefUpd
PutCall	PutOrCall	201		SecDef, SecDefUpd
Px	LegPrice	566		SecDef, SecDefUpd
PxPrcsn	InstrumentPricePrecision	2576		SecDef,SecDefUpd
PxRngPctg	PriceRangePercentage	2554		MktDef
PxRngProdCmplx	PriceRangeProductComplex	2555		MktDef
PxRngRuleID	PriceRangeRuleID	2556		MktDef, SecDef, SecDefUpd
PxRngRules	PriceRangeRules	Group	Y	MktDef, SecDef, SecDefUpd
PxRngValu	PriceRangeValue	2553		MktDef
PxTyp	PriceType	423		SecDef, SecDefUpd
QtSideInd	QuoteSideIndicator	2559		MktDef
Qual	InstrumentPartyRoleQualifier	2378		SecDef, SecDefUpd
QuotSizeRules	QuoteSizeRules	Group	Y	MktDef, SecDef, SecDefUpd
R	InstrumentPartyRole	1051		SecDef, SecDefUpd
RatioQty	LegRatioQty	623		SecDef, SecDefUpd
RefTickTblID	RefTickTableID	1787		SecDef, SecDefUpd
RelatedMktSeg	RelatedMarketSegments	Group	Y	MktDef
RelIdClsPx	RelatedClosePrice	2589		SecStat
RemClndrDays	RemainingCalendarDays	28880		SecStat
RelNdxVal	RelatedIndexValue	28882		SecStat
RFR	RiskFreeRate	1190		SecStat
Rltnshp	MarketSegmentsRelationship	2547		MktDef
RlzdVarnc	RealizedVariance	2587		SecStat
SecTrdgRules	SecurityTradingRules	Group	N	SecDef, SecDefUpd
SecTyp	InstrumentScopeSecurityType	1547		MktDef
SecTyp	LegSecurityType	609		SecDef, SecDefUpd
SecTyp	SecurityType	167		SecDef, SecDefUpd, SecStat
SetPx	ClearingSettlPrice	2528		SecStat
SetPx	SettlPrice	730		SecStat
SettlBizDays	SettlBusinessDays	28878		MktDef

FIXML Attribute	FIX Field Name	FIX Tag / Group	Repeating	FIX Messages
SettlCcy	SettlCurrency	120		SecDef,SecDefUpd
SettlMeth	SettlMethod	1193		SecDef,SecDefUpd
SettlSubMeth	SettlSubMethod	2579		SecDef,SecDefUpd
Side	LegSide	624		SecDef, SecDefUpd
Src	InstrumentPartyIDSource	1050		SecDef, SecDefUpd
Src	LegSecurityIDSource	603		SecDef, SecDefUpd
Src	SecurityIDSource	22		SecDef, SecDefUpd, SecStat
Src	UnderlyingSecurityIDSource	30305		MktDef
StartPxRng	StartPriceRange	2551		MktDef
StartTickPxRng	StartTickPriceRange	1206		MktDef
Status	SecurityStatus	965		SecDef, SecDefUpd
StdVarnc	StandardVariance	2588		SecStat
StrkPx	StrikePrice	202		SecDef, SecDefUpd
StrkPxPrcsn	StrikePricePrecision	2577		SecDef,SecDefUpd
SubTyp	SecuritySubType	762		SecDef, SecDefUpd
SvcLctnID1	PrimaryServiceLocationID	2567		MktDef
SvcLctnID2	SecondaryServiceLocationID	2568		MktDef
SvcLctnSubID1	PrimaryServiceLocationSubID	28591		MktDef
SvcLctnSubID2	SecondaryServiceLocationSubID	28593		MktDef
Sym	LegSymbol	600		MktDef
Sym	MarketSegmentSymbol	7177		MktDef
Sym	Symbol	55		MktDef, SecDef, SecDefUpd
Sym	UnderlyingSymbol	30311		MktDef
TickIncr	TickIncrement	1208		MktDef
TickRuleID	TickRuleID	28887		MktDef
TickRuleProdCmplx	TickRuleProductComplex	2571		MktDef
TickRules	TickRules	Group	Y	MktDef
TotTrdgBizDays	TotalTradingBusinessDays	2585		SecStat
TrdgBizDays	TradingBusinessDays	2586		SecStat
TxnTm	TransactTime	60		SecDef, SecDefUpd
Typ	InstrAttribType	871		SecDef, SecDefUpd
Undly		Group	N	MktDef
UpdActn	SecurityUpdateAction	980		SecDefUpd
USFirmFlag	USFirmFlag	9543		MktDef
Val	InstrAttribValue	872		SecDef, SecDefUpd
ValMeth	ValuationMethod	1197		SecDef,SecDefUpd
VegaMult	VegaMultiplier	2583		SecStat
Vol	Volatility	1188		SecStat

15 Change log

This document has been through the following iterations:

No	Chapter, page	Date	Change
2.0	General	11-Apr-2013	Creation based on version 1.0.
2.0.1	11.3.1, pg. 97	21-Aug-2013	Added two new values for MDEntryType due to exchange wide FAST template alignments.
2.0.1	11.3.2, pg. 102	21-Aug-2013	Added two new values for MDEntryType due to exchange wide FAST template alignments.
2.0.1	11.3.9, pg. 113	22-Aug-2013	Format for *SecurityUpdateAction (980)* changed from enumeration to string.
2.0.1	2.4, pg. 9	22-Aug-2013	New interface version numbers.
2.0.1	11.1.6, pg. 81	22-Aug-2013	Explained why new values have been added to *MDReportEvent (28827)*.
2.0.1	11.1.2, pg. 78	26-Aug-2013	Specified minimum value for *PerformanceIndicator*.
2.0.1	11.3.1, pg. 97	18-Sep-2013	Added a footnote for two MDEntryType entries.
2.0.1	11.3.2, pg. 102	18-Sep-2013	Added a footnote for two MDEntryType entries.
2.0.1	11.2.2, pg. 87	20-Sep-2013	Corrected list of values for *ExerciseStyle (1194)*.
2.0.2	14.3, pg. 125	25-Sep-2013	Replaced field names mapping PrevSettlPx / PrevSettlPx by PriSetPx / PriorSettlPrice.
2.0.2	11.3.1, pg. 97	25-Sep-2013	Added one more possible value (Last Closing Price) for *TradeCondition (277)*.
2.0.2	11.3.2, pg. 102	25-Sep-2013	Changed sequence of possible values for *TradeCondition (277)*.
2.0.2	11.3.1, pg. 97	25-Sep-2013	Changed sequence of possible values for *MDEntryType (269)*.
2.0.2	11.3.2, pg. 102	25-Sep-2013	Changed sequence of possible values for *MDEntryType (269)*.
2.0.2	9.5.2, pg. 58	8-Oct-2013	A trade cannot be reversed partially but only completely.
2.0.2	11.3.1, pg. 97	6-Nov-2013	Moved a *TradeCondition (277)* entry to the bottom of the list.
2.0.2	11.3.2, pg. 102	6-Nov-2013	Moved a *TradeCondition (277)* entry to the bottom of the list.
2.0.2	2.4, pg. 9	6-Nov-2013	New interface version numbers.
2.0.2	11.4.1, pg. 116	6-Nov-2013	Mentioned the existence of the FIXML schema file provided by Eurex.
2.0.2	11.2.2, pg. 87	6-Nov-2013	Changed the web address for *SecuritySubType (762)*.
2.0.2	11.3.9, pg. 113	6-Nov-2013	Changed the web address for *SecuritySubType (762)*.
2.0.2	11.3.1, pg. 97	6-Nov-2013	Amended description for *TrdType (828)*.
2.0.2	11.3.2, pg. 102	6-Nov-2013	Amended description for *TrdType (828)*.
2.0.2	11.2.1, pg. 82	14-Nov-2013	Removed footnote for *MarketSegmentStatus (28599)*.
2.0.2	11.2.2, pg. 87	14-Nov-2013	Removed footnote for *SecurityStatus (965)*.
2.0.2	14.3, pg. 125	28-Nov-2013	Changed tag number from 28818 to 734 for PriSetPx.
2.1.0	2.4, pg. 9	21-Jan-2014	New interface version numbers.
2.1.0	5.2.3, pg. 23	21-Jan-2014	Changed packet header values following the introduction of release 2.1. Specified that values are changed, and not increased, with each new release.

No	Chapter, page	Date	Change
2.1.0	11.2.2, pg. 87	21-Jan-2014	Amended description for *PriorSettlPrice (734)*. Amended description and added one more value for *SecurityStatus (965)*.
2.1.0	5.8, pg. 26	21-Jan-2014	Completed the FAST set example with the latest *TradeCondition (277)* entries.
2.1.0	9.6, pg. 59	21-Jan-2014	Added a new paragraph following the introduction of new value (Volume Only) for *TradeCondition (277)* and added paragraph about market data feed restart.
2.1.0	9.6.2, pg. 61	21-Jan-2014	Amended use case "Direct match of complex instruments" following the introduction of new value (Volume Only) for *TradeCondition (277)*.
2.1.0	9.6.4, pg. 63	21-Jan-2014	Amended use case "Complex versus simple/complex instruments" following the introduction of new value (Volume Only) for *TradeCondition (277)*.
2.1.0	9.7, pg. 67	21-Jan-2014	Amended second paragraph; a direct match of complex instruments increases now the total traded volume of the simple instruments involved in the trade.
2.1.0	9.8.4, pg. 71	21-Jan-2014	Added information about Trade Statistics on the incremental feed.
2.1.0	11.3.2, pg. 102	21-Jan-2014	Added new value (Volume Only) for *TradeCondition (277)*.
2.5.0	I, pg. 5	22-Jan-2014	Added IPS acronym.
2.5.0	2.4, pg. 9	22-Jan-2014	New interface version numbers.
2.5.0	2.6, pg. 10	22-Jan-2014	Removed reference to the "Eurex Market and Reference Data Manual, Delta" document.
2.5.0	4.3.2, pg. 19	22-Jan-2014	Added new Top Of Book Implied message.
2.5.0	5.2.3, pg. 23	22-Jan-2014	Changed packet header values following the introduction of release 2.5.
2.5.0	7.1, pg. 33	22-Jan-2014	Amended paragraph related to duplicate packet detection.
2.5.0	9.2.10, pg. 46	22-Jan-2014	Added Use Case for product pool concept.
2.5.0	11.2.1, pg. 82	22-Jan-2014	Added new values for *TickRuleProductComplex (28823)*, *MatchRuleProductComplex (28824)*, *FlexProductEligibilityComplex (28831)* and *PriceRangeProductComplex (28810)*. Added new fields *MarketSegmentPool (28847)* and *MarketSegmentPoolType (28848)*. Replaced field *RefMarketSegmentID (28700)* by new fields *NoRelatedMarketSegments (28844)*, *RelatedMarketSegmentID (28845)* and *MarketSegmentRelationship (28846)*. Completed description for *MarketSegmentID (1300)*. Fields *MarketSegmentSymbol (7177)*, *ParentMktSegmID (1325)* and *Currency (15)* have been made optional. Following the introduction of the product pool concept, the field *MarketSegmentSymbol (7177)* has been made optional
2.5.0	11.2.2, pg. 87	22-Jan-2014	Added new values for *ProductComplex (1227)*, completed description for *SecurityDesc (107)* and added new field *LegSymbol (600)*.
2.5.0	11.3.1, pg. 97	22-Jan-2014	Added new values for *ProductComplex (1227)* and added two new fields: *MDBookType (1021)* and *MDSubBookType (1173)*.

No	Chapter, page	Date	Change
2.5.0	11.3.5, pg. 106	22-Jan-2014	Added new values for *InstrumentScopeProductComplex (1544)*.
2.5.0	11.3.9, pg. 113	22-Jan-2014	Added new values for *ProductComplex (1227)*, added new field *LegSymbol (600)* and completed description for *SecurityDesc (107)*.
2.5.0	11.3.3, pg. 105	22-Jan-2014	Added new Top Of Book Implied message.
2.5.0	13, pg. 119	22-Jan-2014	Specified that two FAST templates have been merged into one file.
2.5.0	14.3, pg. 125	22-Jan-2014	Updated tables with new fields and obsolete fields.
2.5.0		22-Jan-2014	Added use case for Top of Book Implied message
2.5.0		22-Jan-2014	FAST templates for snapshot and incremental feeds have been merged into one single file.
			To Reconcile (comes from 2.1)
2.1.0	11.3.1, pg. 97	21-Jan-2014	Amended description for *TrdType (828)*, removed list of values that are not used for the message and removed footnote.
2.1.0	11.3.2, pg. 102	21-Jan-2014	Added new value (Volume Only) for *TradeCondition (277)*. Amended description for *TrdType (828)*, removed list of values that are not used for the message and removed footnote.
2.1.1	11.2.4, pg. 93	6-Jun-2014	VarianceFuturesStatus sent even when Variance Futures are not traded at T7.
2.5.0		24-July-2014	Numerous updates for the release 2.5.0.
2.5.1		13-Aug-2014	Minor corrections to document.
2.5.2	9.3, pg. 47	22-Aug-2014	Top of Book aspects clarified.
2.5.3	11.2.1, pg. 82	3-Sep-2014	ParentMktSegmID field updated.
2.5.4	11.3.3, pg.105	8-Sep-2014	Removed last sentence from the description of the QuoteCondition field.
2.5.5	3.4, pg. 16	25-Sep-2014	Added paragraph concerning the availability of the Eurex reference data interface during infrastructure maintenance.
2.5.5	11.3.1, pg. 97	25-Sep-2014	Added description for MDEntryType="B".
2.5.5	5.2.3, pg. 23	25-Sep-2014	Update TopOfBookImplied template IDs.
2.5.5	11.2.1, pg. 82	29-Sep-2014	Update instrument scope security fields.
2.5.6	5.2.3, pg. 23	1-Oct-2014	Update packet header IDs.
2.5.7	11.3.1, pg. 97	7-Oct-2014	MDBookType optional in depth snapshot message.
	11.2.1, pg. 82	7-Oct-2014	ParentMktSegmID field rollback.
	14.3, pg. 125	7-Oct-2014	ParentMktSegmID field rollback.
2.5.8	11.3.1, pg. 97	9-Oct-2014	TradingSessionID description updated.
	11.2.2, pg. 87	9-Oct-2014	SecuritySubType description updated.
	11.3.9, pg. 113	9-Oct-2014	SecuritySubType description updated.
	11.3.2, pg. 102	9-Oct-2014	Correct order of RequestTime/AggressorTimestamp fields to match xml template.
2.5.9	11.2.1, pg. 82	29-Oct-2014	Correct typo in field name (now MarketSegment**s**Relationship). Correct MarketSegmentPoolType data type.
	11.3.3, pg.105	29-Oct-2014	Correct MDUpdateAction description and QuoteCondition data type.
	11.3.1, pg.97	29-Oct-2014	Correct QuoteCondition data type.

No	Chapter, page	Date	Change
	11.2.4, pg.93	29-Oct-2014	Correct OvernightInterestRate description.
	4.3.2, pg.19, 9.3, pg.47, 9.3.2, pg.50, 11.3.3, pg.105	30-Oct-2014	Improved descriptions for Top Of Book Implied message.
2.5.10	9.3.2, pg. 50	20-Nov-2014	Correct errors in example.
2.5.11	9.3, pg. 47	03-Dec-2014	Small clarification to saving of orderbook depth.
3.0	5.8, pg. 26	1-Mar-2015	Add reference to System Documentation.
	11, pg. 78	1-Mar-2015	Correct reference to external documentation.
	11.1.6, pg. 81	1-Mar-2015	Remove unused MDReportEvents
	11.3.1, pg. 97	1-Mar-2015	Remove unused MDEntryTypes
	11.3.2, pg. 102	1-Mar-2015	Remove unused MDEntryTypes
	5.2.3, pg. 23	5-May-2015	Update template IDs for packet header
	9.6, pg. 59	5-May-2015	Introduce RestingCxlQty
	11.3.2, pg.102	5-May-2015	Introduce RestingCxlQty
	11.2.2, pg. 87	10-Jun-2015	Make EventDate mandatory
3.0	11.2.1, pg. 82 14.3, pg. 125	28-Sep-2015	Removed field *RefMarketSegmentID (28700)* from document. Updated type of *RelatedMarketSegmentID (28845)* to uInt32.
	14.3, pg. 125	09-Oct-2015	Added group *RelatedMktSeg*, *RelatedMarketSegmentID (28845)* and *MarketSegmentsRelationship (28846)*
4.0	11.3.1, pg. 97	11-Apr-2016	Added new fields and new enumeration values for TES.
4.0	11.3.2, pg. 102	11-Apr-2016	Added new fields and new enumeration values for TES.
	11.3.4, pg. 106	11-Apr-2016	Added new fields and new enumeration values for TES.
	11.3.5, pg. 106	11-Apr-2016	Added new fields and new enumeration values for TES.
	11.2.1, pg. 82	11-Apr-2016	Added new value for *FlexProductEligibilityComplex (2561)*. Replaced *TradeDate (75)* by *EffectiveBusinessDate (2400)*. Added *NextEffectiveBusinessDate(28871)*.
	11.2.2, pg. 87	11-Apr-2016	Added new value for *ProductComplex (1227)* Changed *OptAttribute (206)* and *ContractGenerationNumber (25034)* from string to uInt32.
4.0	11.2.2, pg. 87	12-Apr-2016	Added new field *LegSecurityType (609)*. Field *LegSymbol (600)* changed from required to optional. Removed *PrevAdjustedOpenInterest (28816)* and *PrevUnadjustedOpenInterest (28817)*. Field *BlockTradeEligibilityIndicator (2575)* changed from required to optional and description changed. Fields *TransactTime (60)* and *SecurityDesc (107)* moved from *ComplexInstrumentDescriptorGroup* to the top level structure. Description adapted for flexible instruments. Added *CFICode (461)*. Introduced new value for *SecurityAltIDSource (456)* and changed data type from uInt64 to string.

No	Chapter, page	Date	Change
	11.3.9, pg. 113	12-Apr-2016	Added new field *LegSecurityType (609)*.
	5.2.3, pg. 23	12-Apr-2016	Add new template id for FlexibleInstrumentUpdate.
	4.3.2, pg. 19	12-Apr-2016	Added new Flexible Instrument Update message.
	11.3.10, pg. 115	12-Apr-2016	Added new Flexible Instrument Update message.
4.0	11.4.1, pg. 116	13-Apr-2016	One RDF file per market in release 4. Removed footnote.
	1, pg. 5	14-Apr-2016	Added entry for flexible instruments.
	9.2.2, pg. 43	14-Apr-2016	Renamed *MDCount* (and all further references) consistently into *MDReportCount (2536)* and changed description. Renamed *TotNoInstruments* (and all further references) consistently into *TotNoInstrumentReports (2538)* and changed description. Renamed *TotNoMarketSegments* (and all further references) consistently into *TotNoMarketSegmentReports (2537)* and changed description. Added footnote.
	11.1.6, pg. 81	14-Apr-2016	Added *MarketID (1301)* and *SenderSubID (50)*.
4.0	5.2.3, pg. 23	17-May-2016	Updated template IDs for packet headers
4.0	11.1.6, pg. 81	23-May-2016	Updated some FIX names and tags to latest standard.
	11.2.1, pg. 82	23-May-2016	Updated some FIX names and tags to latest standard.
	11.2.2, pg. 87	23-May-2016	Updated some FIX names and tags to latest standard.
	11.2.4, pg. 93	23-May-2016	Updated some FIX names and tags to latest standard.
	11.3.1, pg. 97	23-May-2016	Updated some FIX names and tags to latest standard.
	11.3.2, pg. 102	23-May-2016	Updated some FIX names and tags to latest standard.
	11.3.4, pg. 106	23-May-2016	Updated some FIX names and tags to latest standard.
	11.3.5, pg. 106	23-May-2016	Updated some FIX names and tags to latest standard.
	11.3.6, pg. 109	23-May-2016	Updated some FIX names and tags to latest standard.
4.0	9.2.2, pg. 43	23-May-2016	Chapter updated according to new FIX names and tags for *Market Data Report* message.
	9.2.4, pg. 44	23-May-2016	Chapter updated according to new FIX names and tags for *Market Data Report* message.
	9.2.9, pg. 46	23-May-2016	Chapter updated according to new FIX names and tags for *Market Data Report* message.
	4.2.1, pg. 17	23-May-2016	Chapter updated according to new FIX names and tags for *Product state change* message.
	4.2.2, pg. 18	23-May-2016	Chapter updated according to new FIX names and tags for *Mass instrument state change* message.
	7.2, pg. 33	23-May-2016	Chapter updated according to new FIX names and tags for *Product snapshot message*.
	12, pg. 118	23-May-2016	Chapter updated according to new FIX names and tags for *Product Snapshot message*.
	9.5, pg. 57	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.
	9.6, pg. 59	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.
	9.6.3, pg. 62	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.

No	Chapter, page	Date	Change
	9.6.4, pg. 63	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.
	9.6.5, pg. 64	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.
	9.7, pg. 67	23-May-2016	Chapter updated according to new FIX names and tags for *Depth Incremental* message.
4.0	11.2.5, pg. 95	1-June-2016	Added description for new message.
4.0	11.2.1, pg. 82	1-June-2016	Added new values for *InstrumentScopeSecurityType (1547)*. Changed *TickRuleProductComplex (2571)* to optional. Added *TickRuleID (28887)*.
4.0	11.2.4, pg. 93	1-June-2016	Added new field *SecurityType (167)*.
4.0	9.2.12, pg. 46	16-June-2016	New use case for total return futures.
4.0	14.3, pg. 125	17-June-2016	FIXML mapping table updated for 4.0.
4.0	2.4, pg. 9	19-June-2016	New Web Documentation links for release 4.0 New interface version numbers for release 4.0.
4.0	9.8.3, pg. 70	3-Aug-2016	Change all products to effected products
4.0	9.6.6, pg. 65	3-Aug-2016	Add use cases for self match prevention
4.0	4.2.1, pg. 17	3-Aug-2016	Add TES product states
4.0	9.4, pg. 57	3-Aug-2016	Add reporting of TES trades
4.0	11.3.9, pg. 113	3-Aug-2016	Add Flexible Instrument
4.0	2.3.1, pg. 8	8-Aug-2016	One RDI per marketID
4.0	9.2.6, pg. 45	8-Aug-2016	One RDI per marketID
4.0	9.6, pg. 59	8-Aug-2016	Off-Book trades are now disseminated
4.0	12, pg. 118	9-Aug-2016	Remove reference to very old release. Added reference data snapshot and incremental channels addresses for market EEX.
4.0	11.3.2, pg. 102	31-Aug-2016	Changed description for *MDEntryType (269)* "B" = Trade Volume for EMDI and *MDOriginType (1024)* "Off-Book".
4.0	9.3, pg. 48	31-Aug-2016	Updated bullet point ”During an auction...” and removed all references to implicit auction related orderbook rules.
4.0	12, pg. 118	31-Aug-2016	Updated reference data multicast addresses for EEX Simulation and EEX Production.
4.0	4.8, pg. 21	8-Sep-2016	Replaced "End-Of-Day" with "Post-End-Of-Day" in paragraph "Market data messages are sent from ...".
4.0	11.2.2, pg. 87 11.3.10, pg, 115	8-Sep-2016	Added field *Symbol (55)*.
4.0	11.3.1, pg. 97	8-Sep-2016	Updated description for fields *MDOriginType (1024)*, *TradingSessionID (336)*, *TradingSessionID (336)*, *TESTradSesStatus (25044)*, *SecurityTradingEvent (1174)* and *NonDisclosedTradeVolume (28873)*.
4.0	11.3.2, pg. 102	8-Sep-2016	Updated description for field *MDOriginType (1024)*.
4.0	11.3.1, pg. 97, 11.3.5, pg. 106, 11.3.6, pg. 109	12-Oct-2016	Added new value 9 = *Suspended* for *TESSecurityStatus (25045)*.

No	Chapter, page	Date	Change
4.0	4.2.1, pg. 17	12-Oct-2016	Added Eurex TES Activity Status mapping for *TESTradSesStatus (25044)*.
4.0	9.4, pg. 57	13-Oct-2016	Changed *MDOriginType (1024)* 0 = Off-Book into 1 = Off-Book and 1 = Book into 0 = Book.
4.0	9.4, pg. 57	21-Oct-2016	Added example for an Off-Book Volume Only entry.
4.0	pg. 6, 97, 102 ff.	21-Oct-2016	Minor syntactical corrections.
4.0	11.3.1, pg. 97	21-Oct-2016	Added missing valid value 6 = After-Hours.
4.0	9.6, pg. 59	3-Nov-2016	The field *MDEntryID (278)* is a unique id on product level and origin type for each business day. It is not set for future legs of volatility option strategies.