BSE Exchange's New Trading Architecture

BSE Market Data Interfaces

Manual

Version 1.4.2

Date: March 28, 2024

Strictly for private circulation only. This document must not be circulated to other users without prior permission of BSE.

BSE - PUBLIC

Contents

1 List of abbreviations 8
2 Introduction 9
2.1 Purpose of this document 10
2.2 Main audience 10
2.3 Data feeds 10
2.3.1 Reference data interface 11
2.3.2 Market data interfaces 12
2.4 Interface version number 12
2.5 Further reading matter for this topic 12
2.6 How to read this document 13
3 Differences between the interfaces 15
3.1 Distribution sequence for BSE EMDI 16
3.2 Distribution sequence for MDI / RDI 16
3.3 Choosing between the EMDI and the MDI 17
3.4 Choosing between the BSE RDI and BSE RDF 18
4 Overview of the BSE Public Interfaces 20
4.1 Infrastructure requirements 20
4.2 Trading states 20
4.2.1 Product State Changes 20
4.2.2 Instrument State Changes 21
4.3 Overview of the various message types 23
4.3.1 RDI 23
4.3.2 EMDI/MDI 23
4.4 What is not included in these interfaces 24
4.5 FIX over FAST 25
4.6 Freedom of choice 25
4.7 Testing 25
4.8 Hours of operation/availability of messages 26
5 FIX/FAST-Implementation 27
5.1 Structure of Messages 27

BSE - PUBLIC

5.2 FAST terminology 28
5.2.1 FAST reset message 28
5.2.2 Presence Map (PMAP) 28
5.2.3 Template ID (TID) 29
5.2.4 Dictionaries 30
5.2.5 Stop bit encoding 30
5.2.6 FAST operators 30
5.3 Decoding the FAST-message 30
5.4 Transfer decoding 31
5.5 Composing the Actual FIX-Message 31
5.6 New features in FAST version 1.2 31
5.7 Data types 32
5.8 FAST version 1.1 compatible templates 32
6 Description of a typical trading day 34
6.1 Start of day operation 34
6.2 Receiving reference data via RDI at start of day 34
6.3 Receiving reference data file (RDF) at start of day 36
6.4 Build the initial order book 36
6.4.1 Build the initial order book with the EMDI 36
6.4.2 Build the initial order book with the MDI 37
6.5 Update the order book 38
6.5.1 Update the order book with the EMDI 38
6.5.2 Update the order book with the MDI 38
7 Recovery 40
7.1 Detecting duplicates and gaps by means of the packet header 40
7.2 How to recover data via the respective other service (A or B) 41
7.3 Delayed packets 42
7.4 Missing packets 43
7.4.1 Recovery (EMDI) 44
7.4.2 Recovery (MDI) 46
8 Various time stamps in BSE and how to use them 48
8.1 Time stamps (EMDI) 48

BSE - PUBLIC

8.2 Time stamps (MDI) ... 50
9 Important topics with use cases and examples ... 51
9.1 Reference data messages ... 51
9.2 General reference data rules ... 53
9.2.1 General structure of the snapshot cycle ... 53
9.2.2 Counters as part of the market data report message ... 54
9.2.3 Use case 1: Reference data at the start of the reference data service ... 55
9.2.4 Use case 2: Reference data after intraday addition of complex instruments ... 55
9.2.5 Use case 3: Reference data after intraday deletion of a complex instrument ... 56
9.2.6 Use case 4: Reference data on the next business day ... 57
9.2.7 Use case 5: Failover or restart of RDI ... 57
9.2.8 Use case 6: Chronological order of messages for complex instrument creation ... 58
9.3 General order book rules and mechanics ... 58
9.3.1 Determination of the price sources ... 62
9.3.2 New price level ... 63
9.3.3 Change of a price level ... 64
9.3.4 Overlay ... 64
9.3.5 Deletion of a price level ... 65
9.3.6 Deletion of multiple price levels from a given price level onwards ... 66
9.3.7 Deletion of multiple price levels up to a given price level ... 66
9.4 Trade Volume Reporting (EMDI) ... 68
9.4.1 Use case 1: Direct match of simple instruments ... 68
9.4.2 Use case 2: Direct match of complex instruments ... 69
9.4.3 Use case 4: Complex versus simple/complex match ... 71
9.4.4 Use case 5: Opening auction ... 71
9.5 Trade Volume Reporting (MDI) ... 72
9.6 Failure of the market data feed/ matching engine ... 73
9.6.1 Normal processing ... 73
9.6.2 Market data feed fail-over (EMDI) ... 74
9.6.3 Market data feed fail-over (MDI) ... 75
9.6.4 Market data feed restart (EMDI) ... 76
9.6.5 Market data feed restart (MDI) ... 77

BSE - PUBLIC

9.6.6 Failure of the matching engine 77
9.7 Trading states for a sample business day 78
9.7.1 Start-Of-Day 78
9.7.2 Pre-Trading 79
9.7.3 Trading 79
9.7.4 Continuous Trading 79
9.7.5 Intraday Expiry 80
9.7.6 Closing 80
9.7.7 Post Closing 81
9.7.8 End-Of-Day 81
10 Fine tuning client applications 82
10.1 Buffer size 82
10.2 Packet and message processing 82
10.3 Application level 83
10.3.1 Discarding duplicate packets within the live-live environment 83
10.3.2 Order book processing 83
10.3.3 Optimal processing of desired products (EMDI) 84
11 Detailed data feed description and layout 85
11.1 Service messages 85
11.1.1 FAST reset message 85
11.1.2 Packet header (EMDI) 85
11.1.3 Packet header (MDI / RDI) 87
11.1.4 Functional beacon message 87
11.1.5 Technical heartbeat message 88
11.1.6 Market data report message 88
11.2 Reference data messages 90
11.2.1 Product snapshot message 90
11.2.2 Instrument snapshot message 96
11.2.3 Instrument incremental message 101
11.3 Market data messages 103
11.3.1 Depth snapshot message 103
11.3.2 Depth incremental message 108

BSE - PUBLIC

11.3.3 Product state change message 111
11.3.4 Mass instrument state change message 113
11.3.5 Instrument state change message 115
11.3.6 Complex instrument update message 116
11.3.7 Index change message 119
Data files 120
11.3.8 Reference data from file (BSE RDF) 120
11.3.9 File name format of the reference data files 120
11.3.10 Reference data file on the next business day 121
11.3.11 Reference data file after a failover or restart of BSE RDI 121
11.3.12 What receiving applications need to do 122
12 Multicast addresses 123
12.1 Reference data snapshot feed 123
12.2 Reference data incremental feed 123
13 FAST templates 125
14 Appendix 126
14.1 Example for a XML FAST template 126
14.2 Example for determination of the price source 127
14.2.1 Fully implied (example for 9.3.1, Determination of the price sources) 127
14.2.2 Fully outright on level 1 (example for 9.3.1, Determination of the price sources) 127
14.2.3 Partially implied (example for 9.3.1, Determination of the price sources) 128
14.2.4 Several fully implied orders at Best Market (example for 9.3, General order book rules and mechanics) 129
14.3 FIXML mapping table 131
14.4 Index Code and Mapping Table 134
15 Change log 135
15.1 Changes compared to version 1.3.3 135
15.2 Changes compared to version 1.3.4 135
15.3 Changes compared to version 1.3.5 136

BSE - PUBLIC

Part I

1 List of abbreviations

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

EMDI	Enhanced Market Data Interface
MDI	Market Data Interface
RDI	Reference Data Interface
RDF	Reference Data File
ETI	Enhanced Transaction Interface
FAST	FIX Adapted for STreaming (FAST Protocol) (FAST ProtocolSM). 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	Incremental and snapshots are delivered in the same channel.
Match event	Part of the matching event having a unique match price.
Out-of-band	Incremental 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.
Live-live concept	The concept whereby data is disseminated simultaneously via two separate channels called "Service A" and "Service B".
OTC	Over the Counter
PMAP	Presence Map
ToB	Top of Book

BSE - PUBLIC

2 Introduction

BSE offers public market and reference data via three interfaces as part of the BSE Exchange's new trading architecture. All three interfaces distribute information via UDP multicast; following FIX 5.0 SP2 semantics and are FAST 1.2[1] 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 message and rebuild the order book.

There are two types of **Market Data Interface:**

- The Enhanced **Market Data Interface (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 **Market Data Interface (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 EMDI. On-exchange trades are not reported individually but statistical information (daily high/low price, last trade price and quantity) is provided instead.

The EMDI and MDI provide the following information to the participant:

- Price level aggregated order book depth and on-exchange trade statistics.
- Product and instrument states.
- Information on newly created/deleted complex instruments.

Reference data is sent with:

- The Reference **Data Interface (RDI):** This interface provides reference data for instruments that are available for trading on the BSE Exchange's new trading architecture. 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 essential for any trading application. **The interface is currently unavailable and would be made available soon.**

- The **Reference Data File (RDF):** Reference data is delivered as a start-of-day file. The file is available on the BSE website. **The interface is currently unavailable and would be made available soon.**

[1]Fast Templates are provided as well.

BSE - PUBLIC

2.1 Purpose of this document

The purpose of this document is to provide guidance for programmers during development of applications that read the BSE Market & Reference Data Interfaces.
It covers a complete reference for the three-multicast based public interfaces, describes the general business behavior and provides concepts for the implementation.

The most recent version is available at: www.bseindia.com/nta.aspx

2.2 Main audience

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

Prior knowledge of developing for a capital 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 behavior 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 incremental as can be seen in Figure 1. The two public market data interfaces, the EMDI for a high bandwidth network and the MDI for a low bandwidth network, disseminate information across the BSE network to the receiving application. The RDI is considered for participants with a high bandwidth network while the RDF should be used if only a low bandwidth network is available.

BSE - PUBLIC

Figure 1: Reference data and Market data interfaces

2.3.1 Reference data interface

Public reference data delivered by RDI contains the technical configuration, e.g. Multicast address and port combinations for both market data interfaces for all products and instruments.
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 incremental.
The **reference data incremental feed** delivers reference information about intraday created and deleted complex instruments.

BSE - PUBLIC

2.3.2 Market data interfaces

The EMDI and the MDI disseminate public market data information in the form of incremental (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 to 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 also listed within the FAST XML templates. This manual relates to the following interface version numbers:

- EMDI: 500.002.860-500002860.20
- MDI: 500.002.860-500002860.23
- RDI:

The version numbers for the interfaces are available at the beginning of the FAST XML files.

2.5 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 specifications plus the extension Proposal.
  www.fixprotocol.org > fastspec
- **FIX specification documents:** FIX-messages and FIX-tags
  www.fixprotocol.org > Specifications

BSE - PUBLIC

- **FIX-Tags:** Specifies all FIX-Tags
  www.fixprotocol.org > FIXimate3.0

2.6 How to read this document

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

For example, section 7.4.2, Recovery (MDI), refers to the "netted" MDI only. Participants who are interested in the "un-netted" 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: Description 0: Beacon
<Group Name> (Optional) group starts				
<Sequence Name> Sequence start				
….	…..	….	….	….
….	….	….	….	….
<Group Name> (Optional) group starts				
<Sequence Name> Sequence start				

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.

BSE - PUBLIC

– 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 its "valid values" in table format for this particular message.
– Group Name, Sequence Name: 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 "incremental" and "snapshots" are used in various contexts. Within this document "incremental" 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.

BSE - PUBLIC

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 EMDI. The following diagram illustrates the available feeds for the three multicast based public interfaces:

Figure 2: Overview of the three interfaces

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

The snapshot and incremental messages for the **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 20. 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.

BSE - PUBLIC

In contrast to the **EMDI**, the snapshot and incremental messages for the **MDI** are sent on one feed only (in-band), therefore there is no need to synchronize both messages. The feed is also divided into several channels grouped on product basis.
The snapshot and incremental feed for the **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 are the same. This is called "live-live" concept. Product groups are distributed across several partitions on the BSE backend side. Service A and Service B cannot be published at exactly the same time.

3.1 Distribution sequence for BSE 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, in reference data file 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.

The time difference between publication of Service A and B is currently not known; lab testing indicates 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 MDI / 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 MDI and 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 EMDI and the 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 **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 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. Currently, BSE plans to provide market data for benchmark products with a netting interval of 0.47 sec and depth of 5. For all other products, a netting interval of 0.67 sec is envisaged. This interface has less price levels than the 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 EMDI and the MDI:

Area	EMDI	MDI
In-band/Out-of- band delivery	Incremental 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	Incremental 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.

BSE - PUBLIC

	needed only for start-up/recovery.	
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 13.	Messages on the combined market data incremental + snapshot feed have one sequence number range per product as shown in Table 14
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.
Packet header	A Performance Indicator[4] is provided for incremental within the Packet Header as shown in figure 21	A Performance Indicator does not exist as shown in figure 22
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 EMDI and the MDI

3.4 Choosing between the BSE RDI and BSE RDF

The Reference data is provided via the **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 instruments are added and deleted. 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 messages via the market data incremental feed of the **EMDI** and the market data feed of the **MDI**. During normal operations participants do not need to listen to the incremental feed of the **RDI**, because the complex instrument update[6] can be received on the market data feed. Furthermore, market data for new complex instruments is never provided ahead of their reference data on **EMDI** or **MDI** but may come ahead of its publication via RDI.

BSE - PUBLIC

Participants have the choice between the two different reference data sources. However, it is assumed that bandwidth conscious users will use BSE 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 difference between the BSE reference data messages and the reference data from File:

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

[4] Time between arrival of an incoming order/quote transaction on the BSE 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)

BSE - PUBLIC

4 Overview of the BSE Public Interfaces

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

4.1 Infrastructure requirements

The BSE market and reference data interfaces disseminate market and reference data over the BSE 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 EMDI and the MDI market data feeds. Trading state information is not communicated over the Enhanced Transaction Interface (ETI) or FIX interface.

The EMDI and the MDI market data feeds follow the FIX protocol for the publication of trading state information. The BSE product and instrument states are displayed by these interfaces as shown in the following tables.
Section 9.8, Trading states for a sample business day illustrates state messages for a typical business day. The hours of operations for the BSE system are 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			
Product State	FIX TradingSessionID (336)	FIX TradingSessionSubID (625)	FIX TradeSesStatus (340)
Start of Day	3 = Morning	9 = 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
Post- Closing	5 = Evening	6 = Post-Closing	2 = Open
End of Day	5 = Evening	9 = Quiescent	3 = Closed

BSE - PUBLIC

Halt	1 = Day	9 = Quiescent	1 = Halted
Holiday	7 = Holiday	9 = Quiescent	3 = Closed

Table 5: Product states

A Halt state is additionally indicated by the FIX field TradSesStatus (340) containing the value 1 = Halted.

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.

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

Table 6: Instrument states

The field FastMarketIndicator (28828) 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

BSE - PUBLIC

message is sent when a product is set to Fast Market (or back) without a change in the instrument states.

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

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

BSE - PUBLIC

4.3 Overview of the various message types

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

4.3.1 RDI

Service messages:

- Technical heartbeat message is sent out periodically by the BSE 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].

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 instruments that existed at start-of-day.

- Instrument incremental message used with intraday added or deleted complex instruments. Identical messages are also sent on the market data incremental feed of the BSE EMDI as well as on the market data feed of the BSE MDI.

4.3.2 EMDI/MDI

Service messages:

- Technical heartbeat message is sent out on all multicast addresses of the EMDI/MDI. The description is the same as for RDI.

- Functional beacon message (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 MDI because the snapshots act as a functional beacon.

[8] It is used to keep spanning tree alive

BSE - PUBLIC

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.

- Product state change message is used to publish the state of the BSE 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 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.

- Complex instrument update message is used to publish new or deleted complex instruments. This message is sent via the market data incremental feed of the BSE EMDI and the market data feed of the MDI. A message is sent for each newly created complex instrument.

- Index Change Message is used to publish the indices current values and also day's high, low open and close values. This message is sent via the market data incremental feed of the BSE EMDI and the market data feed of the MDI. The message is sent periodically at a defined interval.

- 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.

BSE - PUBLIC

- Market Supervision News is not provided. This information is available via the BSE ETI in recoverable form.

- In case of derivatives contracts, the prices for underlying are not provided. These prices will be available in multicast form in the multicast stream of the respective segment.

- 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 EMDI, MDI and RDI. Alternatively participants can use their own FAST decoder implementation.

4.6 Freedom of choice

BSE does not need to provide any software for accessing the services offered. The BSE 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 BSE service fail-over scenarios.

BSE - PUBLIC

4.8 Hours of operation/availability of messages

The product state “Pre-Trading” (TradingSessionSubID (625) = 1) for many of the BSE products begins at 6:30 IST.

“Post-Trading” (TradingSessionSubID (625) = 5) for some BSE products lasts until 22:30 CET.

- BSE is available from approximately 6:30 IST. It is recommended to start applications between 6:30 IST and 7:00 IST.
- Market data messages are sent from the time a product changes to the state “Start-Of-Day” and stops when it changes to the state “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 pro- duct 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 7: Availability of messages per instrument state

BSE - PUBLIC

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 datagram 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 (Message1).
- Possibly followed by one or more messages ($Message_2$ -$Message_n$).

BSE - PUBLIC

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 datagram.

5.2 FAST terminology

5.2.1 FAST reset message

The BSE 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.

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

BSE - PUBLIC

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 BSE interfaces:

Message	TID RDI	TID EMDI	TID MDI
Functional Beacon		109	
Packet header for RDI / EMDI / MDI	114	116	122
FAST Reset Message	120	120	120
MarketDataReport	125		
ProductSnapshot	122		
InstrumentSnapshot	123		
InstrumentIncremental	121		
ComplexInstrumentUpdate		124	105
DepthSnapshot		93	101
DepthIncremental		94	102
ProductStateChange		97	108
MassInstrumentStateChange		99	104
InstrumentStateChange		98	103
IndexChange		50	51

Table 8: Template identifiers for RDI/EMDI/MDI

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

Example: The TID=116 indicates the packet header for MDI in the current release. In the next release the TID for the packet header would increase to the next available integer, i.e. TID=117.

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 is Byte Vectors as they are used in the packet header of EMDI/MDI/RDI.

5.2.6 FAST operators

Field operators are used to remove redundancies in the data values. Message templates are the metadata 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 BSE 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. Mistype), FIX tags (e.g. 35) And FAST operators (e.g. Increment). The FAST templates also contain information about repeating groups (sequences).

BSE - PUBLIC

A typical example for a XML FAST template with a repeating group is shown in figure 23 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 participant's 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.

BSE - PUBLIC

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 BSE market data interface. A time stamp is an integer that represents a number of time units since an epoch.

5.7 Data types

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

- Decimal
- Length
- String
- uInt32/uInt64
- 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 BSE. 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 in the FIX tables, chapter 11, Detailed data feed description and layout.

- **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 in the FIX tables, chapter 11; Detailed data feed description and layout.

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

BSE - PUBLIC

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="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="AN" id="PreviousClose"/>
        </set>
```

The wire format of the values U, R, AX, AY, AJ, AW, k is 1, 2, 4, 8, 16, 32, 64 and 128 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 is 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”.

BSE - PUBLIC

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 BSE 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 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 to receive instrument details.

BSE - PUBLIC

- Instrument incremental to receive intraday updates.

Port information and multicast addresses for the reference data feeds as well as the address ranges for market data are 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 instrument incremental messages.
If an application starts listening to the reference data messages early enough, there are no instrument incremental 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:

- MDCount, 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 - MDCount.

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 incremental before the start of the actual trading.

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

BSE - PUBLIC

Note: Participants interested in complex instruments should use the complex instrument update messages. These are published on the market data incremental feed of the EMDI as well as on the market data feed of the 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 instruments are added and deleted.

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.

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 EMDI

For each instrument within the desired products do the following:

BSE - PUBLIC

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

Figure 6: EMDI initial order book

6.4.2 Build the initial order book with the MDI

The following sequence is recommended for the MDI:

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

Figure 7: MDI initial order book

BSE - PUBLIC

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

> Note: MDI applications must process depth snapshots beside the depth incremental 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 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 as 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.

6.5.2 Update the order book with the MDI

As longs 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.

---

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

[12] only if the Refresh Indicator (1187) = Y

BSE - PUBLIC

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 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.

BSE - PUBLIC

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 may 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 EMDI or 8 Bytes for MDI of a datagram containing SenderCompId and PacketSeqNum as shown in figure 21, Structure of the packet header for EMDI and figure 22, Structure of the packet header for MDI and 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.

BSE - PUBLIC

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

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.

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

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

Time	Service A: MsgSeqNum	Message	Time	Service B: MsgSeqNum	Message
10:30:00	206	New 151@4	10:30:01	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 9: Recovery via Service B (live-live concept)

BSE - PUBLIC

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 discards the second (identical) message.

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

- The order book needs to be recreated by using the **depth snapshot** messages in conjunction with the **depth incremental** messages. This procedure is similar to the Start Up procedure. Please see section 6.4, Build the initial order book.

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 10: Packets arriving in correct sequence

In this example, messages arrive in the correct order. The message was not delayed between BSE 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, BSE 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 11: Delayed Packet 207

BSE - PUBLIC

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, recognizes that 207 is missing.

3. Start an appropriate timed operation to trigger the recovery actions if the out-of-sequence message 207 fail 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 12: Missing seqNum 207

The correct behavior 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.

BSE - PUBLIC

(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 (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, Overview of the three interfaces).

Incremental 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		depth incremental	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

BSE - PUBLIC

1001	B		depth incremental	A_2^{I}

Table 13: Snapshots and incremental within the EMDI

The appropriate recovery action for missing depth incremental is the same as the logic described in section 6.4.1; Build the initial order book with the 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.

- The corollary to the above is that an application may have missed a complex instrument update indicating a strategy deletion message. Deleted strategies are immediately removed from the snapshot feed, so having seen a complete cycle for a product and noting that no order book information was received for a strategy, the application should consult the reference data snapshot feed to confirm that it has been deleted.

- 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:

$P_1 I_1$ | $P_1 I_2$ | $P_1 I_3$ | $P_1 I_n$ | $P_2 I_1$ | $P_2 I_2$ | $P_2 I_3$ | $P_2 I_n$ | $P_3 I_1$ | $P_3 I_2$ | $P_3 I_3$ | $P_3 I_q$ | [...]

BSE - PUBLIC

with:
P_n : Product n
I_q : Simple 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 snap- shot 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 (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	depth incremental	$A_1^{S,i}$
6	A	depth incremental	$A_1^{S,i}$

BSE - PUBLIC

Lost	A	depth incremental	$A_1^{S,i}$
25	B	depth incremental	$A_2^{S,i}$
8	A	depth incremental	A_1^{S}
9	A	depth snapshot	A_1^{S}
10	B	depth snapshot	$A_2^{S,i}$
11	A	depth incremental	$A_1^{S,i}$
26	B	depth snapshot	$A_2^{S,i}$
27	B	depth snapshot	$A_2^{S,i}$

Table 14: Snapshots and incremental within the 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 incremental 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 MDI.

BSE - PUBLIC

8 Various time stamps in BSE and how to use them

This section provides a list of each time stamp field in the two BSE Market Data Interfaces; it describes the measurement point and explains what is being measured.

All time stamps are provided in UTC (nanoseconds since “Unix Epoch” (01.01.1970)). While the format is provided in nanoseconds the actual precision of time stamps can be in microseconds. In that case the last three digits of the time stamp field is “000”.

8.1 Time stamps (EMDI)

The following picture shows all time stamps on the BSE Matching Partition for requests/responses for orders and execution messages:

Figure 8: Time stamps and their measurement points

BSE - PUBLIC

Message	Field Name	Time stamp	Description of time stamp
Packet header	PerformanceIndicator	t_3 -t_0	Time between the arrival of an incoming order transaction on the BSE matching engine and send time of the corresponding outgoing market data. The PerformanceIndicator is sent for Incremental only.
	Sending Time	t_3	Time, the BSE Market Data is written onto the socket for the fastest Service (A or B). For more information please see "distribution sequence". The time stamp is the same for Service A and Service B even though packets on both services are not sent out at the same time.
Depth incremental	MDEntryTime[14]	t_2	Two possibilities: • In case of an order: Time of the last order book update. • In case of a trade: Match time.
Depth incremental	AggressorTimeStamp[15]	t_0	Entry time of the incoming order that triggered the trade. This time stamp is only available in case of a trade (MDEntryType=2). The AggressorTimeStamp is empty if • The trade resulted from an auction.
Depth snapshot	LastUpdateTime	t_2	Time of the last order book update.
Instrument incremental **Complex instrument update**	TransactTime	t_2	Creation time of complex instrument. This field is empty for deletions of complex instruments.
	TransactTime	t_2	Time when request was

BSE - PUBLIC

Mass instrument state change **Product state change** **Instrument state change**			processed by the matcher.

Table 15: Meaning of the time stamps

Further information about time stamps are available in the ETI manual, section "Timestamps"

8.2 Time stamps (MDI)

The field PerformanceIndicator in the packet header message is not available for the BSE MDI. Also the field AggressorTimeStamp (28820) will not have any value. All other fields are the same as for the BSE EMDI.

The field LastUpdateTime in the depth snapshot message contains the time at which the last update was applied to the order book. In the special situation that several orders entered in the middle of a netting interval cancel each other out, this field shows the time stamp of the last order entry even though the net result is that there is no actual change.

An example of when this would happen is the creation and subsequent deletion of an order within the netting interval.

[14] This field maps to ExecID (17) in BSE ETI order / quote responses and notifications as well as to TransactTime (60) in the BSE ETI Trade Notification.

[15] This field maps to TrdRegTSTimeIn (21002) in BSE ETI.

BSE - PUBLIC

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 BSE. Reference data messages are sent within different feeds:

- **Snapshot feed of BSE RDI** provides a snapshot of all products and instruments (simple and complex) and is sent out on a regular basis throughout the day. Additions or deletions of complex instruments are incorporated into the next snapshot cycle.

- **Incremental feed of BSE RDI** is event triggered and provides real-time information about complex instruments[16] that are added or deleted intraday. Any change is incorporated within the next snapshot cycle. The incremental feed never contains simple instruments.

- **Market data incremental feed of EMDI** is event triggered and provides real-time information about complex instruments that are added or deleted intraday on the same channel as market data.

- **Market data feed of MDI** is event triggered and provides real time information about complex instruments that are added or deleted intraday.

The following messages are sent via different feeds:

a) Snapshot feed of RDI:

- **Product snapshot** for products available at start of day.

- **Instrument snapshot** for simple and complex instruments available at start of day.

- **Instrument incremental** for complex instruments added or deleted intraday.

- **Market data report** indicates the start of reference data (MDReportEvent=1).

BSE - PUBLIC

- **Market data report** indicates the end of reference data (MDReportEvent=2).

b) **Incremental feed of RDI:**

- Instrument incremental for complex instruments added or deleted intraday.

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

- Complex instrument update[17] for complex instruments added or deleted intraday.

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

- Complex instrument update[17] for complex instruments added or deleted intraday.

[16] No product information is delivered

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

BSE - PUBLIC

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 and complex), repeating for all products and instruments.

A dynamically growing sequence of instrument incremental messages (complex instruments).

Finally **market data report** message (MDReportEvent = 2 = "**EndOfReferenceData**").

Figure 9: Entire snapshot cycle on the RDI snapshot feed

with:

P_n : Product n

I_{nq} : Simple instrument q for product n

C_{nm} : 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 complex instruments are added or deleted. Figure 10 illustrates how more instruments incremental are added over the course of n cycles:

BSE - PUBLIC

Figure 10: Reference data with constant snapshots and extending incremental

> 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, incremental and overall number of messages within the current cycle.

The market data report message, type: "**StartOfReferenceData**", contains the following sequence number fields:

- **MDCount (5488):** 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 products + the number of instruments (simple + complex) at start-of-day. If a failure of BSE RDI occurs the number of messages in the reference data snapshot and herewith MDCount (5488) 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 - MDCount.

- **TotNoMarketSegments (8825):** Contains the number of product messages sent in the snapshot feed. This value remains constant intraday as products are not created or deleted intraday.

BSE - PUBLIC

- **TotNoInstruments (8826):** Contains the number of instrument messages sent in the snapshot feed. This value changes as more complex instruments are created or deleted intraday.

TotNoMarketSegments (8825**)** and TotNoInstruments (8826) can be used as sanity check and to pre-allocate the product and instrument containers.

The market data report message, type: "**EndOfReferenceData**" marks the end of reference data messages and doesn't 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 incremental for complex instruments. [18]

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 and complex 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 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 figure12). The reference data incremental messages for complex instruments C_{11}, C_{41} and C33 are appended to the reference data snapshot messages:

[18] 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.

BSE - PUBLIC

Figure 12: Reference data snapshot after intraday addition of complex instruments C_{11}, C_{41} and C33

LastMsgSeqNumProcessed (369) in the market data report, type: "Start of Reference Data" increases to 103. The no. of incrementals (products + instruments) can be calculated as LastMsgSeqNumProcessed - MDCount = 103 - 100 = 3.

New complex instruments predefined by the exchange are sent in instrument incremental 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 after intraday deletion of a complex instrument

The next example shows an **intraday** deletion of a complex instrument C_{41} * (figure 13). The deleted complex instrument C_{41} * is sent as reference data incremental message and appended again to previously generated complex instruments C_{11} , C_{41} and C_{33} :

Figure 13: Reference data snapshot after intraday deletion of complex instrument C_{41} *

BSE - PUBLIC

Since one complex instrument is deleted, LastMsgSeqNumProcessed (369) increases to 104 snapshot and incremental messages.

9.2.6 Use case 4: 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 incremental 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 14:

Figure 14: Reference data snapshot on the next business

At the beginning of the next business day[19] LastMsgSeqNumProcessed (369) is 102 as this reflects the number of remaining complex instruments C_{11} and C33. C_{41} has already been deleted on the previous business day (see use case 3). At this point in time MDCount (5468) has increased to 102 as well.
The same scenario would be true if the reference data server would have been restarted intraday (due to a technical failure).

9.2.7 Use case 5: Failover or restart of RDI

In the event that RDI fails, another instance takes over. Receiving applications can detect this by a change of the SenderCompID 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 RDI at start of day. The same recovery actions apply in case of a complete restart of RDI.

If RDI fails over the instrument snapshot contains any complex instruments already created and deleted during the day, i.e. the entire history. If RDI needs to be restarted by the exchange a new snapshot cycle is generated. This cycle contains the currently existing complex instruments but not the entire history of creations and deletions.

[19] also in case of a feed restart on the exchange side

BSE - PUBLIC

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

The intraday creation and deletion of complex instruments results in the following sequence of messages:

1. On the **market data incremental feed** of the EMDI and **market data feed** of the MDI: An instrument incremental message is sent to inform the participant as fast as possible. In case a new complex instrument has been created the corresponding message is sent prior to the publication of any order book data for the new complex instrument.

2. On the **reference data incremental feed** of the 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 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, even if complex instruments are added and deleted intraday.

9.3 General order book rules and mechanics

The BSE Market Data Interfaces, 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 EMDI.
- Each BSE EMDI packet relates only to a single product. In other words, although each BSE EMDI packet may contain multiple messages, those messages will always relate to the same product. This does not apply to BSE MDI where a single packet may relate to multiple products.

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

BSE - PUBLIC

- 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”, “Pre-Trading”, “Post-Trading” and “End-Of-Day” or when no price levels exist, an empty book (MDEntryType=J) is disseminated for the depth snapshot message (not for incremental). In addition to an empty book, statistical information is sent in “Pre-Trading”, “Post-Trading” and in “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, the sums of the quantities are reported for the particular price. An example is provided in section 14.2.4, table 49.

- 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 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 a 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).

BSE - PUBLIC

The change from a crossed to an uncrossed book situation and vice versa is implicitly identified by sending ToB information instead of a auction clearing price and vice versa.
The previous ToB information or auction clearing price is only explicitly removed (MDUpdateAction 2=Delete) whenever an uncrossed or crossed book changes to an empty book during an auction.
A instrument state change message identifying an instrument leaving an auction also implicitly deletes the Auction Clearing Price. 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 and the auction starts with an uncrossed order book.

Figure 15 illustrates a particular scenario for an opening auction. The three situations order book "Crossed", "uncrossed" and "empty" can appear in any sequence within the auction

Figure 15: Example for an Opening Auction with implicit and explicit "Deletes"

Note: In general, BSE sends explicit deletes during an auction. However, during the transitions described below, receiving applications need to perform the following actions:
- From ToB to Auction Clearing Price: Receiving applications need to delete ToB.
- From Auction Clearing Price to ToB: Receiving applications need to delete the Auction Clearing Price.
- Whenever the auction ends and another instrument state starts: Receiving applications need to delete the Auction Clearing Price.

BSE - PUBLIC

- 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 MDI carry a common and contiguous sequence number per product. The incremental message of EMDI contains a contiguous sequence number per product across all messages, while the snapshot message provides the last sequence number (LastMsgSeqNumProcessed) sent in the incremental message.

- 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.

BSE - PUBLIC

Figure 16 illustrates a typical order book and terminology used in the following chapters.

Figure 16: 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

The new trading platform 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.

The BSE system 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 16) 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

BSE - PUBLIC

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

9.3.2 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 on this level
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_0	official time of book entry

Table 16: 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).

BSE - PUBLIC

9.3.3 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	1068	
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	10	Quantity
346	> NumberOfOrders	1	Number of order on this level
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_1	official time of book entry

Table 17: MDUpdateAction "Change"

9.3.4 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. The overlay function is normally used when there is just one price level disseminated.

BSE - PUBLIC

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	89	Product
268	NoMDEntries	1	
279	> MDUpdateAction	5	
269	> MDEntryType	0	Bid
48	> SecurityID	8852	Instrument
22	> SecurityIDSource	M	Marketplace-assigned identifier
270	> MDEntryPx	**2.48**	Price
271	> MDEntrySize	N/A	Quantity remain same in this example
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_5	official time of book entry

Table 18: MDUpdateAction "Overlay"

9.3.5 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

BSE - PUBLIC

270	> MDEntryPx	58.22	Price
271	> MDEntrySize	N/A	Quantity not populated for "Delete"
1023	> MDPriceLevel	1	Book level
273	> MDEntryTime	t_1	official time of book entry

Table 19: MDUpdateAction "Delete"

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

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

- Delete all price levels ≥ specified price level.

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

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	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 20: MDUpdateAction "Delete From"

9.3.7 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.

BSE - PUBLIC

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	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 on this level
1023	> MDPriceLevel	3	Book level
273	> MDEntryTime	t_4	official time of book entry

Table 21: MDUpdateAction “Delete Thru”

BSE - PUBLIC

9.4 Trade Volume Reporting (EMDI)

All on-exchange trades executed on the new BSE trading platform 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 EMDI only disseminates information about on-exchange trades. OTC trade information is not disseminated via the market data incremental and market data snapshot feed of the BSE EMDI.

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

The field MDEntryID (278) is a unique id on product level for each business day. A synthetic match can result in more than one trade volume record with the same MDEntryID (278) as shown in use case 3, section 9.5.3 and use case 4, section 9.5.4.

Every match step occurring in the exchange has an identifier in BSE 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 BSE ETI with the market data incremental feed of the BSE EMDI.

The following 4 use cases illustrate the MDEntryID (278) and how Trade Volume Reporting works:

9.4.1 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

BSE - PUBLIC

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	Total
FESX Sep	1	NEW	50@2884	U,R,AX,AY	BUY	1	1	+50
FESX Sep	2	NEW	100@2885	U,AX	BUY	1	1	+100

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

9.4.2 Use case 2: Direct match of complex instruments

An incoming complex order[23] involving 2 legs are 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
	100@8 Sep/Dec

Trade Volume Reporting: Only the entire strategy trade is reported and no trade information is published for the individual legs.

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell	Total
Sep/Dec	3	NEW	50@7	U,R,AX,AY	BUY	1	1	+50
Sep/Dec	4	NEW	100@7	U,AX	BUY	1	1	+100

[23] In general, a complex order is any combination of simple instruments. This example is based on a Future Time Spread. However,

BSE - PUBLIC

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 (28821) and NumberOfSellOrders (28822) 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	Total
Sep/Dec	5	NEW	150@8	U	BUY	1		+150
Sep	5	NEW	150@2886	U,AX			3	+150
Dec	5	New	150@2878	U,R,AX,AY		2		+150

In case the incoming buy spread order is matching against sell spread orders in the same instrument (direct matching), only one trade volume reporting record is published. As a general rule, only one trade

BSE - PUBLIC

volume record is published per match step in case of direct matching (as an example, see section 9.5.2) while there are more than one trade volume reporting record per match step in case of synthetic matching as described in the example of this section.

9.4.3 Use case 4: Complex versus simple/complex match

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

Existing Order book:

Bid	Ask
120@2878 Dec	150@2886 Sep
30@2878 Dec	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 (28821), NumberOfSellOrders (28822). The synthetic match can be identified by the missing entry for NumberOfSellOrders (28822).

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

9.4.4 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 Trade- Condition (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 **25@24.39 Sep** **20@24.39 Sep** **55@24.39 Sep** **5@24.39 Sep**	60@24.39 Sep 57@24.39 Sep 18@24.39 Sep

BSE - PUBLIC

Trade Volume Reporting: All orders are matching on the same price level. Therefore they are reported only once but with different NumberOfBuyOrders (28821) /NumberOfSellOrders (28822) . The AggressorSide (28731) is left empty because during an auction, orders are not considered to be aggressive.

The following depth incremental message is sent:

Instr.	MDEntryID	MDUpdateAction	size@prc	TradeCond.	AggrSide	#Buy	#Sell	Total
Sep	1	New	135@24.39	U,R,AX,AY,AW	OPENING	5	3	+135

The following depth snapshots belong to the depth incremental above:

Instr.	MDUpdateAction	size@prc	TradeCond.	TrdType	Total
Sep	New	135@24.39	U,R,AX,AY,AW		+135
Sep	New	135@24.39	AW	OPENING	N/A

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.

The TradeVolume (1020) is only set if TradeCondition (277) is “U”.

9.5 Trade Volume Reporting (MDI)

The BSE 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 (28821) , NumberOfSellOrders (28822) are not provided.

MDI reports the total volume traded through the respective order book only. This means, for example, that a direct match of two futures spread orders does not increase the total traded volume of the involved single leg contracts.

BSE - PUBLIC

9.6 Failure of the market data feed/ matching engine

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

9.6.1 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 EMDI.
- The MsgSeqNum's in the data message are contiguous per product on the market data feed of the MDI[25].

Figure 17 provides an example for constant SenderCompID's and increasing sequence numbers:

[24] the content is the same.

[25] because the BSE MDI delivers incremental and snapshots on the same channel.

BSE - PUBLIC

Figure 17: Normal processing of sequence numbers for the BSE EMDI

9.6.2 Market data feed fail-over (EMDI)

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

Incremental:

- 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.

BSE - PUBLIC

Figure 18 illustrates the different behavior for incremental and snapshot messages:

Figure 18: Data feed fail-over for EMDI

9.6.3 Market data feed fail-over (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 all products to 1.

The snapshots are sent for all instruments before the incremental are generated.

BSE - PUBLIC

Figure 19 illustrates the different behavior for incremental and snapshot messages:

Figure 19: Data feed fail-over for MDI

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

9.6.4 Market data feed restart (EMDI)

A new SenderCompID, available in the packet header and in each data message for incremental 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.

BSE - PUBLIC

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 EMDI.

9.6.5 Market data feed restart (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 incremental 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 MDI.

9.6.6 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).

BSE - PUBLIC

9.7 Trading states for a sample business day

Section 4.2, Trading states introduced the trading state information. This section describes a typical trading day with the new BSE Exchange Trading Architecture. The examples refer to an equity product A10O5A and a derivatives product FDAX future on a typical expiry day. The times for each trading phase are valid for FDAX and A10O5A.

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.

9.7.1 Start-Of-Day

The system startup occurs in the morning. Note that with the BSE New Trading Architecture, 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, A10O5A 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 9 = 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 and equity securities (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.

BSE - PUBLIC

9.7.2 Pre-Trading

At 7:50 IST, the FDAX, A10O5A product goes into the product state Pre-Trading while all its instruments change their instrument state to Restricted. Traders have no access to the order book.

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 201 = Restricted, which says that all instruments are in the state Restricted. This message is sent once for simple instruments and once for futures spreads.

9.7.3 Trading

At 9:00 IST, the A10O5A product goes into the product state Trading. At the same time, all the simple instruments belonging to product A10O5A change their instrument state to Opening Auction. 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. Traders can do full order maintenance.

At 9:15 IST, the FDAX product goes into the product state Trading. At the same time, all the simple instruments belonging to product A10O5A change their instrument state to Continuous. Traders can do full order maintenance.

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.

For A10O5A product 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.

For FDAX product the system furthermore sends one mass instrument state change message with the field SecurityMassTradingStatus (1679) containing 203 = Continuous, which says that all instruments are in the state opening auction. This message is sent only for simple instruments.

9.7.4 Continuous Trading

BSE - PUBLIC

At 9:15 IST, the opening auction period of the A10O5A 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. Incoming orders 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.

9.7.5 Intraday Expiry

At 12:30 IST, 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 spread) 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 Security-Status (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.

9.7.6 Closing

At 15:30 ISTT, the A10O5A, FDAX product is set into the product state Closing. At the same time, all its simple instruments (futures contracts) and complex instruments (futures spreads) change their instrument state to Restricted. Traders can do only order deletion.

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 and complex instruments, the system sends one mass instrument state change message with the field Se- curityMassTradingStatus (1679) containing 201 = Restricted. The message carries an

BSE - PUBLIC

exception list which contains the expired instrument as the only list item. For this instrument, the list item field SecurityTradingStatus (326) contains 201 = Restricted.

9.7.7 Post Closing

At 15:40 IST, the product A10O5A goes into the product state Post Closing. The simple instruments that have been in the instrument state restricted now change to the state continuous.

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 5 =Evening and the field TradingSessionSubID (625) set to 6 = Post Closing. This indicates the product state: Post Closing.

For simple instruments, the system sends one mass instrument state change message with the field SecurityMassTradingStatus (1679) containing 203 = Continuous. 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.

9.7.8 End-Of-Day

After 16:00 IST, 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 9 = 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.

BSE - PUBLIC

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 datagram 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 affects the ideal buffer size is the ratio of joined multicast streams to available band- width of a 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 BSE 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.

BSE - PUBLIC

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 client's application is interested in:

- Throw it away.

If the message is already in the cache:

- The client's 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 within 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.

BSE - PUBLIC

10.3.3 Optimal processing of desired products (EMDI)

Receiving applications interested in certain products need to join a multicast address which contains the desired products according to the mapping table provide in the reference data. Packets may arrive from different partitions on the same multicast address as shown in figure 20. The PartitionID in the packet header for the BSE 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 20: Discarding packets with unwanted products based on the PartitionID of the BSE EMDI packet header

The example provided in figure 20 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.

BSE - PUBLIC

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.

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

11.1.2 Packet header (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 datagram 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

BSE - PUBLIC

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

them to determine that a packet has already been processed without incurring any decoding overhead at all. 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 EMDI only and is not provided for the MDI.

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 21: Structure of the packet header for BSE 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 is 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 BSE EMDI incremental feed. The PartitionID is available in messages on both incremental and snapshot feed of the EMDI.

BSE - PUBLIC

11.1.3 Packet header (MDI / RDI)

Delivered in: every UDP-datagram

The packet header of BSE MDI and BSE RDI doesn't contain the PerformanceIndicator with length field and the PartitionID. The rest of the packet header is identical to 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 22: Structure of the packet header for MDI and RDI

11.1.4 Functional beacon message

Delivered on: EMDI incremental only

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.

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

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	String	Value: Description 0: Beacon

BSE - PUBLIC

49	SenderCompID	Y	uInt32	Unique id of a sender.
50	SenderSubID	Y	uInt32	Product Identifier, e.g. 89.
369	LastMsgSeqNumProcessed	Y	uInt32	Last sequence number on the incremental feed for this product.

11.1.5 Technical heartbeat message

Delivered on: every channel for EMDI, MDI and RDI

The technical heartbeat 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: BSE 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 (8827). The fields MDCount (34) and LastMsgSeqNumProcessed (369) allow determining 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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Beacon</td></tr></table>
5468	MDCount	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

BSE - PUBLIC

				the start of the cycle.
369	LastMsgSeqNumProcessed	N	uInt32	MDCount 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 - MDCount. Only sent for MDReportEvent = 1, i.e. at the start of the cycle.
28827	MDReportEvent	Y	MDReport-Event (enum)	Defines start/end of reference data. Value: Description 1: StartofRefernce Data 2: EndOfReference Data
60	TransactTime	Y	Timestamp	Creation time of the snapshot.
28825	TotNoMarketSegments	N	uInt32	Contains the number of product messages sent in the snapshot. Only send for MDReportEvent = 1.
28826	TotNoInstruments	N	uInt32	Contains the number of instrument messages sent in the snapshot. Only sent for MDReportEvent = 1.

BSE - PUBLIC

11.2 Reference data messages

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

11.2.1 Product snapshot message

Delivered on: BSE 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 reference data feed (instrument snapshot, instrument incremental)

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	String	Reference data snapshot (one per product). Value: BU Description: Market Definition
34	MsgSeqNum	Y	uInt32	Contiguous across all messages on the reference data snapshot feed regardless of message type. See figure 9 chapters 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".
75	TradeDate	Y	uInt32	Current business date, e.g. "20131128".
7703	MarketSegment	Y	string	Product name, e.g. "FDAX".
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	strin	Standard BSE product types, e.g. "FINX".

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
			g	
15	Currency	Y	string	Currency as published in ISO 4217
28599	MarketSegmentStatus	Y	MDS tatus [27] (enum)	Defines if on-exchange trading on the new trading architecture 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 "BSE 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 "BSE platform for OTC trading with no intention for on-exchange trading. **Value** / **Description** 1 / Active 10 / Published
9543	USFirmFlag	Y	USFir m-Flag (enum)	Flag to identify whether product may be traded from the USA. **Value** / **Description** N / No Y / Yes
5948	PartitionID	Y	uInt32	Partition of the product.
6426	UnderlyingSecurity	N	string	Underlying security symbol, e.g. "DAX".
5336	UnderlyingID	N	string	ISIN code of the underlying, e.g. "DE0008469008".
9939	UnderlyingLastPx	N	decimal	Closing price of the underlying on the previous day.
28700	RefMarketSegmentID	N	uInt32	ID of the product required to refer while creating volatility strategy.
28814	QuoteSideIndicator	Y	QuoteSide	Defines if one sided quotes are allowed. **Value** / **Description**

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
			- Indicator (enum)	0 OneSidedQuoteNotAllowed 1 OneSidedQuoteAllowed
28833	FastMarketPercentage	N	decimal	Percentage by which range resulting from PriceRangeValue (28808) and PriceRangePercentage (28809) 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
<TickRules> sequence starts				
1205	NoTickRules	Y	length	Number of tick rules defining the price step table for the product.
28823	>TickRuleProductComplex	Y	Product-Complex (enum)	Defines the instrument type for the tick rule. Value / Description: 1 SimpleInstrument 2 StandardOptionStrategy 3 NonStandardOptionStrategy 4 VolatilityStrategy 5 FuturesSpread
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).
1208	> TickIncrement	Y	decimal	Tick increment for stated price range. Specifies the valid price increments at which instrument of the product can be quoted and traded.
<TickRules> sequence ends <MatchRules> sequence starts				

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
1235	NoMatchRules	Y	length	Used to convey allocation rules for matching
28824	> MatchRuleProduct- Complex	Y	Product-Complex (enum)	Indicates the instrument type. Value: Description 1: SimpleInstrument 2: StandardOptionStrategy 3: NonStandardOptionStrategy 4: VolatilityStrategy 5: FuturesSpread
1142	> MatchAlgorithm	Y	Match-Algorithm (enum)	Defines the order allocation method. Value: Description PT: PriceTime PR: ProRata TPR: TimeProRata
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: AutoMatch 7: CallAuction
<MatchRules> sequence ends <QuoteSizeRules> sequence starts				
28829	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).
28828	> FastMarketIndicator	N	FastMarket - Indicator (enum)	Indicates if product is in the state Fast Market. Value: Description 0: No

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				1 Yes
<QuoteSizeRules> sequence ends				
<FlexRules> sequence starts				
28830	NoFlexProductEligibilities	N	length	
28831	> FlexProductEligibility- Complex	Y	Product-Complex (enum)	Defines the strategy types available for user defined strategies. Value: Description 1: Simple Instrument 2: StandardOptionStrategy 3: NonStandardOptionStrategy 4: VolatilityStrategy 5: FuturesSpread
1242	> FlexProductEligibilityIndi- cator	Y	Flex-Product-Eligibility-Indicator (enum)	Defines if user defined strategies are allowed. **Value: Description** N: No Y: Yes
<FlexRules> sequence ends <Feeds> sequence starts				
1141	NoMDFeedTypes	Y	length	Number of feeds.
1022	> MDFeedType	Y	MDFeed-Type (enum)	Type of feed. **Value: Description** HI: HighIncremental (unnetted feed) HS: HighSnapshot (unnetted feed) L: Low (netted feed)
1021	> MDBookType	Y	String	**Value: Description** 2: Price Depth

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				3 Order Depth
264	> MarketDepth	N	uInt32	Maximum number of price levels for the product.
28850	>MarketDepthTimeInterval	N	uInt32	Netting interval for low bandwidth feeds (0=no netting).
28851	> MDRecoveryTimeInterval	N	uInt32	Recovery interval (duration of one cycle)
28590	> MDPrimaryFeedLineID	Y	string	IP Address for Service A.
28591	> MDPrimaryFeedLineSubID	Y	uInt32	Port number for IP address Service A.
28592	> MDSecondaryFeedLineID	N	string	IP Address Service B.
28593	> MDSecondaryFeedLine- SubID	N	uInt32	Port number for IP address Service B.
<Feeds> sequence ends				
<PriceRangeRules> sequence starts				
28805	NoPriceRangeRules	Y	length	Price Range Tables.
28806	> PriceRangeRuleID	Y	uInt32	Table identifier.
28810	> PriceRangeProductCom- plex	Y	Product-Complex (enum)	Value / Description 1 SimpleInstrument 2 StandardOptionStrategy 3 NonStandardOptionStrategy 4 VolatilityStrategy 5 FuturesSpread
28812	> StartPriceRange	Y	decimal	Start of price range (inclusive).
28813	> EndPriceRange	Y	decimal	End of price range (non-inclusive).
28808	> PriceRangeValue	N	decimal	Maximum allowable quote spread (absolute value). Conditionally required if PriceRangePercentage is absent.

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
28809	> PriceRangePercentage	N	decimal	Maximum allowable quote spread (percentage value). Condition- ally required if PriceRangeValue is absent.
<PriceRangeRules> sequence ends				

[27] The actual wire format for "Published" is 3 as MDStatus has 4 values defined in the FAST 1.2 XML files.

11.2.2 Instrument snapshot message

Delivered on: BSE RDI snapshot feed

The instrument snapshot message is used to describe both simple and complex instruments. For complex instrument the LegSecurityID (602) is the SecurityID (48) of the instrument which is part of the complex instrument.
A link to reference information for BSE classic 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: d — Description: Securitydefinition
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: M — Description: Market Place Assigned Identifier
<SecurityAlt> (optional) sequence starts				
454	NoSecurityAltID	N	length	
455	> SecurityAltID	N	uInt64	Contract ID from the BSE Legacy System (4 Bytes).
456	>SecurityAltIDSource	N	String	Value — Description

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				M — Market Place Assigned Identifier
<SecurityAlt> (optional) sequence ends				
167	SecurityType	Y	Security-Type (enum)	Type of security. Value: OPT — Option; FUT — Future; MLEG — MultiLegInstrument
762	SecuritySubType	N	uInt32	Standard strategy type for complex instruments, e.g. BER-C.
1227	ProductComplex	Y	Product-Complex (enum)	Type of instrument. Value: 1 — SimpleInstrument; 2 — StandardOptionStrategy; 3 — NonStandardOptionStrategy; 4 — VolatilityStrategy; 5 — FuturesSpread
207	SecurityExchange	N	string	MIC (ISO 10383), used to identify an instrument of a co-operation partner.
<SimpleInstrumentDescriptor> (optional) group starts				
541	MaturityDate	N	uInt32	Actual expiration day of the instrument (YYYYMMDD).
200	MaturityMonthYear	N	uInt32	Expiration month (YYYYMM).
202	StrikePrice	N	decimal	Strike price, e.g. 52.00.
28836	StrikePricePrecision	N	uInt32	Exercise Price Decimals.
28835	PricePrecision	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: 0 — Put

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				1 Call
206	OptAttribute	N	string	Version of an option. Version can change as a result of corporate actions or events.
1194	ExerciseStyle	N	Exercise-Style (enum)	Style family of an option. Value: Description N: No Y: Yes
28832	OrigStrikePrice	N	decimal	Original strike price prior to corporate action, e.g. 5.20.
25100	ContractGenerationNumber	N	string	Contract generation.
28838	LowExercisePriceOptionIndicator	N	LepoFlag (enum)	LEPO Flag Value: Description N: No Y: Yes
28837	BlockTradeEligibilityIndicator	Y	BlkTrdFlag (enum)	Block trade flag Value: Description N: No Y: Yes
1197	ValuationMethod	Y	Valuation-Method (enum)	Traditional or futures margin style Value: Description EQTY: PremiumStyle FUT: FuturesStyleMarkToMarket
1193	SettlMethod	Y	Settl-Method (enum)	Settlement type Value: Description C: Cash P: Physical
28834	SettlSubMethod	N	Settl-SubMetho	Physical settlement type

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
			d (enum)	<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Shares</td></tr><tr><td>2</td><td>Derivatives</td></tr><tr><td>3</td><td>PaymentVsPayment</td></tr><tr><td>4</td><td>Notional</td></tr><tr><td>5</td><td>Cascade</td></tr><tr><td>99</td><td>Other</td></tr></table>
<Events> (optional) group starts				
864	NoEvents	N	length	
865	> EventType	N	uInt32	<table><tr><th>Value</th><th>Description</th></tr><tr><td>7</td><td>Last Eligible Trade Date</td></tr></table>
866	> EventDate	N	uInt32	
<Events> (optional) group ends <SimpleInstrumentDescriptor> (optional) group ends				
<ComplexInstrumentDescriptorGroup> (optional) group starts				
60	TransactTime	N	timestamp	Creation time of complex instruments. This field is empty for deletions of complex instruments.
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
<InstrmtLegGrp> sequence starts				
555	NoLegs	N	length	
602	> LegSecurityID	Y	uInt64	Instrument identifier of the leg security.
603	> LegSecurityIDSource	Y	String	<table><tr><th>Value</th><th>Description</th></tr><tr><td>M</td><td>Market Place assigned identifier</td></tr></table>
624	> LegSide	Y	LegSide	

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
			(enum)	<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>
623	> LegRatioQty	N	uInt32	The ratio of quantity for this individual leg relative to the entire multileg security. The quantity for the leg of an order can be calculated as Leg-RatioQty(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				
969	MinPriceIncrement	Y	decimal	Defines the minimum increment for trade prices in ticks (tick size). This value is identical for all instruments of a product.
1146	MinPriceIncrement Amount	Y	decimal	Defines the minimum price movement in the respective currency (tick value). This value is identical for all instruments of a product.
965	SecurityStatus	Y	MDStatus (enum)[28]	Defines if product is still traded on "BSE classic (=Published) or on NTA (=Active). Instruments may also 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. Active instruments can expire or be suspended which is conveyed with the same field but only within intraday state change messages (EMDI or MDI). <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><tr><td>10</td><td>Published</td></tr></table>
28816	PrevAdjustedOpenI	N	uInt64	Previous day's adjusted open interest.

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
	nterest			
28817	PrevUnadjustedOpenInterest	N	uInt64	Previous day's unadjusted open interest.
28818	PrevSettlPx	N	decimal	Previous day's settlement price.
140	PrevClosePx	N	decimal	Previous day's closing price. The field PrevClosePx is currently not populated.
<MarketSegmentGrp> sequence starts				
1310	NoMarketSegments	Y	length	Always "1".
1300	> MarketSegmentID	Y	uInt32	Product identifier, e.g. 89.
1144	> ImpliedMarketIndicator	N	Implied-Market-Indicator (enum)	Defines the matching method. Value: 0 = NotImplied; 3 = BothImpliedInAndImpliedOut
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. Value: 0 = PredefinedMultilegSecurity; 1 = UserDefinedMultilegSecurity
><PriceRangeRules> (optional) sequence starts				
28805	> NoPriceRangeRules	Y	length	Only one table allowed.
28806	> > PriceRangeRuleID	Y	uInt32	
<MarketSegmentGrp> sequence ends				
><PriceRangeRules> (optional) sequence ends				

[28]The actual wire format for "Published" is 3 as MDStatus has 4 values defined in the FAST 1.2 XML files.

11.2.3 Instrument incremental message

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

The instrument incremental message is used to report additions or deletions of complex instruments to

BSE - PUBLIC

the reference data.

<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>BP</td><td>SecurityDefinitionUpdateReport</td></tr></table></td></tr>
<tr><td>980</td><td>SecurityUpdateAction</td><td>Y</td><td>Security-string</td><td>Intraday creation or deletion of complex instruments.<table><tr><th>Value</th><th>Description</th></tr><tr><td>A</td><td>Add</td></tr></table></td></tr>
</table>

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

On the BSE RDI incremental feed: The *MsgSeqNum* (34) starts with 1 and increments by 1 for each message.

On the BSE 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.

BSE - PUBLIC

11.3 Market data messages

The market data feeds disperse public market data via the BSE EMDI and the BSE MDI. The exchange is able to define the contents of the feeds, the multicast addresses, format of the feed and map exchange offered messages to the feeds.

Public market data for all instruments is 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 addresses and port combinations are different for the BSE EMDI and the BSE 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 BSE EMDI and BSE MDI is the same.

11.3.1 Depth snapshot message

Delivered on: BSE EMDI snapshot feed, BSE 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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>W</td><td>MarketDataSnapshotFullRefresh</td></tr></table>
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

BSE - PUBLIC

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>number sent regardless of message type.</td></tr>
<tr><td>1187</td><td>RefreshIndicator</td><td>N</td><td>Refresh-Indicator (enum)</td><td>Used by netted feed (MDI) only. If set then the depth snapshot information has not been sent with the depth incremental before.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Y</td><td>MandatoryRefresh</td></tr>
<tr><td>N</td><td>OptionalRefresh</td></tr>
</table></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>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>Source Identification.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>M</td><td>Marketplace-assigned Identifier</td></tr>
</table></td></tr>
<tr><td>1227</td><td>ProductComplex</td><td>Y</td><td>uInt32</td><td>The instrument type
<table>
<tr><td>Value</td><td>Description</td></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>
</table></td></tr>
<tr><td>965</td><td>SecurityStatus</td><td>Y</td><td>MDStatus (enum)</td><td>Status of the instrument.
<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>
<tr><td>10</td><td>Published</td></tr>
</table></td></tr>
<tr><td>134</td><td>BidSize</td><td>N</td><td>Int64</td><td>Total bid size of specified instrument.</td></tr>
<tr><td>135</td><td>OfferSize</td><td>N</td><td>Int64</td><td>Total offer size of specified instrument.</td></tr>
<tr><td>779</td><td>LastUpdateTime</td><td>Y</td><td>timestamp</td><td>Time of last change for SecurityID (nanoseconds).</td></tr>
</table>

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				This can be any trade, change of the order book on any price level, or also a product or instrument state change information conveyed in this message.
<MDSshGrp> sequence starts				
268	NoMDEntries	Y	length	
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","Pre-Trading", "Post-Trading" and "End-Of-Day" when no price levels exist. Value / Description: 0 Bid 1 Offer 2 Trade 7 Upper Circuit Limit 8 Lower Circuit Limit J EmptyBook Q AuctionClearingPrice 6 SettlementPrice C OpenInterest W BlockDealReferencePrice 10 Upper Limit Execution Price 11 Lower Limit Execution Price
828	> TrdType	N	TrdType[29] (enum)	Defines when the trade happens. Only present for MDEntryType=2 and TradeCondition=AW. Value / Description: 1 BlockTrade[29] 2 EFP[29] 12 ExchangeForSwap[29] 55 ExchangeBasisFacility [29] 1000 VolaTrade[29] 1001 EFPFinTrade[29]

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				<table><tr><td>1002</td><td>EFPIndexFuturesTrade[30]</td></tr><tr><td>1100</td><td>OpeningAuctionTrade</td></tr><tr><td>1101</td><td>IntradayAuctionTrade</td></tr><tr><td>1102</td><td>VolatilityAuctionTrade</td></tr><tr><td>1103</td><td>ClosingAuctionTrade</td></tr><tr><td></td><td></td></tr></table>
336	> TradingSessionID	N	Trading-SessionID (enum)	Attached to MDEntryType 2=Trade and TradeCondition U=Exchange Last unless there has been no trade so far.In this case it is attached to book information which can simply be Q=Auction Clearing Price or all bids and offers for the currently visible depth (uncrossed book). If there are no trades and no book information then it is attached to J=Empty Book. <table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Day</td></tr><tr><td>3</td><td>Morning</td></tr><tr><td>5</td><td>Evening</td></tr><tr><td>7</td><td>Holiday</td></tr></table>
625	> TradingSessionSubID	N	Trading-Session-SubID (enum)	See description for TradingSessionID. <table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>PreTrading</td></tr><tr><td>3</td><td>Trading</td></tr><tr><td>4</td><td>Closing</td></tr><tr><td>5</td><td>PostTrading</td></tr><tr><td>9</td><td>Quiescent</td></tr></table>
28828	> FastMarketIndicator	N	Fast-Market-Indicator (enum)	See description for TradingSessionID. <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>

BSE - PUBLIC

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td>326</td><td>> SecurityTradingStatus</td><td>N</td><td>Security-Trading-Status (enum)</td><td>See description for TradingSessionID.
<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>OpeningAuction</td></tr>
<tr><td>205</td><td>OpeningAuctionFreeze</td></tr>
<tr><td>206</td><td>IntradayAuction</td></tr>
<tr><td>207</td><td>IntradayAuctionFreeze</td></tr>
<tr><td>208</td><td>CircuitBreakerAuction</td></tr>
<tr><td>209</td><td>CircuitBreakerAuctionFreeze</td></tr>
<tr><td>210</td><td>ClosingAuction</td></tr>
<tr><td>211</td><td>ClosingAuctionFreeze</td></tr>
</table></td></tr>
<tr><td>277</td><td>> TradeCondition</td><td>N</td><td>Trade-Condition (set)</td><td>AW is not be combined with any other value and have 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>ExchangeLast</td></tr>
<tr><td>R</td><td>OpeningPrice</td></tr>
<tr><td>AX</td><td>HighPrice</td></tr>
<tr><td>AY</td><td>LowPrice</td></tr>
<tr><td>AJ</td><td>OfficialClosingPrice</td></tr>
<tr><td>AW</td><td>LastAuctionPrice</td></tr>
<tr><td>k</td><td>Out of sequence</td></tr>
<tr><td>AN</td><td>PreviousClosePrice</td></tr>
</table></td></tr>
<tr><td>270</td><td>> MDEntryPx</td><td>N</td><td>decimal</td><td>Price.</td></tr>
<tr><td>236</td><td>>Yield</td><td>N</td><td>decimal</td><td>Yield to maturity</td></tr>
<tr><td>30630</td><td>>PutYield</td><td>N</td><td>decimal</td><td>Yield to Put</td></tr>
<tr><td>30631</td><td>>CallYield</td><td>N</td><td>decimal</td><td>Yield to Call</td></tr>
<tr><td>271</td><td>> MDEntrySize</td><td>N</td><td>uInt32</td><td>Quantity.</td></tr>
<tr><td>346</td><td>> NumberOfOrders</td><td>N</td><td>uInt32</td><td></td></tr>
</table>

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
1023	> MDPriceLevel	N	uInt32	Book level. Absent for implied bid/offer prices.
273	> MDEntryTime	N	timestamp	Time of entry (nanoseconds) for last trade entry only (Trade-Condition="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.
1020	> TradeVolume	N	uInt64	Cumulative volume of units traded in the day. Only sent for MDEntryType 2=Trade.
381	> GrossTradeAmt	N	decimal	The total Traded value for the security Trade Value = Quantity * Rate * lot
897	> NoTrades	N	uInt32	Total number of trades.
<MDSshGrp> sequence end				

[29] The values 1, 2, 12, 55, 1000, 1001, 1002 are OTC related and are therefore never sent in BSE EMDI/MDI.

11.3.2 Depth incremental message

Delivered on: BSE EMDI incremental feed, BSE 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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>X</td><td>MarketDataIncremental Refresh</td></tr></table>
34	MsgSeqNum	N	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".

BSE - PUBLIC

<table>
<tr><th>Tag</th><th>Field Name</th><th>Req'd</th><th>Data Type</th><th>Description</th></tr>
<tr><td colspan="5"><MDSshGrp> sequence starts</td></tr>
<tr><td>268</td><td>NoMDEntries</td><td>Y</td><td>length</td><td></td></tr>
<tr><td>279</td><td>> MDUpdateAction</td><td>Y</td><td>MDUpdate-Action (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>New</td></tr><tr><td>1</td><td>Change</td></tr><tr><td>2</td><td>Delete</td></tr><tr><td>3</td><td>DeleteThru</td></tr><tr><td>4</td><td>DeleteFrom</td></tr><tr><td>5</td><td>Overlay</td></tr></table></td></tr>
<tr><td>269</td><td>> MDEntryType</td><td>Y</td><td>MDEntry-Type (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Bid</td></tr><tr><td>1</td><td>Offer</td></tr><tr><td>2</td><td>Trade</td></tr><tr><td>7</td><td>Upper Circuit Limit</td></tr><tr><td>8</td><td>Lower Circuit Limit</td></tr><tr><td>J</td><td>EmptyBook</td></tr><tr><td>Q</td><td>AuctionClearingPrice</td></tr><tr><td>6</td><td>SettlementPrice</td></tr><tr><td>C</td><td>OpenInterest</td></tr><tr><td>W</td><td>BlockDealReferencePrice</td></tr><tr><td>10</td><td>Upper Limit Execution Price</td></tr><tr><td>11</td><td>Lower Limit Execution Price</td></tr></table></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>Source Identification<table><tr><th>Value</th><th>Description</th></tr><tr><td>M</td><td>Marketplace-assigned Identifier</td></tr></table></td></tr>
<tr><td>270</td><td>> MDEntryPx</td><td>N</td><td>decimal</td><td>Price of market data (trade or order).</td></tr>
<tr><td>236</td><td>>Yield</td><td>N</td><td>decimal</td><td>Yield to maturity</td></tr>
<tr><td>30630</td><td>>PutYield</td><td>N</td><td>decimal</td><td>Yield to Put</td></tr>
<tr><td>30631</td><td>>CallYield</td><td>N</td><td>decimal</td><td>Yield to Call</td></tr>
<tr><td>271</td><td>> MDEntrySize</td><td>N</td><td>uInt32</td><td>N Quantity of market data (trade or order).</td></tr>
</table>

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
346	> NumberOfOrders	N	uInt32	
1023	> MDPriceLevel	N	uInt32	Book level. Absent for implied bid/offer prices.
134	> BidSize	N	Int64	Total bid size of specified instrument.
135	> OfferSize	N	Int64	Total offer size of specified instrument.
273	> MDEntryTime	N	timestamp	For bids and offers the official time of book entry, for trades official time of execution (all in nanoseconds).
<TradeEntryGrp> (optional) group starts				
828	> TrdType	N	TrdType[30] (enum)	Defines when the trade happens. Only presents for MDEntryType=2 and TradeCondition=AW. For trades outside the auctions, this field is not set. Value / Description 1 / BlockTrade[30] 2 / EFP[30] 12 / ExchangeForSwap[30] 55 / ExchangeBasisFacility [30] 1000 / VolaTrade[30] 1001 / EFPFinTrade[30] 1002 / EFPIndexFuturesTrade[30] 1100 / OpeningAuctionTrade 1101 / IntradayAuctionTrade 1102 / VolatilityAuctionTrade 1103 / ClosingAuctionTrade
1020	> TradeVolume	N	uInt64	
277	> TradeCondition	N	Trade-Condition (set)	Defines the type of price for MDEntryPx. Only present for MDEntryType=2. Value / Description U / ExchangeLast R / OpeningPrice AX / HighPrice AY / LowPrice AJ / OfficialClosingPrice AW / LastAuctionPrice k / Out of sequence

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				AN: PreviousClosePrice
28819	> MDGapIndicator	N	uInt32	Reserved for future use.
28820	> AggressorTimestamp	N	timestamp	Entry time of the incoming order that triggered the trade. Only present for MDEntryType=2.
28731	> AggressorSide	N	Aggressor-Side (enum)	Side of the incoming order, which created the trade. Only present for MDEntryType=2. Value: 1 = Buy; 2 = Sell
28821	> NumberOfBuyOrders	N	uInt32	Number of buy orders involved in this trade. Only present for MDEntryType=2.
28822	> NumberOfSellOrders	N	uInt32	Number of sell orders involved in this trade. Only present for MDEntryType=2.
278	> MDEntryID	N	uInt32	Represents the match step ID. This field is unique together with MarketSegmentID. Only present for MDEntryType = 2.
<TradeEntryGrp> (optional) group ends				
<MDIncGrp> sequence ends				

[30] The values 1, 2, 12, 55, 1000, 1001,1002 are OTC related and part of the Extended Reference & Market Data Service.

11.3.3 Product state change message

Delivered on: BSE EMDI incremental feed, BSE 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

BSE - PUBLIC

<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>h</td><td>TradingSessionStatus</td></tr></table></td></tr>
<tr><td>34</td><td>MsgSeqNum</td><td>N</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>336</td><td>TradingSessionID</td><td>Y</td><td>Trading-SessionID (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Day</td></tr><tr><td>3</td><td>Morning</td></tr><tr><td>5</td><td>Evening</td></tr><tr><td>7</td><td>Holiday</td></tr></table></td></tr>
<tr><td>625</td><td>TradingSessionSubID</td><td>Y</td><td>Trading-Session-SubID (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>PreTrading</td></tr><tr><td>3</td><td>Trading</td></tr><tr><td>4</td><td>Closing</td></tr><tr><td>5</td><td>PostTrading</td></tr><tr><td>6</td><td>PostClosing</td></tr><tr><td>9</td><td>Quiescent</td></tr></table></td></tr>
<tr><td>340</td><td>TradSesStatus</td><td>Y</td><td>TradSes-Status (enum)</td><td><table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Halted</td></tr><tr><td>2</td><td>Open</td></tr><tr><td>3</td><td>Closed</td></tr></table></td></tr>
<tr><td>28828</td><td>FastMarketIndicator</td><td>Y</td><td>Fast-Market-Indicator (enum)</td><td>Indicates if product is in state "Fast Market".<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></td></tr>
</table>

BSE - PUBLIC

11.3.4 Mass instrument state change message

Delivered on: BSE EMDI incremental feed, BSE 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* (28828) 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.*

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	string	<table><tr><th>Value</th><th>Description</th></tr><tr><td>CO</td><td>SecurityMassStatus</td></tr></table>
34	MsgSeqNum	N	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".
1544	InstrumentScopeProduct-Complex	Y	Instrument-Scope-Product-Complex (enum)	Instrument type of affected instruments.<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>SimpleInstrument</td></tr><tr><td>2</td><td>StandardOptionStrategy</td></tr><tr><td>3</td><td>NonStandardOptionStrategy</td></tr><tr><td>4</td><td>VolatilityStrategy</td></tr><tr><td>5</td><td>FuturesSpread</td></tr></table>
1679	SecurityMassTradingStatus	Y	Security-Mass-Trading-Status (enum)	<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></table>

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				<table><tr><td>203</td><td>Continuous</td></tr><tr><td>204</td><td>OpeningAuction</td></tr><tr><td>205</td><td>OpeningAuctionFreeze</td></tr><tr><td>206</td><td>IntradayAuction</td></tr><tr><td>207</td><td>IntradayAuctionFreeze</td></tr><tr><td>208</td><td>CircuitBreakerAuction</td></tr><tr><td>209</td><td>CircuitBreakerAuctionFreeze</td></tr><tr><td>210</td><td>ClosingAuction</td></tr><tr><td>211</td><td>ClosingAuctionFreeze</td></tr></table>
28828	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. <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>
60	TransactTime	Y	timestamp	Time when request was processed by the matcher (nanoseconds).
<SecMassStatGrp> sequence starts				
146	NoRelatedSym	N	length	
48	> SecurityID	Y	Int64	Instrument identifier, e.g. "8852".
22	> SecurityIDSource	Y	string	<table><tr><th>Value</th><th>Description</th></tr><tr><td>M</td><td>Marketplace-assigned Identifier</td></tr></table>
965	> SecurityStatus	Y	Security-Status (enum)	<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><tr><td>10</td><td>Published</td></tr></table>
326	> SecurityTradingStatus	Y	Security-	

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
			Trading-Status (enum)	<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>OpeningAuction</td></tr><tr><td>205</td><td>OpeningAuctionFreeze</td></tr><tr><td>206</td><td>IntradayAuction</td></tr><tr><td>207</td><td>IntradayAuctionFreeze</td></tr><tr><td>208</td><td>CircuitBreakerAuction</td></tr><tr><td>209</td><td>CircuitBreakerAuctionFreeze</td></tr><tr><td>210</td><td>ClosingAuction</td></tr><tr><td>211</td><td>ClosingAuctionFreeze</td></tr></table>
<SecMassStatGrp> sequence ends				

11.3.5 Instrument state change message

Delivered on: BSE EMDI incremental feed, BSE 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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>f</td><td>SecurityStatus</td></tr></table>
34	MsgSeqNum	N	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	<table><tr><th>Value</th><th>Description</th></tr></table>

BSE - PUBLIC

Tag	Field Name	Req'd	Data Type	Description
				M: Marketplace assigned Identifier
965	> SecurityStatus	Y	Security-Status (enum)	Value / Description: 1 Active; 2 Inactive; 4 Expired; 9 Suspended; 10 Published
326	> SecurityTradingStatus	Y	Security-Trading-Status (enum)	Value / Description: 200 Closed; 201 Restricted; 202 Book; 203 Continuous; 204 OpeningAuction; 205 OpeningAuctionFreeze; 206 IntradayAuction; 207 IntradayAuctionFreeze; 208 CircuitBreakerAuction; 209 CircuitBreakerAuctionFreeze; 210 ClosingAuction; 211 ClosingAuctionFreeze
28828	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
60	TransactTime	Y	timestamp	Time when request was processed by the matcher (nanoseconds).

11.3.6 Complex instrument update message

BSE - PUBLIC

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

This message provides information for new or deleted 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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>BP</td><td>SecurityDefinitionUpdateReport</td></tr></table>
34	MsgSeqNum	N	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	<table><tr><th>Value</th><th>Description</th></tr><tr><td>A</td><td>Add</td></tr></table>
48	SecurityID	Y	Int64	The instrument identifier uniquely identifies an instrument, e.g.72057800196358145.
22	SecurityIDSource	Y	String	<table><tr><th>Value</th><th>Description</th></tr><tr><td>M</td><td>Marketplace-assigned identifier</td></tr></table>
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 <table><tr><th>Value</th><th>Description</th></tr><tr><td>MLEG</td><td>Complex Instrument</td></tr></table>

BSE - PUBLIC

762	SecuritySubType	N	uInt32	Standard strategy type for complex instruments, e.g. BER-C. The mapping of strategy types with the corresponding integer value is published in the file "Instrument Subtype Tables" on the BSE website
1227	ProductComplex	Y	Product- Complex (enum)	Type of instrument **Value** / **Description** 1 SimpleInstrument 2 StandardOptionStrategy 3 NonStandardOptionStrategy 4 VolatilityStrategy 5 FuturesSpread
<InstrmtLegGrp> sequence starts				
555	NoLegs	N	length	
602	> LegSecurityID	Y	uInt64	Instrument identifier of the leg security.
603	> LegSecurityIDSource	Y	String	**Value** / **Description** M Marketplace assigned identifier
623	> LegRatioQty	N	uInt32	The ratio of quantity for this individual leg relative to the entire multileg security.
624	> LegSide	Y	LegSide (enum)	**Value** / **Description** 1 Buy 2 Sell
566	> LegPrice	N	decimal	Price for the leg instrument. Used for the underlying leg of a Volatility Strategy.
<InstrmtLegGrp> sequence ends				
<MarketSegmentGrp> sequence starts				
1310	> NoMarketSegments	N	length	Always "1".
1300	> MarketSegmentID	Y	uInt32	Product identifier, e.g. 89.
1144	> ImpliedMarketIndica	N	Implied- Market-Indicator (enum)	Matching method **Value** / **Description**

BSE - PUBLIC

	tor			0 NotImplied 3 BothImpliedInAndImpliedOut
<MarketSegmentGrp> sequence ends				
60	TransactTime	N	timestamp	Creation time of complex instruments. This field is empty for dele- tions of complex instruments.

11.3.7 Index change message

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

This message provides information for the change in the index value for the indices. The message is delivered in the equity multicast stream only. The message is complete each time it is delivered and thus does not require recovery.

Tag	Field Name	Req'd	Data Type	Description
35	MsgType	Y	String	Type of message **Value** / **Description** I / Index broadcast
<IDXGrp> Sequence Starts				
30621	NoOfEntries	Y	length	No. of entries of indices
30622	> Divisor	N	uInt32	Divisor for received index related values (default value = 100)
30623	> Indexcode	Y	uInt32	Index code. The Index code and the corresponding index name are published in the section 14.4
30624	> IndexHigh	Y	uInt32	Index high value for the day.
30625	> IndexLow	Y	uInt32	Index low value for the day.
30626	> IndexOpen	Y	uInt32	Index open value for the day.
30627	> PrevIndexClose	Y	uInt32	Index previous close value.
30628	> IndexClose	Y	uInt32	Index close value for the day.
30629	> IndexValue	Y	uInt32	Current Index value.
<IDXGrp> Sequence Ends				

BSE - PUBLIC

Data files

11.3.8 Reference data from file (BSE 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.

As per the FIXML specification, BSE RDF field names are different from the BSE 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 BSE RDI, the BSE RDF mechanism operates on a snapshot plus incremental basis.

The initial file (“snapshot”) is created around 6:00 IST 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 and deletion events for complex instruments 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 intraday created complex instruments are described in section 9.2.8, Use case 6: Chronological order of messages for complex instrument creation.

11.3.9 File name format of the reference data files

BSE - PUBLIC

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 look as follows:

Content	File Name
Start-Of-Day file	[@@][report name][member][businessdate][mic_1][mic_2][fileset identifier][nnn].XML
Intraday updates	[@@][report name][member][business date][mic_1][mic_2][file set identifier][nnn].XML

Table 43: 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_1 (length 4): mic code, always XEUR for BSE

mic_2 (length 4): mic code, always 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 BSE: 95FILRDF01PUBLI20120815XEURXEEECZA80000.XML

First intraday updated file for BSE: 95FILRDF01PUBLI20120815XEURXEEECZA80001.XML second intraday updated file for BSE: 95FILRDF01PUBLI20120815XEURXEEECZA80002.XML

...

Note: The file name always contains 2 MIC codes because it holds products from two markets.

11.3.10 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.

11.3.11 Reference data file after a failover or restart of BSE RDI

The file set identifier changes after a restart or failover of the BSE 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.

BSE - PUBLIC

Example:
Start-Of-Day file **after** the failover: 95FILRDF01PUBLI20120815XEURXEEE**CZL**C0000.XML
First intraday updated file **after** the failover: 95FILRDF01PUBLI20120815XEURXEEE**CZL**C0001.XML
Second intraday updated file **after** the failover: 95FILRDF01PUBLI20120815XEURXEEE**CZL**C0002.XML

If BSE RDI fails over a new initial reference file is generated with a new file set identifier. This file contains any complex instruments, already created and deleted during the day, i.e. the entire history.

If BSE 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 creations and deletions.

11.3.12 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.

BSE - PUBLIC

12 Multicast addresses

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

12.1 Reference data snapshot feed

Simulation	Service A	Service B
Currency Derivatives	226.1.3.7/10407	226.1.3.8/10408
Equity Derivatives	226.1.1.7/11407	226.1.1.7/11408
Equity	226.1.0.7/12407	226.1.0.8/12408

Production	Service A	Service B
Currency Derivatives	229.0.0.13/10986	229.0.0.13/10987
Equity Derivatives	228.0.0.13/11986	228.0.0.13/11987
Equity	227.0.0.13/12986	227.0.0.13/12987

Table 44: Multicast addresses and ports for the reference data snapshot feeds

12.2 Reference data incremental feed

Simulation	Service A	Service B
Currency Derivatives	226.1.3.5/10405	226.1.3.6/10406
Equity Derivatives	226.1.1.5/11405	226.1.1.6/11406
Equity	226.1.0.5/12405	226.1.0.6/12406

Production	Service A	Service B
Currency Derivatives	229.0.0.11/10984	229.0.0.12/ 10985
Equity Derivatives	228.0.0.11/11984	228.0.0.12/ 11985
Equity	227.0.0.11/12984	227.0.0.12/ 12985

Table 45: Multicast addresses and ports for the reference data incremental feed

BSE - PUBLIC

BSE - PUBLIC

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 BSE in separate files:

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

The FAST templates can be downloaded from BSE website at: www.bseindia.com/nta.aspx

BSE - PUBLIC

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 23: Example for a FAST template with repeating group

BSE - PUBLIC

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 46: 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.

BSE - PUBLIC

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	empty indicates implied price
273	> MDEntryTime	t_1	official time of book entry

Table 47: 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

BSE - PUBLIC

Tag number	Tag name	Value	Description
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	empty indicates implied price
273	> MDEntryTime	t_2	official time of book entry

Table 48: Partially Implied

14.2.4 Several fully implied orders at Best Market (example for 9.3, General order book rules and mechanics)

Precondition: Fully implied Buy FDAX Sep 100@117

Action: Buy FDAX Jun 100@118, Sell FDAX Jun/Sep 100@1.
This creates an implied for Buy FDAX Sep at 100@ (118-1) = 100@117.

Result: The reported quantity is the sum of the quantities for the two implied orders (100@117) 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

BSE - PUBLIC

Tag number	Tag name	Value	Description
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 49: Quantities are added for two or more fully implied orders at the same price.

BSE - PUBLIC

14.3 FIXML mapping table

The following table provides a mapping between FIXML and FIX fields. The mapping table belongs to chapter 11.4.1, Reference data from file (BSE RDF). SecDef refers to the *instrument snapshot* message, SecDefUpd refers to *instrument incremental* message and *MktDef* refers to the *product snapshot* message.

The initial file ("snapshot") includes all three message types while the incremental file contains only the *SecDefUpd* message.

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

The *market data report* message is not used in the BSE 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 Field Name	FIX Field Name	FIX Tag/Group	Repeating	FIX Messages
AID	SecurityAlt	Group	Y	SecDef, SecDefUpd
AltID	SecurityAltID	455		SecDef, SecDefUpd
AltIDSrc	SecurityAltIDSource	456		SecDef, SecDefUpd
BaseTrdgRules	BaseTradingRules	Group	N	MktDef, SecDef, SecDefUpd
BlckTrdElig	BlockTradeEligibilityIndicator	28837		SecDef, SecDefUpd
Ccy	Currency	15		MktDef
CntrctGenNum	ContractGenerationNumber	25100		SecDef, SecDefUpd
Desc	SecurityDesc	107		SecDef, SecDefUpd
Dt	EventDate	866		SecDef, SecDefUpd
EndPxRng	EndPriceRange	28813		MktDef
EndTickPxRng	EndTickPriceRange	1207		MktDef
EventTyp	EventType	865		SecDef, SecDefUpd
Evnt	Events	Group	Y	SecDef, SecDefUpd
Exch	SecurityExchange	207		SecDef, SecDefUpd
ExerStyle	ExerciseStyle	1194		SecDef, SecDefUpd
FastMktInd	FastMarketIndicator	28828		MktDef
FastMktPctg	FastMarketPercentage	28833		MktDef
FlexProdElig	FlexProductEligibilityIndicator	1242		MktDef

BSE - PUBLIC

FIXML Field Name	FIX Field Name	FIX Tag/Group	Repeating	FIX Messages
FlexProdEligCmplx	FlexProductEligibilityComplex	28831		MktDef
FlexProdEligs	FlexRules	Group	Y	MktDef
ID	LegSecurityID	602		SecDef, SecDefUpd
ID	SecurityID	48		SecDef, SecDefUpd
ImpldMktInd	ImpliedMarketIndicator	1144		SecDef, SecDefUpd
Instrmt	Instrument	Group	N	SecDef, SecDefUpd
LEPOInd	LowExercisePriceOptionIndicator	28838		SecDef, SecDefUpd
Leg	InstrmtLegGrp	Group	Y	SecDef, SecDefUpd
MDBkTyp	MDBookType	1021		MktDef
MDFeedLineID1	MDPrimaryFeedLineID	28590		MktDef
MDFeedLineID2	MDSecondaryFeedLineID	28592		MktDef
MDFeedLineSubID1	MDPrimaryFeedLineSubID	28591		MktDef
MDFeedLineSubID2	MDSecondaryFeedLineSubID	28593		MktDef
MDFeedTyp	MDFeedType	1022		MktDef
MDFeedTyps	Feeds	Group	Y	MktDef
MMY	MaturityMonthYear	200		SecDef, SecDefUpd
MarketSegmentDesc	MarketSegmentDesc	1396		MktDef
MatDt	MaturityDate	541		SecDef, SecDefUpd
MinBidSz	MinBidSize	647		MktDef
MinOfrSz	MinOfferSize	648		MktDef
MinPxIncr	MinPriceIncrement	969		SecDef, SecDefUpd
MinPxIncrAmt	MinPriceIncrementAmount	1146		SecDef, SecDefUpd
MktDepth	MarketDepth	264		MktDef
MktID	MarketID	1301		MktDef
MktSeg	MarketSegment	7703		MktDef
MktSegGrp	MarketSegmentGrp	Group	Y	SecDef, SecDefUpd
MktSegID	MarketSegmentID	1300		MktDef ,SecDef, SecDefUpd
MktSegStat	MarketSegmentStatus	28599		MktDef
MlegModel	MultilegModel	1377		SecDef, SecDefUpd
MtchAlgo	MatchAlgorithm	1142		MktDef
MtchRuleProdCmplx	MatchRuleProductComplex	28824		MktDef
MtchRules	MatchRules	Group	Y	MktDef
MtchTyp	MatchType	574		MktDef
Mult	ContractMultiplier	231		SecDef, SecDefUpd
OptAt	OptAttribute	206		SecDef, SecDefUpd
OrigStrkPx	OrigStrikePrice	28832		SecDef, SecDefUpd
ParentMktSegmID	ParentMktSegmID 1325	1325		MktDef

BSE - PUBLIC

FIXML Field Name	FIX Field Name	FIX Tag/Group	Repeating	FIX Messages
	MktDef			
PartID	PartitionID	5948		MktDef
PrevAdjOpenInt	PrevAdjustedOpenInterest	28816		SecDef, SecDefUpd
PrevClsPx32	PrevClosePx32	140		SecDef, SecDefUpd
PrevSettlPx	PrevSettlPx	28818		SecDef, SecDefUpd
PrevUnadjOpenInt	PrevUnadjustedOpenInterest	28817		SecDef, SecDefUpd
ProdCmplx	ProductComplex	1227		SecDef, SecDefUpd
PutCall	PutOrCall	201		SecDef, SecDefUpd
Px	LegPrice	566		SecDef, SecDefUpd
PxPrcsn	PricePrecision	28835		SecDef, SecDefUpd
PxRngPctg	PriceRangePercentage	28809		MktDef
PxRngProdCmplx	PriceRangeProductComplex	28810		MktDef
PxRngRuleID	PriceRangeRuleID	28806		MktDef ,SecDef, SecDefUpd
PxRngRules	PriceRangeRules	Group	Y	MktDef ,SecDef, SecDefUpd
PxRngValu	PriceRangeValue	28808		MktDef
QtSideInd	QuoteSideIndicator	28814		MktDef
QuotSizeRules	QuoteSizeRules	Group	Y	MktDef
RatioQty	LegRatioQty	623		SecDef, SecDefUpd
RefMktSegID	RefMarketSegmentID	28700		MktDef
SecTrdgRules	SecurityTradingRules	Group	N	SecDef, SecDefUpd
SecTyp	SecurityType	167		SecDef, SecDefUpd
SettlMeth	SettlMethod	1193		SecDef, SecDefUpd
SettlSubMeth	SettlSubMethod	28834		SecDef, SecDefUpd
Side	LegSide	624		SecDef, SecDefUpd
Src	LegSecurityIDSource	603		SecDef, SecDefUpd
Src	SecurityIDSource	22		SecDef, SecDefUpd
StartPxRng	StartPriceRange	28812		MktDef
StartTickPxRng	StartTickPriceRange	1206		MktDef
Status	SecurityStatus	965		SecDef, SecDefUpd
StrkPx	StrikePrice	202		SecDef, SecDefUpd
StrkPxPrcsn	StrikePricePrecision	28836		SecDef, SecDefUpd
SubTyp	SubTyp SecuritySubType	762		SecDef, SecDefUpd
Sym	MarketSegmentSymbol	7177		MktDef
TickIncr	TickIncrement	1208		MktDef
TickRuleProdCmplx	TickRuleProductComplex	28823		MktDef
TickRules	TickRules	Group	Y	MktDef
TrdDt	TradeDate	75		MktDef
TxnTm	TransactTime	60		SecDef, SecDefUpd

BSE - PUBLIC

FIXML Field Name	FIX Field Name	FIX Tag/Group	Repeating	FIX Messages
USFirmFlag	USFirmFlag	9543		MktDef
UndID	UnderlyingID	5336		MktDef
UndLastPx	UnderlyingLastPx	9939		MktDef
UndSec	UnderlyingSecurity	6426		MktDef
UpdActn	SecurityUpdateAction	980		SecDefUpd
ValMeth	ValuationMethod	1197		SecDef, SecDefUpd

14.4 Index Code and Mapping Table

Index Code	Index Name	Description
1	SENSEX	BSE sensitive index
2	BSE100	BSE 100 scrips index
3	BSE200	BSE 200 scrips index
4	BSE500	BSE 500 scrips index
34	BSE IT	S&P BSE Information Technology
32	BSEFMC	S&P BSE Fast Moving Consumer Goods
7	BSE CG	BSE Capital Goods Index
8	BSE CD	BSE Consumer Durables index
33	BSE HC	S&P BSE Healthcare
10	BSE PSU	BSE Public Sector Unit Index
11	TECK	BSE Teck Index
12	BANKEX	BSE Bank Index
13	AUTO	BSE AUTO index
14	METAL	BSE METAL index
15	OILGAS	BSE OIL&GAS index
30	MIDCAP	S&P BSE MidCap
31	SMLCAP	S&P BSE SmallCap
18	DOL30	Dollex-30
19	DOL100	Dollex-100
20	DOL200	Dollex-200
21	REALTY	BSE Realty Index
22	POWER	BSE Power Index
23	BSEIPO	BSE IPO Index
25	GREENX	BSE GREENEX

BSE - PUBLIC

26	CARBON	BSE CARBON
27	SMEIPO	BSE SME IPO
28	INFRA	S&P BSE India Infrastructure Index
29	CPSE	S&P BSE CPSE
35	MFG	S&P BSE MFG
36	ALLCAP	S&P BSE AllCap
37	BASMTR	S&P BSE Basic Matrl
38	CDGS	S&P BSE Cons Discr
39	ENERGY	S&P BSE Energy
40	FIN	S&P BSE Finance
41	INDSTR	S&P BSE Industrials
42	LRGCAP	S&P BSE LargeCap
43	MIDSEL	S&P BSE MidCap Sel
44	SMLSEL	S&P BSE SmallCapSel
45	TELCOM	S&P BSE Telecom
46	UTILS	S&P BSE Utilities
47	SNSX50	S&P BSE SENSEX 50
48	SNXT50	S&P BSE SENSEX Next 50
49	BHRT22	S&P BSE Bharat 22
50	ESG100	S&P BSE 100 ESG Index

15 Change log

This document has been through the following iterations:

15.1 Changes compared to version 1.3.3

Section	Description
4.2.2 Instrument State Changes	New instrument states added
9.8 Trading States for normal Business day	New Product State Post Closing Added.

15.2 Changes compared to version 1.3.4

Section	Description
11.3.1 Depth Snapshot Message	New Functional Fields added
11.3.2 Depth Incremental Message	New Functional Fields added

BSE - PUBLIC

15.3 Changes compared to version 1.3.5

Section	Description
2.3 Data feeds 2.3.1 Reference Data Interface 3 Difference between the Interfaces 3.4 Choosing between RDI and RDF 4.3.1 RDI 9.1 Reference data messages 11.2.1 Product snapshot message 11.2.2 Instrument snapshot message 11.2.3 Instrument incremental message 11.3.6 Complex instrument update message	New Sections for the reference data feed are added. The Reference data or the master data can be retrieved using multicast mode in addition to the existing file mode. The messages and the mechanisms are published in the document in these new sections. However the availability of these messages on multicast channel is not present currently. It would be made available soon.
4.3.2 EMDI / MDI 11.3.7 Index Change Message	New index change message introduced to disseminate the index values for the equity segment. The message will be sent periodically on EMDI and MDI incremental channel only.
11.3.1 Depth Snapshot Message	The data type for the field TradeValue (381) is changed from uInt64 to decimal

15.4 Changes compared to version 1.3.6

Section	Description
11.3.1 Depth Snapshot Message	The position of the fields NoOfTrades(897) and TradeValue(381) are moved within MDSshGrp. The field name are changed as follows **Old Name** / **New Name** NoOfTrades(897) / NoTrades(897) TradeValue(381) / GrossTradeAmt(381)

BSE - PUBLIC

	New Enumeration *PreviousClosePrice* added for the field TradeCondition(277)
11.3.2 Depth Incremental Message	New Enumeration *PreviousClosePrice* added for the field TradeCondition(277)

15.5 Changes compared to version 1.3.7

Section	**Description**
11.3.1 Depth snapshot message	Missing Enumeration *out of sequence* added for the field TradeCondition(277) New Field *ProductComplex(1227)* added.
11.3.5 Instrument state change message	*MsgType(35*) value corrected from CO to f
11.3.3 Product state change message	Enumeration Postclosing = 6 added for the field *TradingSessionSubID(625)* Enumeration value for Quiescent changed from 7 to 9
14.4 Index code mapping table	New Index S&P BSE India Infrastructure Index (28) added.
15.5.1 Instrument snapshot message	New Field *LegSide* added under <InstrmtLegGrp> sequence
15.5.2 Product snapshot message	Enumeration added for the field *FlexProductEligibilityComplex(28831)*
5.2.4 Template IDs	New Template IDs defined for new messages Template ID for Index broadcast in MDI stream changed from 50 to 51.
Data Type Changes	SecurityID(48) changed from Uint64 to Int64. SecurityUpdateAction(980) changed from enum to string

BSE - PUBLIC

15.6 Changes compared to version 1.3.8

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>11.2.2 Instrument snapshot Message
11.3.1 Depth Snapshot Message
11.3.4 Mass Instrument State Change Message
11.3.5 Instrument State Change Message</td><td>New Enumerations added for the field SecurityStatus(965)</td></tr>
<tr><td>11.3.1 Depth Snapshot Message
11.3.2 Depth Incremental Message</td><td>New Enumerations added for the field MDEntryType(269)</td></tr>
<tr><td>14.4 Index Code and Mapping Table</td><td>Index code and Index name for 5 Indices have changed
<table>
<tr><th>Existing Index Code</th><th>New Index Code</th><th>New Index Name</th></tr>
<tr><td>16</td><td>30</td><td>S&P BSE MidCap</td></tr>
<tr><td>17</td><td>31</td><td>S&P BSE SmallCap</td></tr>
<tr><td>6</td><td>32</td><td>S&P BSE Fast Moving Consumer Goods</td></tr>
<tr><td>9</td><td>33</td><td>S&P BSE Healthcare</td></tr>
<tr><td>5</td><td>34</td><td>S&P BSE Information Technology</td></tr>
</table></td></tr>
</table>

15.7 Changes compared to version 1.3.9

Section	Description
11.3.1 Depth snapshot message 11.3.2 Depth incremental message	Date type of field BidSize(134) and OfferSize(135) changed from uint32 to int64 New Fields Yield(236), PutYield(30630), CallYield(30631) added.

BSE - PUBLIC

14.4 Index Code and Mapping Table	14 new index code added as mentioned below

15.8 Changes compared to version 1.4

Section	Description
11.3.1 Depth Snapshot Message 11.3.2 Depth Incremental Message	New Enumeration added for the field *MDEntryType(269)= BlockDealReferencePrice* This price to be used for placing Block deal order. The orders placed shall be within ±1% of the Block reference price in the respective windows as per timings specified by Exchange
14.4 Index Code and Mapping Table	5 new index codes added.

15.9 Changes compared to version 1.4.1

Section	Description
11.3.1 Depth Snapshot Message 11.3.2 Depth Incremental Message	New Enumeration added for the field *MDEntryType(269)=* Upper Limit Execution Price and Lower Limit Execution Price

BSE - PUBLIC