

Comparison of Market Data streams offered by BSE

BSE offers multiple market data streams for its market participants. Based on the requirement, market participants are free to choose between any of the streams. The streams offered are

1. EOBI- Enhanced Order Book Interface
2. EMDI- Enhanced Market Data Interface
3. MDI- Market Data Interface

The Table below gives finer details of each stream.

The EOBI is the true Order-By-Order stream wherein each order entered in the exchange is sent out in the market data. As a contrast, EMDI does the same, but within the orders to a level defined by the exchange. Thus EOBI can be used to build the entire order book to all levels whereas EMDI can be used to build a book to a particular level.

Another significant difference is that EMDI has the quantity aggregated at a particular price, while EOBI has the original order quantity & aggregation at price levels is the member's responsibility.

Both EMDI and EOBI are high bandwidth market data streams. EOBI is significantly higher in terms of bandwidth (around 4-5 times of EMDI).

EMDI and EOBI are intended for High frequency trading applications.

MDI is the relatively slower stream, primarily intended for Terminal based Trading applications. MDI aggregates quantities and prices and is disseminated at regular intervals.

All the streams share a common format of data, which is the live-live concept. The data is sent on multiple multicast streams and the members need to listen to both and arbitrate between the two streams to identify the required data.

EMDI and MDI are FAST (FIX Adapted for Streaming) encoded while EOBI is based on binary structures to reduce the overheads of encoding-decoding.

Comparison between Market Data streams

Attribute	EOBI	EMDI	MDI
Price Level	Full depth is available	Changes up to the depth configured by the exchange	Changes up to the depth configured by the exchange
Event	Every order book update at any price level	Every order book update till the configured price level	Order book updates will be netted at an interval defined and consolidated picture would be generated
Price level Maintenance	Responsibility of the end user's application	Available in the message	Available in the message
Aggregation at price level	No aggregation is done	Aggregated book at price level is sent	Aggregated book at price level is sent
Time level aggregation	Not done	Not done	Events are aggregated at time interval and then sent out. The time interval is defined by the exchange
Multicast streams	Separate streams for incremental and recovery	Separate streams for incremental and recovery	Same stream for incremental and recovery
Incremental and Snapshot data	Every order book update is given separately as an incremental data, so the book maintenance is to be done on top of previous data. Recovery is to be initiated using snapshot for any missed sequence number	Every order book update is given separately as an incremental data, so the book maintenance is to be done on top of previous data. Recovery is to be initiated using snapshot for any missed sequence number	Order book netted at an interval is given in incremental data and snapshot as both of them are available on the same stream.
Identification of position of an order	Possible using the priority timestamp of the order	Cannot be if multiple orders are available on the same price point	Cannot be possible as the events are aggregated at a time interval

Functional aspects of market data

Attribute	EOBI	EMDI	MDI
Usage	Used for High Frequency Trading applications where full book may be required.	Used for High Frequency Trading applications where full book may <u>not be</u> required.	Used for Low frequency Terminal Based Trading applications.
Advantages	It is easier to find self-orders in the order book.	It is possible to find self-orders in the order book, but significant code changes are required	It is not possible to find self-orders in the order book
	Entire market book is available, enabling the application to take decisions accordingly	Market book is available to a particular level, which can be useful for applications that don't need to see the entire order book.	
	Members need to aggregate quantities at every price point for trading opportunities	Quantities are aggregated at every price point till a particular level.	
Bandwidth Utilisation	High	High	Low
Recovery	The current pending order book is replayed & should be read again to build the book	The current snapshot up to the configured level is replayed in a separate stream and should be used in case of missed packets	The current snapshot up to the configured level is replayed at regular interval and should be used in case of missed packets

Technical details of EMDI and EOBI

Scenario	EMDI	EOBI
To check the depth incremental message if a limit order is confirmed at a new price level	MDUpdateAction: NEW (0) MDEntryType: Bid(0)/Offer(1) MDEntryPx: Price of the order MDEntrySize: Qty of the order NumberOfOrders: Number of order at that price point MDPriceLevel: Book level Bid Size : Total Bid size for that instrument.	TemplateID: 13100 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side: Bid/Offer Price: Price of the order
To check the depth incremental message if a limit order is confirmed at an existing price level	MDUpdateAction: CHANGE (1) MDEntryType: Bid(0)/Offer(1) MDEntryPx: Price of the order MDEntrySize: Qty of the order NumberOfOrders: Number of order at that price point MDPriceLevel: Book level Bid Size : Total Bid size for that instrument.	TemplateID: 13100 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side: Bid/Offer Price: Price of the order
To check the depth incremental message if a limit order is deleted such that price level is not deleted	The depth incremental message should update the following fields: MDUpdateAction : CHANGE (1) MDEntryType: Bid(0)/Offer(1) MDEntrySize: Qty of the order MDEntryPx: Price of the order NumberOfOrders: Number of order at that price point MDPriceLevel: Book level Bid Size : Total Bid size for that instrument.	TemplateID: 13102 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side: Bid/Offer Price: Price of the order
To check the depth incremental message if a limit order is deleted such that price level is deleted	The depth incremental message should update the following fields: MDUpdateAction : DELETE(2) MDEntryType: Bid(0)/Offer(2) MDEntrySize: Qty of the order MDEntryPx: Price of the order NumberOfOrders: Number of order at that price point MDPriceLevel: Book level Bid Size : Total Bid size for that instrument.	TemplateID: 13102 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side: Bid/Offer Price: Price of the order
To check the depth incremental message if a limit order is modified to a new price point such that the earlier price point is removed	The depth incremental message should update the following fields for the previous price point: MDUpdateAction : Delete (2) MDEntryType: Bid (0) MDEntryPx: Price of the order MDPriceLevel: Book level	TemplateID: 13101 TrdRegTSTimePriority: Priority Timestamp of the order TrdRegTSPrevTime- Priority - Previous order priority timestamp PrevDisplayQty -

	<p>Bid Size :Total Bid size for that instrument.</p> <p>Also for the new price point MDUpdateAction :NEW (0) MDEntryType:Bid(0) MDEntrySize:Qty of the order MDEntryPx: Price of the order NumberOfOrders: Number of order at that price point MDPriceLevel:Book level Bid Size :Total Bid size for that instrument.</p>	<p>DisplayQty: Qty of the order Side:Bid/Offer PrevPrice Price: Price of the order</p>
<p>To check the depth incremental message if a limit order is modified to a new price point such that the earlier price point is not removed</p>	<p>The depth incremental message should update the following fields for the previous price point: MDUpdateAction :Change (1) MDEntryType:Bid (0) MDEntryPx: Price of the order MDPriceLevel:Book level Bid Size :Total Bid size for that instrument.</p> <p>Also for the new price point MDUpdateAction :NEW (0) MDEntryType:Bid(0) MDEntrySize:Qty of the order MDEntryPx: Price of the order NumberOfOrders: Number of order at that price point MDPriceLevel:Book level Bid Size :Total Bid size for that instrument.</p>	<p>TemplateID: 13101 TrdRegTSTimePriority: Priority Timestamp of the order TrdRegTSPrevTime- Priority - Previous order priority timestamp PrevDisplayQty - DisplayQty: Qty of the order Side:Bid/Offer PrevPrice Price: Price of the order</p>
<p>To check the depth incremental message if an incoming order completely matches with the top price point and the price point is removed</p>	<p>The depth incremental message should update the following fields: MDUpdateAction :New (0) MDEntry Type: Trade(2) MD EntryPX: Trade Price MD Entry Size:Traded Quantity Bid Size :Total bid size of specified instrument. Offer Size:Total Offer size of specified instrument. TradeCondition:Exchange Last (U) Aggressor Timestamp:Entry time of the incoming order that triggered the trade. Aggressor side: buy (1) Number of buy order:number of buy orders involved in this trade. Number of sell order:number of sell orders involved in this trade. Also</p>	<p>Following entries are sent for the passive order getting removed:</p> <p>TemplateID: 13102 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side:Bid/Offer Price: Price of the order</p>

	MDUpdateAction :Delete (2) MDEntryType:Offer (1) MDEntryPx: Price of the order MDPriceLevel:Book level	
To check the depth incremental message if an incoming order completely matches with the top price point and the price point is not removed	The depth incremental message should update the following fields: MDUpdateAction :New (0) MDEntry Type: Trade(2) MD EntryPX: Trade Price MD Entry Size:Traded Quantity Bid Size :Total bid size of specified instrument. Offer Size:Total Offer size of specified instrument. TradeCondition:Exchange Last (U) Aggressor Timestamp:Entry time of the incoming order that triggered the trade. Aggressor side :buy (1) Number of buy order:number of buy orders involved in this trade. Number of sell order:number of sell orders involved in this trade. Also MDUpdateAction :Change (1) MDEntryType:Offer (1) MDEntryPx: Price of the order MDEntrySize:Qty of the order MDPriceLevel:Book level	Following entries would be sent for the passive order getting matched TemplateID: 13101 TrdRegTSTimePriority: Priority Timestamp of the order TrdRegTSPrevTime- Priority - Previous order priority timestamp PrevDisplayQty - DisplayQty: Qty of the order Side:Bid/Offer PrevPrice Price: Price of the order
To check the depth incremental message if an incoming order completely matches with the multiple price point	For each price point there will be different depth incremental messages: The depth incremental message should update the following fields: MDUpdateAction: New (0) MDEntry Type: Trade(2) MD EntryPX: Trade Price MD Entry Size: Traded Quantity Bid Size :Total bid size of specified instrument. Offer Size: Total Offer size of specified instrument. Trade Condition: Exchange Last (U) Aggressor Timestamp: Entry time of the incoming order that triggered the trade. Aggressor side: buy(1) Number of buy order: number of buy orders involved in this trade. Number of sell order: number of sell orders involved in this trade.	Following entries would be sent for each passive order getting matched: TemplateID: 13102 TrdRegTSTimePriority: Priority Timestamp of the order DisplayQty: Qty of the order Side:Bid/Offer Price: Price of the order