Market Data Gate

The market data gate broadcasts the following information:

Feed	Data
System events	Various system events
Instrument definitions	The instruments available on the exchange and their trading status
Instrument statuses	Changes of the instrument's properties
Orders snapshots	Full order book
Orders incremental	Anonymous feed of orders
Book1 snapshots	Top of the book
Book1 incremental	Changes in the top of the book
Book5 snapshots	Aggregated order book with 5 depth
Book5 incremental	Changes in Book5
Book25 snapshots	Aggregated order book with 25 depth
Book25 incremental	Changes in Book25
Trades snapshots	Last trade
Trades incremental	Anonymous feed of trades

Transfer and encoding

Market data is transmitted as UDP multicast traffic. Market data receivers will be provided with a receiving interface IPv4 address and a multicast IPv4 address for each feed.

The UDP packet payload consists of:

- Header: the value of `MessageSequenceNo` (FIX tag 34) encoded as a `uint64` number (little endian)
- Message in FIX/FAST encoding

Messages are encoded according to the FAST 1.x.1 standard with FAST version 1.2 Extension Version 10. The provided XML templates should be used for decoding the messages.

Messages use Financial Information Exchange protocol (FIX) Version 5.0 Service Pack 2. However, some fields deviate from the FIX standard when necessary.

If the information to be sent doesn't fit in a single UDP packet, it's split to multiple FIX/FAST messages that are sent sequentially, as allowed by the FIX standard. `FirstFragment` and `LastFragment` fields can be used to determine that these messages are parts of a single update.

Notation

The `=>` sign in the Tag column indicates that this field is part of a repeated sequence.

Fields marked as `optional` can have no value in some messages. They are encoded as specified in the FAST standard. All fields that are not marked as optional are always present.

Fields marked as `const` are never encoded and always have the same value specified in the XML template.

Note that field names, types, tag names, optional and const attributes are specified in the XML template as well.

Snapshots and incremental updates

Connection to live market data feeds is usually done in the following steps:

1. Subscribe to the incremental feed and start receiving update messages
2. Subscribe to the snapshot feed, receive a snapshot for each instrument of interest, and unsubscribe from the snapshot feed
3. For each instrument, discard received incremental updates that have `RptSeq` smaller or equal to the `RptSeq` of the received snapshot for this instrument
4. Apply the remaining incremental updates to the corresponding snapshots
5. Keep updating the data as new incremental updates are received

The data is sent to the snapshot feeds at fixed time intervals. The data is sent for each instrument separately. The `TotNumReports` field present in each snapshot message can be used to determine that a snapshot for each available instrument has been received.

Compatibility notes

All messages contain reserved fields that are currently unused. They may become used in the future, allowing forward-compatible changes to the template.

In addition, new fields may be added to the end of each message. Parsers should ignore unparsed data that may be present at the end of the message.

Recovery gate

Lost packets can be requested at the recovery gate. The recovery gate can be accessed via the HTTP protocol.

Get packets

Endpoint URL: `/v1/[feed]`, where `[feed]` can be one of the following:

`[feed]`	Feed
orders-incremental	Orders incremental
trades-incremental	Trades incremental
book1-incremental	Book1 incremental
book5-incremental	Book5 incremental
book25-incremental	Book25 incremental

Other feeds are not available via the recovery gate.

GET parameters:

Parameter	Description
`from`	`MessageSequenceNo` of the first requested message
`count`	Number of requested messages

The number of messages that can be requested in a single request is limited. The time frame of available packets is also limited.

On success, the status code 200 will be returned. The body of the response will contain binary data of the requested packets with length-delimited encoding:

`[length of packet1] [data of packet1] [length of packet2] [data of packet2] ...`

Length is encoded as `uint64` in Little Endian. Packet data uses FIX/FAST and is identical to the UDP payload in the multicast feed.

If no packets matching the parameters were found or the requested packets are too old, the status code 404 is returned.

The most recent packets may not be available via the recovery gate. In this case the request should be retried after a delay.

On bad request, status code 400 is returned.

Header

Each message starts with the `StandardHeader`:

Tag	Field	Type & presence	Description
1128	`AppliedVersionId`	`string` (const)	FIX version ID
35	`MessageType`	`string` (const)	Message type
34	`MessageSequenceNo`	`uint64`	Sequential message number
52	`SendingTime`	`timestamp`	Sending time

The `MessageType` value corresponding to each message type is specified below in the description of each message.

Session level and application level

Heartbeat and Sequence reset are session level messages. Session level messages can be present in every feed.

Other messages are application level messages. Each message can only appear in the corresponding feed.

Messages reference

Version 2

SequenceReset

Message that is sent when the gate resets its message counter. The message following the sequence reset will have the specified `MsgSeqNum`.

`MessageType` = "4" (Sequence reset)

Tag	Field	Type	Description
36	`NewSequenceNo`	`uInt64`	Sequence number of the next message in the feed (FIX name: NewSeqNo)

Heartbeat

Message that is sent to a feed when there are no other messages for some time. The heartbeat message doesn't have any additional fields. Only fields of the standard header are present.

`MessageType` = "0" (Heartbeat)

OrdersIncrementalUpdate

Message about adding, updating, or deleting an order.

`MessageType` = "X" (Market Data - Incremental Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
268	`EntryCount`	`length`	Number of MDEntry in this message (FIX name: NoMDEntries)
=>83	`ReportSequenceNo`	`uInt64`	Report sequence ID (unique monotonically increasing ID, separately for each instrument) (FIX name: RptSeq)
=>279	`UpdateAction`	`enum`	Type of the incremental update. Not present if entry type is EmptyBook (FIX name: MDUpdateAction). Variants: `0` - New, `1` - Change, `2` - Delete
=>278	`Id`	`uInt64`	Order ID. Not present if entry type is EmptyBook (FIX name: MDEntryID)
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `0` - Buy, `1` - Sell, `J` - EmptyBook
=>48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)

Tag	Field	Type	Description
=>270	`Price`	`decimal`	Order price. Not present if entry type is EmptyBook (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Depends on the update type: for New - order size; for Change - remaining order size; for Delete - remaining size of the order before deletion. Not present if entry type is EmptyBook (FIX name: MDEntrySize)
=>40	`OrderType`	`enum`	Order type. Not present if entry type is EmptyBook (FIX name: OrdType). Variants: `2` - Limit, `1` - Market, `101` - PostOnly
=>59	`TimeInForce`	`enum`	Time in force. Not present if entry type is EmptyBook. Variants: `1` - GoodTillCancel (GTC), `3` - ImmediateOrCancel (IOC), `4` - FillOrKill (FOK)
=>5007	`DeleteReason`	`enum`	Reason of deletion. Only present when `UpdateAction` = 2 (Delete). Variants: `Fulfilled`, `OrderType`, `TimeInForce`, `CrossTrade`, `CancelRequest`, `ReplaceRequest`, `Liquidation`
=>1003	`TradeId`	`uInt64`	ID of the trade that caused this update. Only present for updates caused by order execution
=>31	`TradePrice`	`decimal`	Price of the trade that caused this update. Only present for updates caused by order execution (FIX name: LastPx)
=>32	`TradeSize`	`int32`	Size of the trade that caused this update. Only present for updates caused by order execution (FIX name: LastQty)
=>273	`TradingTimestamp`	`timestamp`	Time of the update (FIX name: MDEntryTime)
=>5005	`EndOfTransaction`	`boolean`	True if this is the last update for this transaction
=>5010	`TraceId`	`uInt64`	Trace Id for internal usage

OrdersSnapshot

Full list of active orders. According to the standard, if the book is empty, the snapshot will contain a single entry with `Type` = `EmptyBook`.

`MessageType` = "W" (Market Data - Snapshot / Full Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update

Tag	Field	Type	Description
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
83	`ReportSequenceNo`	`uInt64`	Value of `ReportSequenceNo` field of the incremental update included in this snapshot (FIX name: RptSeq)
911	`TotalReportCount`	`uInt32`	Total number of reports (can be used to determine that all reports have been received) (FIX name: TotNumReports)
48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
5010	`TraceId`	`uInt64`	Trace Id for internal usage
268	`EntryCount`	`length`	Number of MDEntry in this message (FIX name: NoMDEntries)
=>278	`Id`	`uInt64`	Order ID. Present unless Type is EmptyBook (FIX name: MDEntryID)
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `0` - Buy, `1` - Sell, `J` - EmptyBook
=>270	`Price`	`decimal`	Order price. Present unless Type is EmptyBook (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Remaining order size. Present unless Type is EmptyBook (FIX name: MDEntrySize)
=>1003	`TradeId`	`uInt64`	ID of the last trade for this order. Only present if there was at least one trade for this order

TradesIncrementalUpdate

Information about a trade.

`MessageType` = "X" (Market Data - Incremental Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
268	`EntryCount`	`length`	Number of MDEntry in this message (FIX name: NoMDEntries)
=>83	`ReportSequenceNo`	`uInt64`	Report sequence ID (unique monotonically increasing ID, separately for each instrument) (FIX name: RptSeq)

Tag	Field	Type	Description
=>279	`UpdateAction`	`enum`	Type of the incremental update (FIX name: MDUpdateAction). Variants: `0` - New, `1` - Change, `2` - Delete
=>278	`Id`	`uInt64`	Trade ID (FIX name: MDEntryID)
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `2` - Trade
=>48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
=>270	`Price`	`decimal`	Trade price (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Trade size (FIX name: MDEntrySize)
=>5009	`TradeType`	`enum`	Type of the trade. Variants: `Regular`, `Liquidation`, `Deleverage`
=>5004	`AggressiveSide`	`enum`	Side of the trade initiator. Variants: `Buy`, `Sell`
=>273	`TradingTimestamp`	`timestamp`	Time of the trade (FIX name: MDEntryTime)
=>5005	`EndOfTransaction`	`boolean`	True if this is the last update for this transaction
=>5010	`TraceId`	`uInt64`	Trace Id for internal usage

TradesSnapshot

Information about the last trade for this instrument.

`MessageType` = "W" (Market Data - Snapshot / Full Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
83	`ReportSequenceNo`	`uInt64`	Value of `ReportSequenceNo` field of the incremental update included in this snapshot (FIX name: RptSeq)
911	`TotalReportCount`	`uInt32`	Total number of reports (can be used to determine that all reports have been received) (FIX name: TotNumReports)
48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
5010	`TraceId`	`uInt64`	Trace Id for internal usage
268	`EntryCount`	`length`	Number of MDEntry in this message (at most 1 for trades snapshot, zero if there was no trades) (FIX name: NoMDEntries)
=>278	`Id`	`uInt64`	Trade ID (FIX name: MDEntryID)

Tag	Field	Type	Description
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `2` - Trade
=>270	`Price`	`decimal`	Trade price (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Trade size (FIX name: MDEntrySize)
=>5009	`TradeType`	`enum`	Type of the trade. Variants: `Regular`, `Liquidation`, `Deleverage`
=>5004	`AggressiveSide`	`enum`	Side of the trade initiator. Variants: `Buy`, `Sell`
=>273	`TradingTimestamp`	`timestamp`	Time of the trade (FIX name: MDEntryTime)

BookIncrementalUpdate

Message about adding, updating, or deleting an entry in the aggregated order book. Note that when entries are added or removed to an aggregate order book, only a single update message is sent. The listener should infer changes in the price levels of all entries below the place of modification.

`MessageType` = "X" (Market Data - Incremental Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
268	`EntryCount`	`length`	Number of MDEntry in this message (FIX name: NoMDEntries)
=>83	`ReportSequenceNo`	`uInt64`	Report sequence ID (unique monotonically increasing ID, separately for each instrument) (FIX name: RptSeq)
=>279	`UpdateAction`	`enum`	Type of the incremental update. Not present if entry type is EmptyBook (FIX name: MDUpdateAction). Variants: `0` - New, `1` - Change, `2` - Delete
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `0` - Buy, `1` - Sell, `J` - EmptyBook
=>48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
=>1023	`PriceLevel`	`uInt32`	Price level (from 1 to market depth). Not present if entry type is EmptyBook (FIX name: MDPriceLevel)
=>270	`Price`	`decimal`	Price at this price level. Not present if entry type is EmptyBook (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Total size of orders at this price level. Not present if entry type is EmptyBook (FIX name: MDEntrySize)

Tag	Field	Type	Description
=>273	`TradingTimestamp`	`timestamp`	Time of the update (FIX name: MDEntryTime)
=>5005	`EndOfTransaction`	`boolean`	True if this is the last update for this transaction
=>5010	`TraceId`	`uInt64`	Trace Id for internal usage

BookSnapshot

An order book aggregated by price depth. According to the standard, if the book is empty, the snapshot will contain a single entry with `Type` = `EmptyBook`.

`MessageType` = "W" (Market Data - Snapshot / Full Refresh)

Tag	Field	Type	Description
5006	`FirstFragment`	`uInt32`	True if this is the first fragment of the update
893	`LastFragment`	`uInt32`	True if this is the last fragment of the update
83	`ReportSequenceNo`	`uInt64`	Value of `ReportSequenceNo` field of the incremental update included in this snapshot (FIX name: RptSeq)
911	`TotalReportCount`	`uInt32`	Total number of reports (can be used to determine that all reports have been received) (FIX name: TotNumReports)
48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
5010	`TraceId`	`uInt64`	Trace Id for internal usage
268	`EntryCount`	`length`	Number of MDEntry in this message (FIX name: NoMDEntries)
=>269	`EntryType`	`enum`	Entry type (FIX name: MDEntryType). Variants: `0` - Buy, `1` - Sell, `J` - EmptyBook
=>1023	`PriceLevel`	`uInt32`	Price level (from 1 to market depth). Not present if entry type is EmptyBook (FIX name: MDPriceLevel)
=>270	`Price`	`decimal`	Price at this price level. Present unless Type is EmptyBook (FIX name: MDEntryPx)
=>271	`Size`	`int32`	Total size of orders at this price level. Present unless Type is EmptyBook (FIX name: MDEntrySize)

InstrumentStatus

Information about current properties of the instrument.

`MessageType` = "f" (Security Status)

Tag	Field	Type	Description
48	`InstrumentId`	`uInt32`	Unique numeric ID of the instrument (FIX name: SecurityId)
326	`TradingStatus`	`enum`	Trading status for this instrument (FIX name: SecurityTradingStatus). Variants: `2` - Halted, `17` - Normal, `101` - CancelOnly
5002	`SellPriceBand`	`decimal`	Minimum allowed sell price
5003	`BuyPriceBand`	`decimal`	Maximum allowed buy price
5008	`MarkPrice`	`decimal`	Mark price of the instrument
5010	`TraceId`	`uInt64`	Trace Id for internal usage
746	`OpenInterest`	`int64`	Open interest for this instrument
5011	`TimeUntilFunding` (since v2)	`timestamp`	Estimated time until the next funding
5012	`EstimatedFundingRate` (since v2)	`decimal`	Estimated rate for the next funding

InstrumentDefinition

Information about an instrument.

`MessageType` = “d” (Security definition)

Tag	Field	Type	Description
911	`TotalReportCount`	`uInt32`	Total number of reports (can be used to determine that all reports have been received) (FIX name: TotNumReports)
48	`InstrumentId`	`uInt32`	Unique numeric ID of the instrument (FIX name: SecurityId)
55	`Symbol`	`string`	Textual instrument ID
15	`PriceCurrency`	`string`	Currency used for price (FIX name: Currency)
120	`SettlementCurrency`	`string`	Currency code of settlement denomination (FIX name: SettlCurrency)
1079	`MaturityTime`	`timestamp`	Maturity time of the instrument (only if applicable)
969	`MinPriceIncrement`	`decimal`	Minimal price increment
5010	`TraceId`	`uInt64`	Trace Id for internal usage
1141	`FeedTypeCount`	`length`	Total number of market data feeds in the following list (FIX name: NoMDFeedTypes)
=>1022	`FeedType`	`string`	Feed identifier: “Orders”, “Trades”, “Book1”, “Book5”, “Book25” (FIX name: MDFeedType)
=>264	`MarketDepth`	`uInt32`	Market depth, e.g. 1, 5, 25. Only for book feeds

Tag	Field	Type	Description
=>1021	`BookType`	`enum`	Book type. Only for book feeds (FIX name: MDBookType). Variants: `1` - TopOfBook (1x1), `2` - PriceDepth (5x5, 25x25)

FundingEvent (since v2)

Information about a processed funding.

Tag	Field	Type	Description
5010	`TraceId`	`uInt64`	Trace Id for internal usage
48	`InstrumentId`	`uInt32`	Instrument ID (FIX name: SecurityId)
273	`TradingTimestamp`	`timestamp`	Time of the event (FIX name: MDEntryTime)
5013	`Rate`	`decimal`	Funding rate