

Moscow Exchange

Market Data Multicast

FIX/FAST Platform

User Guide

Moscow Exchange
Version 4.5
January 25, 2017

Contents

- 1. Overview..... 5
 - 1.1. Document History 5
 - 1.2. Streaming Data..... 8
 - 1.3. Incremental Messaging 8
 - 1.4. FIX Format 9
 - 1.5. FAST Compression 9
 - 1.6. Multicast Delivery 9
 - 1.7. Recovery 10
- 2. Getting Started with MOEX Market Data FIX/FAST Multicast Platform..... 11
 - 2.1. Basic Scenario – Connect before the Trade Day Started..... 11
 - 2.2. Connect after the Trade Day Started..... 11
 - 2.3. Incremental Feeds A and B Arbitration 12
- 3. Core Functionality..... 14
 - 3.1. Platform Architecture..... 14
 - 3.2. FAST Implementation..... 17
 - 3.2.1 Introduction..... 17
 - 3.2.2 Stop Bit Encoding 18
 - 3.2.3 Implicit Tagging 18
 - 3.2.4 Field Encoding Operators 19
 - 3.2.5 FAST Template..... 19
 - 3.2.6 Decoding overview 21
 - 3.2.7 Sample Template..... 22

3.3.	Data Feeds	25
3.3.1	Instruments Feed	26
3.3.2	Market Statistics, Orders, and Trades Feeds.....	26
3.3.3	Market Recovery Feeds.....	28
3.3.4	Trading Session Status and HeartBeat messages	28
3.3.5	TCP Replay.....	29
3.4.	Recovery.....	30
3.4.1	Market Recovery Overview	31
3.4.2	Recovering Data – Process	31
3.4.3	TCP Replay.....	33
4.	FIX Message Specification.....	34
4.1.	FIX Component Blocks	35
4.1.1	Standard Message Header	35
4.1.2	Standard Message Trailer.....	36
4.1.3	Instrument.....	36
4.1.4	Instrument Extension.....	39
4.1.5	Market Segment.....	39
4.2.	FIX Session-Level Messages.....	41
4.2.1	Logon (A).....	41
4.2.2	Logout (5).....	42
4.2.3	Heartbeat (0).....	42
4.3.	FIX Application-Level Messages.....	43
4.3.1	Security Definition (d).....	43

4.3.2	Security Status (f)	44
4.3.3	Trading Session Status (h)	46
4.3.4	Market Data Request (V)	47
4.3.5	Market Data - Snapshot/Full Refresh (W)	47
4.3.6	Market Data - Incremental Refresh (X)	56

1. Overview

This document describes the Moscow Exchange (identified as MOEX below) MOEX Market Data Multicast FIX/FAST Platform. This platform provides the new highly efficient mechanism for delivering MOEX Market Data to market data consumers. The mechanism utilizes the FIX protocol for messages structure and syntax, FAST protocol for optimization of data streaming, and UDP protocol for delivering data to multiple users efficiently.

MOEX Market Data Multicast FIX/FAST Platform includes the following aspects: streaming data, incremental messaging, FIX format, FAST compression, multicast delivery, and recovery.

1.1. Document History

Issue	Date	Description
1.0	May 25, 2011	Original version of this document
2.0	December 12, 2012	Clarifications added
3.3	April 08, 2013	Negotiated and REPO deals – specific fields added Message format changes to separate SECBOARD, Trading Status, and Trading Period in individual tags. Additional fields to support REPO with CCP, Closing Auctions, Discrete Auctions, Dark pool auctions, T+2 trading data New FAST compression template Improved readability and fixing document's errata
3.3.1	May 24, 2013	Fixing document errors and adding clarifications per users' feedback. Removing unused fields from document. Compression template has been corrected. Document has revision marks ON to highlight changes.
3.3.2	September 04, 2013	Updated specifications for units (lots or securities) that are used in trading volumes (271)
3.3.3	March 26, 2014	Added field, due to changes in the Listing Rules.

4.0.	December 26, 2014	<p>MFIX Market Data Multicast 4.0 code is based on unified with MFIX Transactional licensed library FIX Antenna C ++ version 2.9.</p> <ul style="list-style-type: none"> • Security Status messages are published in a separate ISF channel and removed from other incremental channels • FAST template change <ul style="list-style-type: none"> • Code change to eliminate temporal crossed book conditions in Order List channel • Aggregated Orderbook publishing latency greatly improved and is now equal to OLR and TLR channels • Aggregated Orderbook (OBR) channel publishes all price levels • MDEntryTime (273) and OrigTime (9412) fields are added to the aggregated Ordebook Channel to indicate the timestamp of last change • Order List Refresh (OLR) channel contains new DealNumber (9885) field indicating trade number that caused order change or deletion. • Market Statistics channel (MSR) contains new entry MDEntryType 269 = 'e', CXFlag (5154) indication prevention of uncovered trading for security • Market Data - Snapshot/Full Refresh (W) contains new field RouteFirst (7944) that marks first FAST message in a set of messages forming snapshot per security • TradingSessionID (336) field was moved outside of repeating group in the Snapshot/Full Refresh (W) message • New fields were added to the Security Definition (d) message: <ul style="list-style-type: none"> o QuoteText (9696) – COMMENTS field of native SECURITIES table o SettlFixingDate (9119) - the closing date of the shareholders' register o DividendNetPx (9982) – dividend value expressed in settlement currency • Added new trading period code for the Opening Auction (625=S and 326=119) • Multiple editings to improve readability, remove unused fields, add
------	-------------------	--

		<p>clarifications</p> <ul style="list-style-type: none"> Added link to a file with technical policies and limitations of TCP replay channel <p>New value added = '3' (Order exists) for tag 7017 VolumeIndicator</p>
	July 10, 2015	Platform Architecture section is updated with a brief description of the service FAST UDP multicast marketdata to participants in the stock and currency markets.
4.5	December 19, 2016	<p>MFIX Market Data Multicast version 4.5 is released for public testing. Expected production launch date is March 13, 2017.</p> <ul style="list-style-type: none"> Aggregated orderbook and aggregated orderbook snapshot (OBR, OBS) feeds are completely removed. OBR/OBS multicast groups are reserved for possible future use. Please use OLR feed to build aggregated data in your application FAST templates change DealNumber (9885) field is removed from Order List feeds (OLR, OLS) RefOrderID (1080) field is added to Trade List feeds (TLR, TLS) to indicate MDEntryID of OLR order being hit or taken with this trade. SendingTime (52) field now contains sending timestamps in MMDDHHSSmmmmmm format having microsecond precision. Data type is changed to uInt64 in all messages. Document is updated to reflect the above changes Additional comments added to the Processing Notes in section 4.3.6 Network Connectivity Guide section is moved to a separate document

4.5	January 25, 2017	<ul style="list-style-type: none"> To avoid confusion while adding leading zeroes to SendingTime (52) tag value in January-September, and to provide uniqueness of values for the nearest 83 years, the data format in FAST udp multicast 4.5 will be changed to uint64 in a form yyMMDDHHmmSSuuuuuu with fixed length of 18 digits, where yy – last pair of digits in the year number (17 for year 2017), MM – month number, DD – day number, HH – hour number, mm – number of minutes, SS – number of seconds, uuuuuu – fraction of a second rounded to microseconds. Each part of formatting template has fixed length. Zeroes are added where necessary. New field LastUpdateTime (779) has been added to MSR channel. Field format is yyMMDDHHmmSSuuuuuu as described above. The meaning of the field is a timestamp of the last processed transaction in the trading system, for which an incremental update to MSR channel, relative to the previous updated is published. While comparing data from MSR, OLR, TLR incremental updates you can now detect, which update contains most recent data by comparing tag 779 value of MSR channel to timestamps from tags 273 and 9412 in OLR and TLR channels. FAST compression template is changed for MSR/MSS channels Listed above changes and comments in document are highlighted in blue. Corrected description for MDEntryType=f, g
-----	------------------	--

1.2.Streaming Data

Streaming data is the model, which allows one to compose a continuous sequence of information of determinate length into one message. It is promote to decrease latency and provide very high volume data routing.

1.3.Incremental Messaging

Incremental data model clearly provides less wasteful on resources. Minimum numbers of instructions are needed to update the book: add, change, delete. An incremental approach sends only necessary data of market events and is intended to significantly reduce data content.

1.4.FIX Format

MOEX Market Data Multicast FIX/FAST Platform uses FIX message format for messages structure and syntax. Message fields are delimited using the ASCII 01 <SOH> character. They are composed of a header, a body, and a trailer.

For more information about used messages and tags, see section 4. FIX Message Specification .

1.5.FAST Compression

FAST is a binary compression algorithm used to purpose of the optimization of FIX messages. FAST benefits include reduced bandwidth and reduced latency. They are achieved at the expense of increased processing time and more complex processing algorithms. The FAST Protocol uses the following approaches to compact data messages: - implicit tagging;

- field encoding;
- presence map;
- stop bit;
- binary encoding.

These approaches assume that the structures of the transferred messages as well as encoding rules are agreed between the counter parties. This is usually done via the exchange of machine readable XML-based FAST templates.

For more information about FAST Implementation in MOEX Market Data Multicast, see section 3.2. FAST Implementation.

1.6.Multicast Delivery

Messages are disseminated over the UDP protocol, which allows the Platform to transfer a single packet to multiple destinations and provides lower than TCP transmission latency.

One FAST encoded FIX message does not occupy more than one UDP packet. This ensures the feed is optimized for bandwidth efficiency by reducing the impact of multiple network headers and provides support for FAST field encoding to utilize the full suite of operators including Increment and Copy. These operators will only be used across a set of messages within a single packet.

Currently MOEX Market Data Multicast FIX/FAST Platform does not send more than one FAST encoded FIX message in a UDP packet, but such possibility can be added in future releases.

To minimize confusion MOEX Market Data Multicast FIX/FAST Platform sends messages from different tables of the Trading System to different multicast groups.

1.7.Recovery

Rapid recovery is increasingly important as clients must be always in the market. Recovery processes are very useful for recipients to minimize the probability of a data loss.

MOEX Market Data Multicast FIX/FAST Platform provides data recovery in two ways:

- Market data recovery using market snapshots – suitable for the recovery of a large-scale data loss (i.e. late joiner or major outage);
- TCP Replay of the sent messages – suitable for the recovery of a small-scale data loss (in case when some messages are lost during the transfer).

2. Getting Started with MOEX Market Data FIX/FAST Multicast Platform

2.1. Basic Scenario – Connect before the Trade Day Started

In general, clients should start listening to MOEX Market Data Multicast FIX/FAST Platform some time before the trading day starts. This ensures that client will start receiving actual market data without performing any recovery process. The procedure is the following:

1. Download the actual multicast IP addresses configuration file from ftp. Configuration file is the XML-file describing the connectivity parameters (feeds multicast addresses, ports, etc.).
2. Download the FAST template from ftp. See section 3.2.5 for the description of the FAST template.
3. Start listening Incremental Feed(s) and sequentially apply received data.

2.2. Connect after the Trade Day Started

If client starts listening to MOEX Market Data Multicast FIX/FAST Platform sometime after the trading day started, it should keep the following procedure:

1. Download the actual multicast IP addresses configuration file from ftp. Configuration file is the XML-file describing the connectivity parameters (feeds multicast addresses, ports, etc.). Download the FAST template from ftp. See section 3.2.5 for the description of the FAST template.
2. Start listening Instrument Definitions feed to get a list of securities. In addition, IDF feed acts as a snapshot channel for the Instrument Status channel
3. Start listening required Orders, Statistics, Trades, Instrument Status feeds and queue received data.
4. Start listening corresponding, Orders Recovery, Statistics Recovery, Trades Recovery. For each instrument, receive snapshot where values of fields 369 and 83 for a given instrument are greater than minimal values of corresponding fields 34 and 83 in the queued updates for that instrument.
5. Apply all updates where tags 34 and 83 are greater than in snapshot for selected instrument.
6. Continue receiving and normal processing incremental data for selected instrument.
7. Repeat steps 5-6 for all instruments you need. Alternatively, you can start queuing data until you get full snapshot cycle from message sequence number 1 to next snapshot cycle message with sequence number 1 and apply all updates for all instruments at once.

8. Stop listening Recovery Feed(s) when all needed instruments are in sync with incremental feed(s).

2.3.Incremental Feeds A and B Arbitration

Data in all UDP Feeds are disseminated in two identical feeds (A and B) on two different multicast IPs. It is strongly recommended that client receive and process both feeds because of possible UDP packet loss. Processing two identical feeds allows one to statistically decrease the probability of packet loss.

It is not specified in what particular feed (A or B) the message appears for the first time. To arbitrate these feeds one should use the message sequence number found in Preamble or in tag 34-MsgSeqNum. Utilization of the Preamble allows one to determine message sequence number without decoding of FAST message.

Processing messages from feeds A and B should be performed using the following algorithm:

1. Listen feeds A and B
2. Process messages according to their sequence numbers.
3. Ignore a message if one with the same sequence number was already processed before.
4. If the gap in sequence number appears, this indicates packet loss in both feeds (A and B). Client should initiate one of the Recovery process. But first of all client should wait a reasonable time, perhaps the lost packet will come a bit later due to packet reordering. UDP protocol can't guarantee the delivery of packets in a sequence.

Example:

Feed A		Feed B
34-MsgSeqNum = 59		34-MsgSeqNum = 59
34-MsgSeqNum = 60		34-MsgSeqNum = 60
34-MsgSeqNum = 62		34-MsgSeqNum = 61
34-MsgSeqNum =		34-MsgSeqNum = 62

63		
34-MsgSeqNum = 65		34-MsgSeqNum = 65

Messages received from Feed A and Feed B.

1. Receive message # 59 from Feed A, process it.
2. Receive message #59 from Feed B, discard it, because this message was processed before from Feed A.
3. Receive message # 60 from Feed A, process it.
4. Receive message #60 from Feed B, discard it, because this message was processed before from Feed A.
5. Receive message #62 from Feed A, discard it and wait for message #61.
6. Receive message # 61 from Feed B, process it.
7. Receive message # 62 from Feed B, process it.
8. Receive message #62 from Feed A, discard it, because this message was processed before from Feed B.
9. Receive message # 63 from Feed A, process it.
10. Receive message #65 from Feed A, discard it and wait for message #64.
11. Receive message #65 from Feed B, discard it and wait for message #64.
12. Begin recovery process, because gap is detected. Message #64 missed.

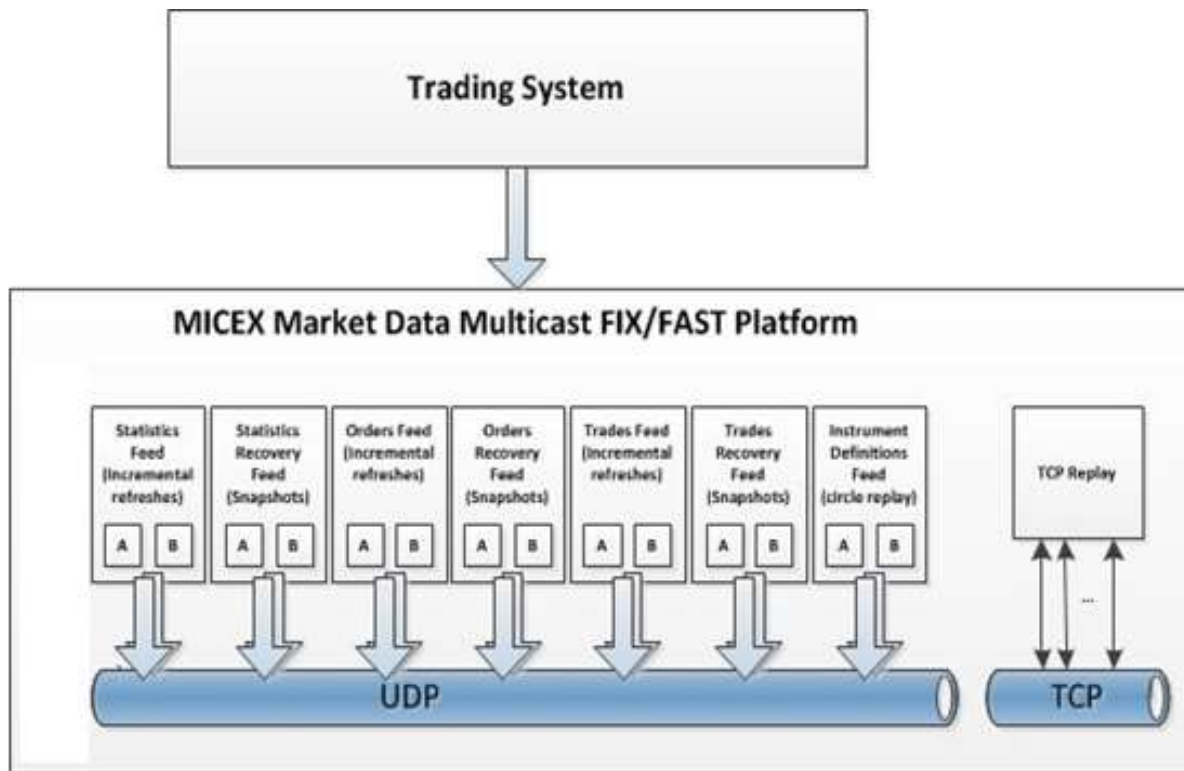
3. Core Functionality

3.1. Platform Architecture

UDP channels used to transfer market data from MOEX. UDP channels are also used for recovery process, TCP connection is used to replay sets of lost messages, already published in the one of UDP Channels.

Following feeds are used in the system:

1. Basic:
 - 1.1. Market Data Incremental Refresh feeds.
 - 1.2. Instrument Definition feeds.
2. Recovery:
 - 2.1. Market Recovery feeds.
 - 2.2. TCP Replay sessions.



MOEX Market Data Multicast broadcast feeds:

- Basic Feeds:
 - o Market Statistics Feeds (MSR)
 - Statistics Feed A
 - Statistics Feed B
 - o Active Orders List Feeds (OLR)
 - Orders Feed A
 - Orders Feed B

- o Trades List Feeds (TLR)
 - Trades Feed A
 - Trades Feed B
- o Instrument Status Feeds (ISF):
 - Status Feed A;
 - Status Feed B;
- Recovery Feeds:
 - o Market Statistics Recovery Snapshots Feeds (MSS)
 - Statistics Recovery Feed A
 - Statistics Recovery Feed B
 - o Active Orders List Recovery Snapshots Feeds (OLS)
 - Orders Recovery Feed A
 - Orders Recovery Feed B
 - o Trades List Recovery Snapshot Feeds (TLS)
 - Trades Recovery Feed A
 - Trades Recovery Feed B
- Instruments Definitions Feeds (IDF), also used as recovery feed for ISF feed:
 - o Instruments Definitions Feed
A
 - o Instruments Definitions Feed
B

Besides publishing market data in UDP channels, MOEX Market Data Multicast FIX/FAST Platform can accept TCP requests from clients. The replay of data from the following feeds can be requested over TCP connection:

- o Statistics Feed (MSR)
- o Orders Feed (OLR)
- o Trades Feed (TLR)
- o Instrument Status Feed (ISF)

There are some restrictions for market data transfer over TCP connection. Effective numeric values can be found in the TCP_Replay_Limits.pdf file located at <ftp://ftp.moex.com/pub/FAST/ASTS/config/> folder.

Moscow Exchange runs two instances of FAST UDP multicast market data feed. This allows switching to the available feed in case of issues with one of the instances. Both instances are statistically identical in publishing latency.

- FAST templates, server-side software, data contents, and MDEntryID fields identical for both services.
- FAST message sequence numbers for ASTS_FAST and ASTS_FAST2 services do NOT match.

3.2.FAST Implementation

This part contains the description of the implementation FIX Adapted for STreaming (FAST) protocol.

3.2.1 Introduction

The FIX Adapted for STreaming (FAST) Protocol has been developed as part of the FIX Market Data Optimization Working Group. FAST is designed to optimize electronic exchange of financial data, particularly for high volume, low latency data dissemination.

FAST is a data compression algorithm that significantly reduces bandwidth requirements and latency between sender and receiver. FAST works especially well at improving performance during periods of peak message rates. FAST extends the base FIX specification and assumes the use of FIX message formats and data structures. FAST is a standalone specification that uses templates to encode an instance of an application type, or part thereof, as a stream of bytes, and to inform the receiver which operations to use in decoding.

MOEX Market Data Multicast Platform distributes FIX messages which are encoded in FAST. The Preamble is found before the FAST encoded message, and contains the sequence number (Fig 1).

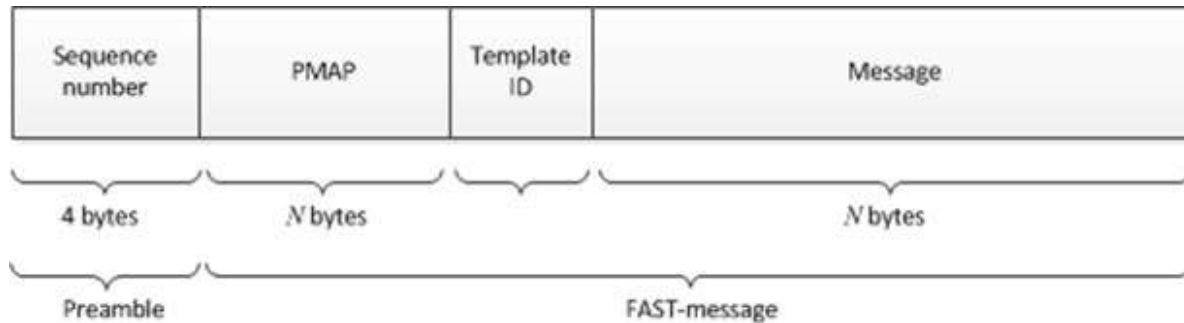


Figure 1

3.2.2 Stop Bit Encoding

An important property of the FAST transfer encoding is the use of stop bit encoded entities. In FAST, a stop bit is used instead of FIX's traditional <SOH> separator byte. Thus 7 bits of each byte are used to transmit data and the eighth bit is used to indicate the end of a field.

3.2.3 Implicit Tagging

In traditional FIX messages each field takes the form "Tag=Value<SOH>" where the tag is a number representing which field is being transmitted and the value is the actual data content. The ASCII <SOH> character is used as a byte delimiter to terminate the field. For example: 35=x|268=3 (message header)

279=0|269=2|270=9462.50|271=5|48=800123|22=8 (trade)

279=0|269=0|270=9462.00|271=175|1023=1|48=800123|22=8|346=15 (new bid 1)

279=0|269=0|270=9461.50|271=133|1023=2|48=800123|22=8|346=12 (new bid 2)

FAST eliminates redundancy with a template that describes the message structure. This technique is known as implicit tagging as the FIX tags become implicit in the data. A FAST template replaces the tag=value syntax with "implicit tagging" as follows:

- tag numbers are not present in the message but specified in the template
- fields in a message occur in the same sequence as tags in the template
- the template specifies an ordered set of fields with operators.

3.2.4 Field Encoding Operators

FAST functions as a state machine and must know which field values to keep in memory. FAST compares the current value of a field to the prior value of that field and determines if the new value should be constant, default, copy, delta (integer or string), increment, or tail.

Some operators rely on a previous value. A dictionary is a cache in which previous values are maintained. All dictionary entries are reset to the initial values specified after each UDP packet. Currently, MOEX sends one message per UDP packet. In this realization delta is not needed.

A field within a FAST template will generally have one of the Field Operators: Constant, Default, Copy, Delta, Increment.

A field within a FAST template will have one of the following Data Types: String, Signed Integer, Unsigned Integer, byte Vector, and Decimal.

3.2.5 FAST Template

A FAST template corresponds to a FIX message type and uniquely identifies an ordered collection of fields. The template also includes syntax indicating the type of field and transfer decoding to apply. A template is communicated between MOEX and client systems in XML syntax using the FAST v1.1 Template Definition Schema maintained by FIX. The XML format is human- and machine-readable and can be used for authoring and storing FAST templates. Session Control Protocol (SCP) will not be used.

A template consists of Field Instructions that define the fields contained in the message. Field Instructions specify the field name, tag number, data type, field operator, and presence attribute that indicate if a field is optional or mandatory.

A sample market data template is shown below (Fig. 2). The syntax is standard XML and can be parsed using a variety of open source tools. Valid template syntax is determined by the FAST Template Schema which is available in the FAST v1.1 specification.

```

103 <!-- Market Data - Incremental Refresh -->
104 <template name="X" id="6" xmlns="http://www.fixprotocol.org/ns/fast/td/1.1">
105   <string name="MessageType" id="35">
106     <constant value="X"/>
107   </string>
108   <string name="ApplVerID" id="1128"><copy/></string>
109   <string name="SenderCompID" id="49"><copy/></string>
110   <uInt32 name="MsgSeqNum" id="34"><increment/></uInt32>
111   <uInt64 name="SendingTime" id="52"><copy/></uInt64>
112   <byteVector name="MessageEncoding" id="347" presence="optional"><default/></byteVector>
113   <sequence name="GroupMDEntries">
114     <length name="NoMDEntries" id="268"/>
115     <uInt32 name="MDUpdateAction" id="279"><copy/></uInt32>
116     <string name="MDEntryType" id="269" presence="optional"><copy/></string>
117     <byteVector name="MDEntryID" id="278" presence="optional"><copy/></byteVector>
118     <byteVector name="Symbol" id="55" presence="optional"><copy/></byteVector>
119     <int32 name="RptSeq" id="83" presence="optional"><copy/></int32>
120     <decimal name="MDEntryPx" id="270" presence="optional"><copy/></decimal>
121     <decimal name="MDEntrySize" id="271" presence="optional"><copy/></decimal>
122     <uInt32 name="MDEntryDate" id="272" presence="optional"><copy/></uInt32>
123     <uInt32 name="MDEntryTime" id="273" presence="optional"><copy/></uInt32>
124     <byteVector name="TradingSessionID" id="336" presence="optional"><copy/></byteVector>
125     <byteVector name="QuoteCondition" id="276" presence="optional"><copy/></byteVector>
126     <byteVector name="TradeCondition" id="277" presence="optional"><copy/></byteVector>
127     <uInt32 name="OpenCloseSettleFlag" id="286" presence="optional"><default/></uInt32>
128     <decimal name="NetChgPrevDay" id="451" presence="optional"><copy/></decimal>
129     <decimal name="Yield" id="236" presence="optional"><copy/></decimal>
130     <decimal name="AccruedInterestAmt" id="5384" presence="optional"><copy/></decimal>
131     <decimal name="ChgFromWAPrice" id="5510" presence="optional"><copy/></decimal>
132     <decimal name="ChgOpenInterest" id="5511" presence="optional"><copy/></decimal>
133     <int32 name="TotalNumOfTrades" id="6139" presence="optional"><copy/></int32>
134     <decimal name="TradeValue" id="6143" presence="optional"><copy/></decimal>
135     <int32 name="OfferNbOr" id="9168" presence="optional"><copy/></int32>
136     <int32 name="BidNbOr" id="9169" presence="optional"><copy/></int32>
137     <decimal name="ChgFromSettlmnt" id="9750" presence="optional"><copy/></decimal>
138     <int32 name="SumQtyOfBest" id="10503" presence="optional"><copy/></int32>
139     <string name="OrderSide" id="10504" presence="optional"><copy/></string>
140     <string name="OrdStatus" id="10505" presence="optional"><copy/></string>
141     <decimal name="OrdBalance" id="10506" presence="optional"><copy/></decimal>
142     <decimal name="OrdValue" id="10507" presence="optional"><copy/></decimal>
143     <decimal name="MinCurrPx" id="10509" presence="optional"><copy/></decimal>
144     <uInt32 name="MinCurrPxChgTime" id="10510" presence="optional"><copy/></uInt32>
145   </sequence>
146 </template>

```

Figure 2

3.2.6 Decoding overview

The FAST template contains the instructions to decode and reconstruct compressed message data into the FIX format and also supports repeating groups (sequences) that allow a single message to convey multiple instructions (i.e. book update, trade, high/low, etc.). Decoding process include following steps:

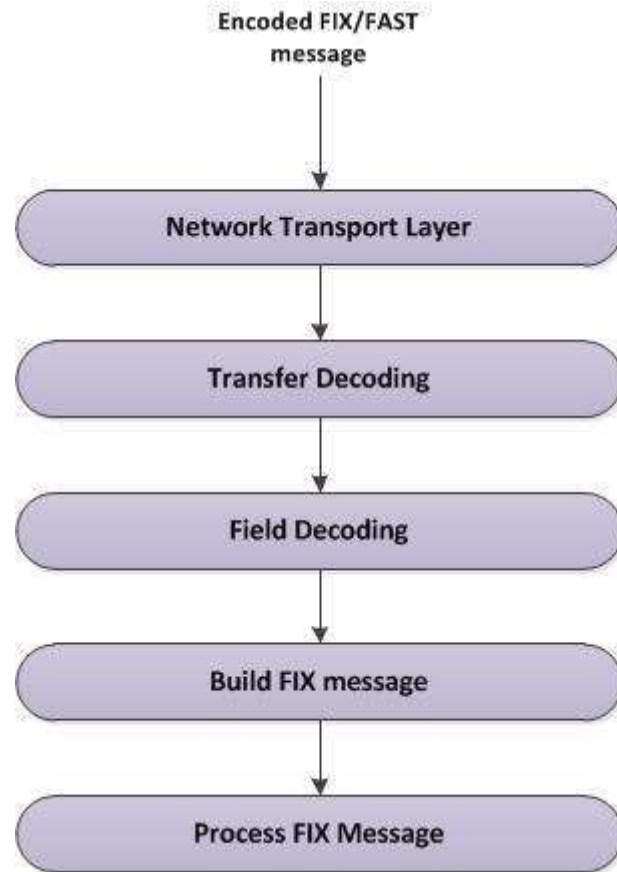


Figure 3

- Transport.
Client System receives encoded FAST message.
- Transfer decoding.

Transfer decoding is the initial step that converts data from the FAST 7-bit binary format. It includes:

- Identify template;
 - Extract binary encoded bits;
 - Map bits to fields per template.
- Field decoding.

Field decoding is the second part of the decompression process that reconstructs data values according to template-specified operations. Field decoding operations are assigned per field within the template; decoding reinstates data as indicated by the template.
 - Build FIX message.

It includes:

 - Decoding begins with the identification of the Pmap bit for each field.
 - The encoded FAST 7-bit binary values are obtained.
 - Then the encoded FAST 7-bit binary values are de-serialized based on the data type specified in the template.
 - The decoder maintains the state of prior values for each field throughout decoding and applies them for fields having operators of Delta, Copy, or Increment.
 - Obtain fully decoded values.
 - Process FIX message.

3.2.7 Sample Template

Table 1

LLi ne #	Template Syntax	Use and Description
1	<code><template name="X" id="6" xmlns="http://www.fixprotocol.org/ns/fast/td/1.1"></code>	Provides the template name and template identifier.
2	<code><string name="MessageType" id="35"> <constant value="X" /> </string></code>	Field instruction for MessageType defined as a string with identifier = 35 corresponding to the FIX tag number. MessageType has a constant field operator with a value of <i>X</i> which indicates the FIX message type—in this case Market Data Incremental Refresh.

3	<string name="AppVerID" id="1128"><copy/></string>	Field instruction for AppVerID defined as a string with an identifier = 1128 corresponding to the FIX tag number. AppVerID has a copy field operator.
4	<string name="SenderCompID" id="49"><copy/></string>	Field instruction for SenderCompID defined as a string with identifier = 49 corresponding to the FIX tag number. SenderCompID has a copy field operator.
5	<uInt32 name="MsgSeqNum" id="34"><increment/></uInt32>	Field instruction for MsgSeqNum defined as an unsigned integer with identifier = 34 corresponding to the FIX tag number. MsgSeqNum has an increment field operator.
6	<uInt64 name="SendingTime" id="52"><copy/></uInt64>	Field instruction for SendingTime defined as an unsigned integer and with identifier = 52 corresponding to the FIX tag number. SendingTime has a copy field operator.
7	<byteVector name="MessageEncoding" id="347"presence="optional"><default/></byteVector>	Field instruction for MessageEncoding defined as a byte vector and with identifier = 347 corresponding to the FIX tag number. MessageEncoding has a default field operator.
8	<sequence name="GroupMDEntries"> <length name="NoMDEntries" id="268"/>	Sequence instruction demarks the beginning of the MDEntries repeating group. The sequence includes a length field called 'NoMDEntries' that specifies the number of repeating groups present in the message.
9	<uInt32 name="MDUpdateAction" id="279" presence="optional"><copy/></uInt32>	Field instruction for MDUpdateAction defined as an unsigned integer and identifier = 279 corresponding to the FIX tag number. MDUpdateAction has a copy field operator.
10	<string name="MDEntryType" id="269" presence="optional"><copy/></string>	Field instruction for MDEntryType which is defined as a string and has an identifier = 269 which corresponds to the FIX tag number. MDEntryType has a copy field operator.
11	<byteVector name="MDEntryID" id="278" presence="optional"><copy/></byteVector>	Field instruction for MDEntryID which is defined as a byte vector and has an identifier = 278 which corresponds to the FIX tag number. MDEntryID has a copy field operator.
12	<byteVector name="Symbol" id="55" presence="optional"><copy/></byteVector>	Field instruction for Symbol which is defined as a byte vector and has an identifier = 55 which corresponds to the FIX tag number. Symbol has a copy field operator.
13	<int32 name="RptSeq" id="83" presence="optional"><copy/></int32>	Field instruction for RptSeq defined as a signed integer with identifier = 83 corresponding to the FIX tag number. RptSeq has a copy field operator.
14	<decimal name="MDEntryPx" id="270" presence="optional"><copy/></decimal>	Field instruction for MDEntryPx defined as a decimal with identifier = 270 corresponding to the FIX tag number. MDEntryPx has a copy field operator.
15	<decimal name="MDEntrySize" id="271" presence="optional"><copy/></decimal>	Field instruction for MDEntrySize defined as a decimal with identifier = 271 corresponding to the FIX tag number. MDEntrySize has a copy field operator.

16	<uInt32 name="MDEntryDate" id="272" presence="optional"><copy/></uInt32>	Field instruction for MDEntryDate defined as an unsigned integer and identifier = 272 corresponding to the FIX tag number. MDEntryDate has a copy field operator.
17	<uInt32 name="MDEntryTime" id="273" presence="optional"><copy/></uInt32>	Field instruction for MDEntryTime defined as an unsigned integer and identifier = 273 corresponding to the FIX tag number. MDEntryTime has a copy field operator.
18	<byteVector name="TradingSessionID" id="336" presence="optional"><copy/></byteVector>	Field instruction for TradingSessionID which is defined as a byte vector and has an identifier = 336 which corresponds to the FIX tag number. TradingSessionID has a copy field operator.
19	<byteVector name="QuoteCondition" id="276" presence="optional"><copy/></byteVector>	Field instruction for QuoteCondition which is defined as a byte vector and has an identifier = 276 which corresponds to the FIX tag number. QuoteCondition has a copy field operator.
20	<byteVector name="TradeCondition" id="277" presence="optional"><copy/></byteVector>	Field instruction for TradeCondition which is defined as a byte vector and has an identifier = 277 which corresponds to the FIX tag number. TradeCondition has a copy field operator.
21	<byteVector name="OpenCloseSettlFlag" id="286" presence="optional"><copy/></byteVector>	Field instruction for OpenCloseSettlFlag which is defined as a byte vector and has an identifier = 286 which corresponds to the FIX tag number. OpenCloseSettlFlag has a copy field operator.
22	decimal name="NetChgPrevDay" id="451" presence="optional"><copy/></decimal>	Field instruction for NetChgPrevDay defined as a decimal with identifier = 451 corresponding to the FIX tag number. NetChgPrevDay has a copy field operator.
23	<decimal name="AccruedInterestAmt" id="5384" presence="optional"><copy/></decimal>	Field instruction for AccruedInterestAmt defined as a decimal with identifier = 5384 corresponding to the FIX custom tag number. AccruedInterestAmt has a copy field operator.
24	<decimal name="ChgFromWAPrice" id="5510" presence="optional"><copy/></decimal>	Field instruction for ChgFromWAPrice defined as a decimal with identifier = 5510 corresponding to the FIX custom tag number. ChgFromWAPrice has a copy field operator.
25	<int32 name="TotalNumOfTrades" id="6139" presence="optional"><copy/></int32>	Field instruction for TotalNumOfTrades defined as a signed integer with identifier = 6139 corresponding to the FIX custom tag number. TotalNumOfTrades has a copy field operator.
26	<decimal name="TradeValue" id="6143" presence="optional"><copy/></decimal>	Field instruction for TradeValue defined as a decimal with identifier = 6143 corresponding to the FIX custom tag number. TradeValue has a copy field operator.

27	<decimal name="Yield" id="236" presence="optional"><copy/></decimal>	Field instruction for Yield defined as a decimal with identifier = 236 corresponding to the FIX tag number. Yield has a copy field operator.
28	<int32 name="OfferNbOr" id="9168" presence="optional"><copy/></int32>	Field instruction for OfferNbOr defined as a signed integer with identifier = 9168 corresponding to the FIX custom tag number. OfferNbOr has a copy field operator.
29	<int32 name="BidNbOr" id="9169" presence="optional"><copy/></int32>	Field instruction for BidNbOr defined as a signed integer with identifier = 9169 corresponding to the FIX custom tag number. BidNbOr has a copy field operator.
30	<string name="OrderSide" id="10504" presence="optional"><copy/></string>	Field instruction for OrderSide defined as a string with an identifier = 10504. OrderSide has a copy field operator.
31	<string name="OrderStatus" id="10505" presence="optional"><copy/></string>	Field instruction for OrderStatus defined as a string with an identifier = 10505. OrderStatus has a copy field operator.
32	<decimal name="MinCurrPx" id="10509" presence="optional"><copy/></decimal>	Field instruction for MinCurrPx defined as a decimal with identifier = 10509. MinCurrPx has a copy field operator.
33	<uInt32 name="MinCurrPxChgTime" id="10510" presence="optional"><copy/></uInt32>	Field instruction for MinCurrPxChgTime defined as an unsigned integer and identifier = 10510. MinCurrPxChgTime has a copy field operator.

3.3.Data Feeds

The use of incremental FIX market data messaging in combination with FAST compression produces highly optimized feeds which are distributed in UDP channels. Each Feed is transferred over separate multicast-address. Feeds have the following structure:

- o Statistics Feeds

- Statistics Feed A
- Statistics Feed B

- o Orders Feeds

- Orders Feed A
- Orders Feed B

- o Trades Feeds

- Trades Feed A
- Trades Feed B

- o Instrument Status Feeds

- Instrument Status Feed A

- Instrument Status Feed B
- o Instruments Feeds
 - Instruments Definitions Feed A
 - Instruments Definitions Feed B

In Feeds A and B the equal market data information is sent. It provides low probability of packets loss, and reduce the need in recovery processes.

3.3.1 Instruments Feed

Instruments Definitions Feed A/B provides the security main parameters in a Security Definition (d) message and changes to the definition and/or identity of the security. In this feeds FIX messages encoded to FAST are sent repeatedly with fixed time interval. One FIX message contains information about one security.

Message example:

```
8=FIXT.1.1|9=400|35=d|1128=9|34=1551|460=5|423=2|911=1572|49=MOEX|55=VRSBP|48=RU000A0DPG75|22=4|461=EPXX
XX|167=PS|
107=Voronezh EnergoSbyt.Comp(pref)|15=RUB|120=RUB|5217=2-01-55029-
E|5385=FOND|969=0.001|5508=0.4|7595=18716678|350=54|351="Воронеж.энергосб.комп"ОАО
ан|5382=20|5383=ВоронЭнСбп|52=2011050308:29:32.968|870=2|871=27|872=3|871=8|872=0|1310=1|561=1|1309=1|336=SMAL|
10=000|
```

Note: each security symbol (55) may be traded in several trading boards that differ by rules. Tag 336 indicates <Board>. There may be multiple different Board values for each security symbol. Please treat each combination of tags 55 and 336 in Security definition as a separate entity with separate stream of market data updates.

3.3.2 Market Statistics, Orders, and Trades Feeds

The following market data is also distributed in separate feeds:

- Statistics Feed A/B – market statistics, changes in SECURITIES table.
Statistics Feeds also include Add, Change, and Delete blocks. Entry types are:
'0' (Bid);

'1' (Offer);
'2' (Last Trade in Market statistics feed);
'3' (Index Value);
'4' (Opening Price);
'5' (Closing Price);
'7' (Trading Session High Price);
'8' (Trading Session Low Price);
'9' (Trading Session VWAP Price);
'A' (Imbalance)
'B' (Trade Volume, expressed in number of securities);
'J' (Empty book);
'N' (Session high bid);
'O' (Session low offer);
'Q' (Auction Clearing Price);
'W'(Closing auction price);
'c'(Closing auction volume);
'e'(Prevention of uncovered trading for security)
'i' (Last bid price);
'j' (Last offer price);
'h' (Open period price);
'k' (Close period price);
'l' (Market price 2); on FX market – FX fixing price as calculated between 11:59 and 12:00 Moscow time.
'm' (Market price); on FX market – FX fixing price
'o' (Official open price);
'p' (Official current price);
'q' (admitted quote); On FX market: international FX fixing price
'r' (Official close price);
'v' (Total bid volume);
'w' (Total offer volume);

's' (Dark pool Auction price)

'x' (Dark Pool Auction volume)

y'(Accrued interest amount per the unit of security at current date, expressed in rubles)

'u' (Duration);

- Orders Feed A/B – changes in ORDERS table.

Orders Feeds also include Add, Change, and Delete blocks. Entry types are: '0' (Bid), '1' (Offer), 'f' (buy market order in closing auction), 'g'(sell market order in closing auction), 'J' (Empty book)

- Trades Feed A/B – changes in TRADES table.

Trades Feeds include only Add block (MDUpdateAction(279) =0) and custom entry type MDEntryType (269) = 'z' (Trade List). 'J' – no trades per instrument.

The data is transmitted in a form of FIX-messages Market Data – Incremental Refresh (X) encoded into FAST format. Each message can include the updates of several financial instruments.

- Instrument Status feeds A/B – changes in security trading status which are published as Security Status (35=f) messages.

3.3.3 Market Recovery Feeds

Each Market Recovery feed (Statistics, Orders, Trades, Instrument Status) sends the Market Data Snapshot / Full Refresh (MsgType (35) = W) messages encoded to FAST. Each message contains the information about one security. Information in Market Data Snapshot / Full Refresh message includes status of the connection with the market (TradSesStatus (340) tag) and changes in status of a security (MDSecurityTradingStatus (1682) tag).

Market Recovery feeds should be used for recovery purposes only. Once the client system has retrieved recovery data, it recommended stopping listening to the Market Recovery feeds.

3.3.4 Trading Session Status and HeartBeat messages

Trading Session Status (h) message is used to represent connection status with appropriate MOEX market. When status of connection changed this message is sent into UDP channel.

If no updates are produced or in pauses between that snapshot cycles UDP multicast feeds periodically publish HeartBeat messages.

Trading Session Status and HeartBeat messages increment the feed message sequence number counter (34).

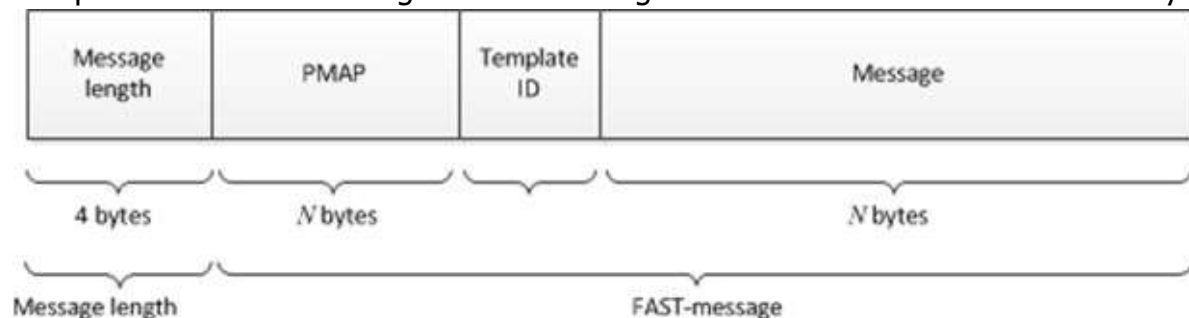
3.3.5 TCP Replay

The TCP replay component allows requesting a replay of a set of messages already published in one of UDP Channels.

The request is submitted by FIX Market Data Request message (35=V) with a range of sequence numbers and UDP Channel identifier.

After establishing TCP-session, client should send the FIX Logon message, always with sequence number 1. It is strongly recommended to use SenderCompID strings that allow client identification at firm level. When requesting the lost data client should specify the channel ID. Channel IDs can be found in MOEX Market Data Multicast FIX/FAST Platform configuration file available on ftp. They are OLR (for Order List feed), TLR (for Trade List feed), MSR (for Market Statistics feed), ISF (Instrument Status feed).

Only single data request is allowed per TCP/FIX session. After processing the request, the server sends requested FAST messages as tcp data stream. The length of the message in TCP stream can be found in 4-bytes number before each message being transmitted:



After sending all data the server initiates termination of session by sending FAST encoded Logout message and then waits for FIX Logout reply. After receiving reply the TCP connection is closed. It is an abnormal condition if client does not send confirming Logout within timeout period.

TCP Replay should be used in case of dropping small numbers of messages in both feed copies.

To limit the server load and network utilization by tcp replay traffic, the following technical limitations and policies are applied:

- Number of requested messages is limited. An attempt to request more messages will be rejected and followed by immediate logout message
- Number of simultaneous TCP sessions per source IP address is limited. An attempt to establish more TCP sessions is rejected
 - Number of TCP connections per day per source IP address is limited. Connection attempts after exceeding the limit are rejected
 - Total number of simultaneous TCP sessions is limited. Extra sessions are rejected.
- Limited waiting time for request and logout is applied. The session is terminated if waiting timeout is exceeded.

Effective numeric values can be found in the TCP_Replay_Limits.pdf file located at <ftp://ftp.moex.com/pub/FAST/ASTS/config/> folder.

3.4.Recovery

MOEX Market Data Multicast FIX/FAST Platform disseminates Market Data in all feeds over two UDP subfeeds: Feed A and Feed B. In Feeds A and B the identical messages are sent. It lowers the probability of packets loss and provides the first level of protection against missed messages.

Sometimes, messages may be missed on both feeds, requiring a recovery process to take place. Message loss can be detected using the FIX message sequence numbers (tag MsgSeqNum (34)), which are also found in the Preamble. The message sequence number is an incrementing number; therefore, if a gap is detected between messages in the tag MsgSeqNum (34) value, or the Preamble sequence number, this indicates a message has been missed. In addition, tag RptSeq (83) can be used to detect a gap between the messages at the instrument level. In this case client system should assume that market data maintained in it is no longer correct and should be synchronized to the latest state using one of the recovery mechanisms.

MOEX Market Data Multicast FIX/FAST Platform offers several options for recovering missed messages and synchronizing client system to the latest state. Market Recovery process together with Instruments Replay Feed is the recommended mechanism for recovery. TCP Replay provides less performance mechanism which is recommended only for emergency recovering of a small amount of lost messages when other mechanisms cannot be used for some reason. Instrument level sequencing and natural refresh can be utilized to supplement the recovery process. Notes:

- We strongly recommend that client systems process both the A and B Incremental UDP feeds. UDP Feed A and UDP Feed B provide the first level of protection against missed messages.
- We recommend Market Recovery as a primary recovery option.

3.4.1 Market Recovery Overview

This recovery method is preferable to use for a large-scale data recovery and for late joiners. Recovery feeds contains Market Data - Snapshot/Full Refresh (W) messages.

The sequence number (LastMsgSeqNumProcessed(369)) in the Market Data - Snapshot/Full Refresh (W) message corresponds to the sequence number (MsgSeqNum(34)) of the last Market Data - Incremental Refresh (X) message of a given instrument in the corresponding feed. Note that these values are **different** for different instruments.

Instrument level sequence number (RptSeq(83)) in Market Data - Snapshot/Full Refresh (W) message correspond to the sequence number (RptSeq(83)) in the MDEntry from the last Market Data - Incremental Refresh (X) message. Thus tag MsgSeqNum(34) shows the gap at the messages level, tag RptSeq(83) shows the gap at the instrument level.

After value of RptSeq(83) tag from Market Data - Incremental Refresh (X) becomes more than value of RptSeq(83) tag from Market Data - Incremental Refresh (X), market data becomes in sync with market.

After value of MsgSeqNum(34) from Market Data - Incremental Refresh (X) message becomes more than value of tag LastMsgSeqNumProcessed(369) from Market Data - Snapshot/Full Refresh (W) message for a given instrument, market data becomes for this instrument becomes in sync with market.

Message sequence numbers start from #1 in Market Data - Snapshot/Full Refresh (W) messages in each cycle.

First Market Data - Snapshot/Full Refresh (W) message in a set of messages for an instrument in Recovery Feeds is marked by tag RouteFirst (7944)=Y.

Last Market Data - Snapshot/Full Refresh (W) message in a set of messages for an instrument in Recovery Feeds is marked by tag LastFragment (893) = 'Y'. So the snapshot is considered as obtained as soon as the message with tag LastFragment (893) = 'Y' is received and all snapshot messages with lower sequence numbers are received.

Clients should keep queuing a real-time data until all missed data is recovered. The recovered data should be applied prior to the queued data.

The steps during Recovery process corresponds to the steps 4 – 7 from point 2.2.

Since clients have retrieved recovery data, it is recommended to stop listening Market Recovery feeds.

3.4.2 Recovering Data – Process

The recovering data process should be applied to affected feeds only. Unaffected feeds can be processed as usual. The process can follow two paths: queuing current data while recovering or processing current data while recovering.

3.4.2.1.1 Queuing

This process implies the queuing the Incremental Market Data from Incremental Feeds while receiving Market Data Snapshots from Recovery Feeds. In order to avoid an excessive number of queued messages, it is recommended to process snapshots and apply the applicable incremental feed as the snapshots arrive.

1. Identify Feed(s) in which the client system is out of sync.
2. Listen to and queue the Incremental Market Data from the affected Feed(s).
3. Listen to the Market Recovery Feed corresponding to the affected Incremental Feed(s), receive and apply snapshots.
4. Verify that all snapshots have been received for a given Market Recovery feed, using one of the following approaches:
 - a. Message sequence numbers in each loop of snapshots start from 1. So to determine the end of the loop one can wait until the next message with 34-MsgSeqNum = 1 arrives.
 - b. Snapshots in the Recovery Feeds are sent in the same order as Security Definitions in Instruments Feed. Tag 7944 RouteFirst marks the first message in a set of messages forming snapshot per instrument. Tag 893-LastFragment in the W-message indicates whether it is the last fragment in a set of messages forming snapshot per instrument. Receiving all messages per instrument from tag 7944=Y to 893=Y ensures getting full snapshot for the instrument.
5. Apply all queued incremental data in the sequence, where
 - a. tag 34-MsgSeqNum (or the Preamble sequence number) is greater than the lowest value for tag 369-LastMsgSeqNumProcessed for a given instrument;

OR

- b. tag 83-RptSeq from the Market Data Incremental – Refresh (X) message is greater than the lowest value for tag 83-RptSeq from the Market Recovery feed for a given instrument.
6. Continue normal processing

3.4.2.1.2 Concurrent Processing

This process implies the possibility to resume normal processing of an instrument while other affected instruments are still being recovered.

1. Identify Feed(s) in which the client system is out of sync.
2. Listen to the Incremental Market Data from the affected Feed(s) and optionally attempt a natural refresh.
3. Listen to the Market Recovery Feed corresponding to the affected Incremental Feed(s)
4. For each instrument:

- a. compare tag 369-LastMsgSeqNumProcessed on the Market Recovery feed to tag 34-MsgSeqNum (or the Preamble sequence number) on the Incremental Market Data feed and verify that the value for tag 34-MsgSeqNum is not lower;

OR

- b. compare tag 83-RptSeq on the Market Recovery feed to tag 83-RptSeq on the Incremental Market Data feed and verify that the value for tag 83-RptSeq on the Incremental Market Data feed is not lower.

5. Continue normal processing

3.4.2.1.3. Instrument Level Sequencing

Market Data Incremental Refresh messages contain instrument sequence numbers (tag 83-RptSeq), in addition to message sequence numbers (tag 34-MsgSeqNum). Every repeating group instance of a market data entry contains an incremental sequence number (tag 83-RptSeq) that is associated with the instrument for which the data is present in the block.

Client systems can keep track of the instrument sequence number (tag 83-RptSeq) for every instrument by inspecting incoming data and determine whether there is a gap in the instrument sequence number.

- If there is a gap in the instrument sequence number, it indicates that data was missed for the instrument when message loss occurred.
- If there is no gap, the data can be used immediately, and it also indicates that the book for this instrument still has a correct, current state.

3.4.2.1.4. Natural Refresh

The client system must track the state of the book at all times with the FIX Market Data Incremental Refresh messages. It is possible, though not guaranteed, that a set of these book update messages can be used to construct the current, correct state of a book without prior book state knowledge. This process is called Natural Refresh. Prior to the beginning of a natural refresh, the entire book should be emptied. Natural refresh assumes no prior knowledge of a book state.

3.4.3 TCP Replay

If market data from Statistics, Orders, and Trades Feeds was missed, it can be recovered over the TCP historical replay component using the sequence number range. TCP Replay is a low performance recovery option and should be only used if other options are unavailable or for a small-scale data recovery. A number of messages which can be requested by client during TCP connection are limited.

TCP replay include the following:

1. Establish TCP connection with MOEX Market Data Multicast.
2. Send FIX Logon(A) message with sequence number 1 to the server. After successful authorization the server sends the FAST-encoded Logon(A) message. 553 (username) and 554 (password) here is one of the pairs user0\pass0, user1\pass1, user2\pass2
 Example: 8=FIXT.1.1|9=94|35=A|49=SimpleClient|56=MOEX|34=1|52=20150530-11:01:44|98=0|108=10|553=user0|554=pass0|1137=9|10=078|
3. Send Market Data Request (V) message with:
 - a. Tag ApplID (1180) - the channel ID (as specified in a server configuration file available on ftp: OLR, TLR, or MSR).
 - b. Range of sequence numbers - ApplBegSeqNum(1182) and ApplEndSeqNum (1183) tags.
 Example: 8=FIXT.1.1 | 9=91 | 35=V | 1128=9 | 49=SimpleClient | 56=MOEX | 34=2 | 52=20150530-11:01:44 | 1180=OLR | 1182=1000 | 1183=1400 | 10=077|

The server processes only a single valid Market Data Request (V). If the request is correct, the server sends FAST messages according to requested sequence numbers.

After server responses, the server sends FAST Logout (5) message.

If the request is incorrect, the server sends FAST Logout (5) message with reject reason.

If no request is received within maximum waiting interval, then the server sends FAST Logout (5) message with logout reason

After sending Logout message the server waits for confirming logout.

TCP connection is closed after receiving confirming logout or after maximum waiting time is reached.

Note: closing connection without sending confirming logout is considered as abnormal situation.

4. FIX Message Specification

This part contains the description of FIX 5.0 SP2 protocol messages, component blocks and fields which are supported by MOEX Market Data Multicast.

This specification is based on FIX 5.0 SP2 standard for application-level messages, FIXT 1.1 for session-level messages (<http://fixprotocol.org/>) and adapted to MOEX's purposes. It's assumed that users have basic knowledge about FIX standard.

Only messages, component blocks and fields which are described in this document are supported by MOEX Market Data Multicast. Note that all fields which are required or conditionally required by FIX 5.0 SP2 standard but absent in MOEX Interface specification are optional and **will be ignored by MOEX**. All field values which are valid according to FIX 5.0 SP2 standard but aren't described in this document will be considered as invalid and incoming messages with such values will be rejected.

4.1.FIX Component Blocks

4.1.1 Standard Message Header

Table 2

Tag	Field name	Req'd	Type	Valid values	Comments
8	BeginString	Y	String (8)	'FIXT.1.1'	Identifies beginning of new message and protocol version. Always unencrypted, must be first field in message.
9	BodyLength	C	Length		Message length, in bytes, forward to the CheckSum field. Always unencrypted, must be second field in message. Should be present if message sends in TCP Recovery service.
35	MsgType	Y	String (10)		Defines message type. Always unencrypted, should be the third tag in the message.
1128	AppVerID	Y	String (1)	'9' (FIX50SP2)	Specifies the service pack release being applied for application-level messages. Always unencrypted. Should be placed next to 35 tag.
49	SenderCompID	Y	String (12)		Assigned value used to identify the firm sending a message. Always unencrypted. If this message is sent to MOEX TCP replay server, SenderCompID may contain an arbitrary string.
56	TargetCompID	Y	String		Assigned value used to identify a receiving firm. Always unencrypted. If this message sent from MOEX, then it will contain USERID assigned to a trader by MOEX.

					If this message sent to MOEX, then it should contain the MOEX server identifier. This parameter is given by MOEX
34	MsgSeqNum	Y	SeqNum		Integer message sequence number.
52	SendingTime	Y	uInt64		<p>Time of message transmission (in UTC time zone) in the following format: yyMMDDHHmmSSuuuuuu with fixed length of 18 digits, where yy – last pair of digits in the year number (17 for year 2017), MM – month number, DD – day number, HH – hour number, mm – number of minutes, SS – number of seconds, uuuuuu – fraction of a second rounded to microseconds. Each part of formatting template has fixed length. Zeroes are added where necessary.</p> <p>Example: 52=170125080709000030 is for a timestamp of 2017-01-25 08:07:09.000030 in human-readable format.</p> <p>Please note that this format is also used in FIX messages of TCP replay channel.</p>
347	MessageEncoding	N	String(11)	'UTF-8' (Unicode)	Type of message encoding (non-ASCII characters). Required if any "Encoding" fields are used except ASCII

4.1.2 Standard Message Trailer

Table 3

Tag	Field name	Req'd	Type	Valid values	Comments
10	Checksum	Y	String(3)		Three byte, simple checksum. Always unencrypted, always last field in message.

4.1.3 Instrument

Table 4

Tag	Field name	Req'd	Type	Valid values	Comments
55	Symbol	Y	String(12)		Ticker symbol. The MOEX internal instrument identifier, SecCode. Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.
48	SecurityID	N	String		Security identifier value of SecurityIDSource (22) type. (for example CUSIP, SEDOL, ISIN etc).
22	SecurityIDSource	N	String	'4' (ISIN)	Identifies class or source of the SecurityID (48) value. Field is obligatory if Security ID (48) is specified.
460	Product	N	int	'3' (CORPORATE); '4' (CURRENCY); '5' (EQUITY); '6' (GOVERNMENT); '7' (INDEX); '9' (General collateral certificate) '10' (MORTGAGE) '11' (MUNICIPAL); '12' (OTHER); '13' (FINANCING).	Indicates the type of product the security is associated with.
461	CFICode	N	String		Indicates the type of security using ISO 10962 standard, Classification of Financial Instruments (CFI code) values.

167	SecurityType	N	String	'CORP' (Corporate Bond); 'FOR' (Foreign Exchange Contract); 'CS' (Common Stock); 'PS' (Preferred Stock); 'EUSOV' (Euro Sovereigns); 'BN' Bank Notes; 'MF' Mutual Fund 'MUNI' (Municipal bonds); 'RDR' – (Russian depository receipt) 'ETF' – (Exchange traded fund); 'COFP' (Certificate Of Participation); 'XCN' (Extended Comm Note); 'STRUCT' (Structured Notes); 'WAR' (Warrant) 'GCD' (General collateral certificate)	Indicates type of security.
541	MaturityDate	N	LocalMktDate		Maturity date for bonds
224	CouponPaymentDate	N	LocalMktDate		Date interest is to be paid.
223	CouponRate	N	Price		Value of the due coupon payment, expressed in the currency of settlement
107	SecurityDesc	N	String		Security description. This tag contains security description in English on MOEX.
350	EncodedSecurityDescLen	N	Length		Byte length of encoded (non-ASCII characters) EncodedSecurityDesc (351) field.
351	EncodedSecurityDesc	N	data		Russian language (non-ASCII characters) name for the security. Encoded format is specified via the MessageEncoding (347) field. If used, the ASCII (English) representation should also be specified in the SecurityDesc (107) field.
5217	StateSecurityID	N	String		State Securities Identification Number.
5382	EncodedShortSecurityDescLen	N	Length		Byte length of encoded (non-ASCII characters) EncodedShortSecurityDesc (5383) field.

5383	EncodedShortSecurityDesc	N	data		Short (non-ASCII characters) security name in Russian language. Field encoding format specified via the MessageEncoding (347) field. in message header.
5556	BaseSwapPx	N	Price		Base SWAP price.
5558	BuyBackPx	N	Price		Base price for yield calculation. If indicated then the yield is calculated using this price. If defined, the field BuyBackDate appears in stream.
5559	BuyBackDate	N	LocalMktDate		Date for yield calculation. If indicated then the yield is calculated for this date

4.1.4 Instrument Extension

Table 5

Tag	Field name	Req'd	Type	Valid values	Comments
870	NoInstrAttrib	N	NumInGroup		Number of repeating InstrAttribType (871) entries.
=> 871	InstrAttribType	N	int	'8' (Coupon period); '27' (Instrument Price Precision).	The code to represent the type of instrument attribute. Required if NoInstrAttrib (870) > 0.
=> 872	InstrAttribValue	N	String		Attribute value appropriate to the InstrAttribType (871) field.

4.1.5 Market Segment

Table 6

Tag	Field name	Req'd	Type	Valid values	Comments
1310	NoMarketSegments	N	NumInGroup		Number of Market Segments on which a security may be traded.
=> 561	RoundLot	N	Qty		The trading lot size of a security.

=> 1309	NoTradingSessionRules	N	NumInGroup		Allows trading rules to be expressed by trading session.
=> => 336	TradingSessionID	N	String(4)		Identifier for trading board. Used to represent SECBOARD. Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.
=> => 625	TradingSessionSubID	N	String	'NA' – No trading 'O' – Opening period 'S' - Opening auction period 'C' – Closing period 'N' – Normal trading period 'L' – Closing auction period 'I' – Discrete auction period 'D' – Dark pool auction period 'E' – Trading at the closing auction price period 'A' - Auction: Order entry phase 'a' - Auction: Trade conclusion phase 'b' - Auction: Bookbuilding phase, orders are locked 'p'- Auction: After auction trade phase	Indicates the trading period Notes: <ul style="list-style-type: none"> • Period is empty before the trading start and after the trading is closed. • Switching between periods typically involves a short stop in trading, in which the period is not defined (625=NA) • The sequence and schedule of periods depends on board code and on market conditions as defined by the Exchange Trading rules. • Period value of this component block indicates a period that is running at the start of Security definition publishing cycle. Security status updates that come after Security definitions publishing cycle start should replace tag 625 values from Security definitions feed.
=> => 326	SecurityTradingStatus	N	int	18 – Not available for trading 118 – Opening period 119 - Opening auction period 18 – Trading closed 103 – Closing period 2 – Break in trading 17 – Normal trading 102 – Closing auction 106 – Dark pool auction	Trading status for a security Notes: <ul style="list-style-type: none"> • a break in any period is indicated by 326=2 and period identifier in tag 625. • Not available for trading and Trading Closed are different technological states in the Trading system. However they both disable trading activity and thus have equal values of tag 326.

				107 – Discrete auction 120 – Trading at Closing auction price	<ul style="list-style-type: none"> Trading status value of this component block indicates a trading status that existed at the start of Security definition publishing cycle. Security status updates that come after Security definitions publishing cycle start should replace tag 625 values from Security definitions feed.
=>=>9680	OrderNote	N	Char		Level of listing

4.2.FIX Session-Level Messages

4.2.1 Logon (A)

Logon message from customer to MOEX:

Table 7

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'A'
553	Username	Y*	String		Userid or username.
554	Password	Y*	String		User password.
1137	DefaultApplVerID	Y	String	'9' (FIX50SP2)	Specifies the service pack release being applied, by default, to message at the session level.

Note: it is strongly recommended to identify your firm via meaningful string in SenderCompID field of a standard message header in FIX TCP replay sessions.

Logon message from MOEX to customer:

Table 8

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'A'
108	HeartBtInt	Y	int		Heartbeat interval (seconds).
1137	DefaultApplVerID	Y	String	'9' (FIX50SP2)	Specifies the service pack release being applied, by default, to message at the session level.

4.2.2 Logout (5)

Table 9

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '5'
58	Text	N	String		Logout reason.

4.2.3 Heartbeat (0)

Table 10

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '0'

4.3. FIX Application-Level Messages

4.3.1 Security Definition (d)

Table 11

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'd'
911	TotNumReports	Y	int		Total number of Security Definition messages in a cycle.
component block <Instrument>		Y			The <Instrument> component block contains all the fields commonly used to describe a security or instrument.
component block <Instrument Extension>		N			The <InstrumentExtension> component block identifies additional security attributes that are more commonly found for Fixed Income securities.
15	Currency	N	Currency		Currency in which security is denominated
component block <Market Segment>		N			Contains all the security details related to listing and trading the security, including its trading status and trading period as they were at the start of Security Definitions publishing cycle. This allows late joiners to get current security trading state if they have missed earlier Security status (35=f) messages.
120	SettlCurrency	N	Currency		Currency code of settlement denomination.
423	PriceType	N	int	'1' (Percentage); '2' (Per unit).	Code to represent the price type. Note: for REPO with CCP this tag value is 1, but indicates the REPO rate, not the price of underlying security (bond or share)

64	SettlDate	N*	LocalMktDate		Specific date of trade settlement (SettlementDate) in YYYYMMDD format For Equities and FX in orders driven market: indicates settlement date For Equities in quote driven market (negotiated): indicates default settlement date. Actual date may vary and is indicated for each trade in the Trade List feed For FX swaps: indicates settlement date for first trade.
5385	MarketCode	N	String		Code of the market where instrument is traded. Note: MarketCode indicates a group of trading boards (SECBOARDS) with similar trading rules. MarketCode value may coincide with <Market> value of tag 335 but have a different purpose.
969	MinPriceIncrement	N	float		Minimum price increase for a given exchange-traded Instrument.
5508	FaceValue	N	Amt		Face value of security.
5850	OrigIssueAmt	N	Int		Number of placed securities in the issue
7595	NoSharesIssued	N	Qty		The number of issued shares.
9119	SettlFixingDate	N	Date		The record date for shareholders
9982	DividendNetPx	N	Numeric		Dividends, in the currency of payments
9696	QuoteText	N	Char		Comments

4.3.2 Security Status (f)

Security Status messages indicate the changes in the current Trading status and period for a security. Security Status messages are published in a separate ISF channel.

Note: to get current Security Status in a scenario of late join, please use the Instrument Definition feed as a snapshot channel for the fields 326 and 625, or use TCP replay channel.

Table 12

Tag	Field name	Req'd	Type	Valid values	Comments
-----	------------	-------	------	--------------	----------

<Standard Message Header>		Y			MsgType = 'f'
55	Symbol	Y	String		<p>Ticker symbol. The Moscow Exchange internal instrument identifier, SecCode.</p> <p>Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55) +TradingSessionID (336) combination as an individual security with its own order book and list of trades and orders.</p>
336	TradingSessionID	N	String		<p>Identifier for trading board. Used to represent SECBOARD.</p> <p>Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with own order book and list of trades.</p>
625	TradingSessionSubID	N	String	<p>'NA' – No trading 'O' – Opening period 'S' - Opening auction period 'C' – Closing period 'N' – Normal trading period 'L' – Closing auction period 'I' – Discrete auction period 'D' – Dark pool auction period 'E' – Trading at the closing auction price period 'A' - Auction: Order entry phase 'a' - Auction: Trade conclusion phase 'b' - Auction: Bookbuilding phase, orders are locked 'p'- Auction: After auction trade phase</p>	<p>Indicates the trading period</p> <p>Notes:</p> <ul style="list-style-type: none"> • Period is empty before the trading start and after the trading is closed. • Switching between periods typically involves a short stop in trading, in which the period is not defined (625=NA) • The sequence and schedule of periods depends on board code and market conditions as defined by the Exchange Trading rules.

326	SecurityTradingStatus	N	int	'18' – Not available for trading '118' – Opening period '119' - Opening auction period '18' – Trading closed '103' – Closing period '2' – Break in trading '17' – Normal trading '102' – Closing auction '106' – Dark pool auction '107' – Discrete auction '120' – Trading at Closing auction price	Trading status for a security Notes: <ul style="list-style-type: none"> a break in any period is indicated by 326=2 and period identifier is in tag 625. Not available for trading and Trading Closed are different technological states in the Trading system. However they both disable trading activity and thus have equal values of tag 326.
5509	AuctionIndicator	N	Boolean	'Y' (Yes); 'N' (No).	Indicates that the primary distribution auction is being held for the security. Primary distribution auction data is currently not published in the feed. Notes: <ul style="list-style-type: none"> 5509=N for ALL other auction types. Boolean values are encoded in FAST messages as binary integers: 1 for Y, and 0 for N.

4.3.3 Trading Session Status (h)

Table 13

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'h'
336	TradingSessionID	Y	String		Identifier for Trading Session is used to represent SECBOARD.

340	TradSesStatus	Y	int	'100' (Connection to MOEX market was established); '101' (Lost connection to MOEX); '102' (Connection to MOEX market was established, trading system wasn't restarted); '103' (Connection to MOEX market was established, trading system was restarted).	The state of the trading session. Informs about the connection state between the MOEX Market Data Multicast FIX/FAST Platform and the trading system. Note: Receiving the very unlikely message 340=103 means that Trading system has been started from scratch and you must remove all feed data on your side and start over.
58	Text	N	String		Free format text string.

4.3.4 Market Data Request (V)

Table 14

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'V'
1180	ApplID	N	String	OLR, TLR, MSR, ISF	The channel ID.
1182	ApplBegSeqNum	N	SeqNum		The beginning range of application sequence numbers.
1183	ApplEndSeqNum	N	SeqNum		The ending range of application sequence numbers. If Market Date Request is for one message then ApplBegSeqNum(1182) = ApplEndSeqNum(1183). If the request is for all messages after a specified message (but no more than a maximum number of sending messages), then ApplEndSeqNum(1183) = '0' (infinity).

4.3.5 Market Data - Snapshot/Full Refresh (W)

Table 15

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'W'

83	RptSeq	Y	int		Sequence number of a message within report series. Value equal to the RptSeq(83) in Market Data - Incremental Refresh (X) message at the time when the snapshot for a given instrument has been prepared.
369	LastMsgSeqNumProcessed	N	SeqNum		Value equal to the MsgSeqNum (34) from the last Market Data - Incremental Refresh (X) message which was published at the time of a snapshot for a given instrument has been prepared. Note: this field value may be different for different instruments within the same snapshot publishing cycle.
340	TradSesStatus	N	int	'100' (Connection to MOEX market was established); '101' (Lost connection to MOEX); '102' (Connection to MOEX market was established, trading system wasn't restarted); '103' (Connection to MOEX market was established, trading system was restarted).	The state of the trading session. Informs about the connection state between the MOEX Market Data Multicast FIX/FAST Platform and the trading system. Note: Receiving the very unlikely message 340=103 means that the Trading system has been started from scratch and you must remove all feed data on your side and start over.
55	Symbol	Y	String		Ticker symbol. The MOEX internal instrument identifier, SecCode. Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.
893	LastFragment	N	Boolean	'N' (Not Last Message); 'Y' (Last Message).	Indicates whether this message is the last in a sequence of messages in the snapshot for a security. Boolean values are encoded in FAST messages as binary integers: 1 for Y, and 0 for N.

7944	RouteFirst	N	Boolean	'Y' (the first message in a set of messages forming a snapshot for an instrument); 'N' (Not the first message in a set of messages forming a snapshot for an instrument).	Indicate that a message is the first in a set of messages forming a snapshot for an instrument.
1682	MDSecurityTradingStatus	N	int	18 – Not available for trading 118 – Opening period 119- Opening auction period 18 – Trading closed 103 – Closing period 2 – Break in trading 17 – Normal trading 102 – Closing auction 106 – Dark pool auction 107 – Discrete auction 120 – Trading at Closing auction price	Current trading status for a security Notes: <ul style="list-style-type: none"> • a break in any period is indicated by 1682=2 and period identifier is in tag 625 • Not available for trading and Trading Closed are different technological states in the Trading system. However, they both disable trading activity and thus have equal values of tag 326. • Switching between trading periods typically involves a short stop in trading in which the period is not defined (625=NA) • The sequence and schedule of periods and trading status values depends on SecBoard code (336) and on market conditions as defined by the Exchange Trading rules.
5509	AuctionIndicator	N	Boolean	'Y' (Yes); 'N' (No).	Indicates that the primary distribution auction is being held for the security. Primary distribution auction data is currently not published in the feed. Notes: <ul style="list-style-type: none"> • 5509=N for ALL other auction types. • Boolean values are encoded in FAST messages as binary integers: 1 for Y, and 0 for N.

451	NetChgPrevDay	N	PriceOffset		Net change from previous day's closing price vs. last traded price.
336	TradingSessionID	N	String		Identifier for trading board. Used to represent SECBOARD. Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.
268	NoMDEntries	Y	NumInGroup		Number of entries in Market Data message.
=> 269	MDEntryType	Y	char	'0' (Bid); '1' (Offer); '2' (Last Trade in Market statistics feed); '3' (Index Value \ Fixings); '4' (Opening Price); '5' (Closing Price); '7' (Trading Session (main or additional) High Price); '8' (Trading Session (main or additional) Low Price); '9' (Trading Session (main or additional) VWAP Price); 'A' (Imbalance), expressed in number of securities; 'B' (Trade Volume), expressed in number of securities; 'J' (Empty book); 'N' (Session (main or additional) high bid); 'O' (Session (main or additional) low offer);	Type Market Data entry. Notes: <ul style="list-style-type: none"> • The availability of this field's values depends on market type (FX or Equities), SecBoard code (336) and the Exchange trading rules. • Different feeds have subsets of possible values, depending on the data contents. • Empty Book (269=J) indicates no data for a security. Empty Book message may be generated market-wide, which indicates that you should remove all previously collected data and start over. • Meaning of some values depends on market type (FX or Equities) and corresponding trading rules

			<p>'Q' (Closing auction Clearing Price), the clearing volume (271) is expressed in lots; 'W' (Opening/Closing auction price); 'c' (Opening/Closing auction volume), expressed in number of securities; 'e' (prevention of uncovered trading for security) 'f' (market in opening auction period/ closing auction buy order); 'g' (market in opening auction period/ closing auction sell order); 'i' (Last bid price); 'j' (Last offer price); 'h' (Open period price); 'k' (Close period price); 'l' (Market price 2); on FX market – FX fixing price as calculated between 11:59 and 12:00 Moscow time. 'm' (Market price); on FX market – FX fixing price 'o' (Official open price); 'p' (Official current price); 'q' (admitted quote); On FX market: international FX fixing price; 'r' (Official close price); 'v' (Total (for main or additional session) bid volume); 'w' (Total (for main or additional session) offer volume); 's' (Dark pool Auction price) 'x' (Dark Pool Auction volume), expressed in number of securities 'y' (Accrued interest amount per the unit of security at current date, expressed in rubles) 'u' (Duration); 'z' (Trades list).</p>	<ul style="list-style-type: none"> • Off-book trading boards do not have data in OrderList snapshot feed (OLS). • Off-book trading boards may have market statistics data for a Symbol taken from on-book trading boards for this Symbol (market, current, WAP prices, etc.) • The set of field values may be extended following the Trading system updates. It is recommended to allow in your code ignoring unknown values of this field, and linked to such entry values of other fields, until the new field meaning can be supported by your application. • Indexes are published in Market statistics (MSR and MSS) channels for EQ market only. For FX market there are fixing values in 269=3 (also MSR\MSS channels). • Previous trading day values are indicated by additional tag 286
--	--	--	--	---

=> 278	MDEntryID	N	String		<p>Unique Market Data Entry identifier.</p> <p>Notes:</p> <ul style="list-style-type: none"> • For trades (269=z) entries, this tag contains a string with Exchange trade number that is equal to trade numbers in all trading interfaces • For OrderList (OLR, OLS channels), it contains a string identifier of Add Order (279=0) update for an order, NOT directly tied to the Exchange Order number in trading interfaces. • MDEntryID value for new Orders are growing but not always for a number up
=> 270	MDEntryPx	C	Price		<p>Price of the Market Data Entry.</p> <p>Conditionally required if MDEntryType (269) is not in ('A', 'B', 'J').</p>
=> 271	MDEntrySize	C	Qty		<p>Quantity represented by the Market Data Entry.</p> <p>Conditionally required if MDEntryType (269) is in ('0', '1', '2', 'A', 'B', 'Q', 'f', 'g', 'v', 'w').</p> <p>Note: For 269='A', 'B', this field value is expressed in number of securities. For all other values of tag 269, this field value is expressed in number of lots.</p>
=> 272	MDEntryDate	N	UTCDateOnly		<p>Date of Market Data Entry.</p>
=> 273	MDEntryTime	N	UTCTimeOnly		<p>Time of Market Data Entry.</p>

=> 625	TradingSessionSubID	N	String	'NA' – No trading 'O' – Opening period 'S' - Opening auction period 'C' – Closing period 'N' – Normal trading period 'L' – Closing auction period 'T' – Discrete auction period 'D' – Dark pool auction period 'E' – Trading at the closing auction price period 'A' - Auction: Order entry phase 'a' - Auction: Trade conclusion phase 'b' - Auction: Bookbuilding phase, orders are locked 'p'- Auction: After auction trade phase	Indicates the trading period For updates and snapshots, Period value indicates a period for an event reported, not necessarily the currently running period. In Trades feeds tag indicates a period when trade was completed
=> 276	QuoteCondition	N	MultipleValueString	'C' (Exchange Best)	Space-delimited list of conditions describing a quote.
=> 277	TradeCondition	N	MultipleValueString	'R' (Opening Price) ; 'AJ' (Official Closing Price); '98' (Minimum value); '99' (Maximum value).	Space-delimited list of conditions describing a trade.
=> 286	OpenCloseSettlFlag	N	MultipleValueString	'4' (Entry from previous business day)	Flag that identifies a market data entry.
=> 40	OrdType	N	Char	'1' (Market)	Order type. Used when MEntryType (269) = 'f', 'g' Note: Market in Closing Auction orders are activated and published in Order List feed in Closing Auction period. Matching occurs at the end of closing auction. Other market orders are not published in the feed because they never stay active.
=> 236	Yield	N	Percentage		Yield percentage.

=> 64	SettlDate	N*	LocalMktDate		Specific date of trade settlement (SettlementDate) in YYYYMMDD format Notes: For trades – settlement date of regular trade or negotiated deal. For REPO trades – settlement date of the first part of REPO.
=> 44	Price	N	Price		REPO rate for REPO trades.
=> 423	PriceType	N	int	'1' percentage	Indicates price type (REPO rate in percentage) for REPO trades.
=> 5154	CXFlag	N	Boolean	'Y' (Yes); 'N' (No).	Prevention of uncovered trading for security (269='e')
=> 5292	BidMarketSize	N	Int		Total volume of market buy orders calculated for currently expected auction price, expressed in number of securities. Used in closing auctions
=> 5293	AskMarketSize	N	Int		Total volume of market sell orders, expressed in number of securities used in closing auctions.
=> 5384	AccruedInterestAmt	N	Amt		Amount of accrued interest.
=> 5459	SettlType	N	Char		MOEX settlement code for trades (269=z)
=> 5510	ChgFromWAPrice	N	PriceOffset		Indicates the change from previous day's weighted average price vs. last traded price.
=> 5558	BuyBackPx	N	Price		For REPO deals - REPO value calculated in roubles for the current date (used in Trade List (269=z) feed).
=> 5559	BuyBackDate	N	LocalMktDate		For REPO deals - the date of the second part of REPO (used in Trade List (269=z) feed). Published as REPO buyback duration REPOTERM+ <Settledate>

=> 5677	Repo2Px	N	Price		Value of the 2nd (buy-back) REPO leg, expressed in settlement currency (used in Trade List (269=z) feed).
=> 5791	TotalVolume	N	Amt		Total volume. Used when MDEntryType (269)= 'f', 'g' Market in auction buy orders have money volume instead of lot quantity. Other orders use lot quantities.
=> 5902	EffectiveTime	N	UTSTimestamp		Order activation time. The order or price level with an activation time specified is not active until that time.
=> 9820	StartTime	N	UTSTimestamp		Auction start time. Used for Dark pool and Discrete auctions
=> 6139	TotalNumOfTrades	N	int		Total number of trades.
=> 6143	TradeValue	N	Amt		Trade Value.
=> 7017	VolumeIndicator	N	char	'0' (No orders) '1' (Total orders value is less than N*) '2' (Total orders value is greater than N*) '3' (Order exists)	Volume indicator of Dark Pool auction active orders. Used when MDEntryType(269)= 'v' or 'w'. N (variable)*- the large order volume factor as determined by the Exchange
=> 9168	OfferNbOr	N	int		Number of sell orders.
=> 9169	BidNbOr	N	int		Number of buy orders.
=> 9412	OrigTime	N	int		Indicates the microseconds portion of the transaction's registration time at the Matching engine. Should be added to tag's 273 value to get microsecond precision timestamp. The field is available in Orders and trades channels.
=> 10504	OrderSide	N	char		Side of aggressive order in TLR feed.

=> 1080	RefOrderID	N	char		In TLR feed: MDentryID of OLR order entry for an order being hit or taken.
=> 10505	OrderStatus	N	char	'O' (Active); 'T' (Order activation time hasn't come yet).	Describes the current state of order. Orders in T status are not active and not used in matching.
=> 10509	MinCurrPx	N	Price		Minimum current price. Used to determine condition when the short sales should be prohibited.
=> 10510	MinCurrPxChgTime	N	UTCTimeOnly		Time when minimum current price was changed.

4.3.6 Market Data - Incremental Refresh (X)

Important processing notes:

- Publishing massive updates in traffic-shaped feeds may take some time.
- Incremental updates are not always published per events in the trading system. Some intermediate states in very fast sequences of changes per security may be skipped and only resulting change is published in MSR or OLR feeds. TLR feed always contains all Add Trade messages.
- OLR feed does not publish updates for orders with zero lifetime: IOC, Market, FOK, and completely filled at registration time. OLR feed may skip publishing Add and Delete updates for an order if its active state duration is less than approximately 100 microseconds.
- MSR feed updates have LastUpdateTime timestamp with the meaning of last processed transaction time, after which the update was generated. It is guaranteed that MSR update with the LastUpdateTime equal to the maximum timestamp in tags 273 and 9412 of OLR or TLR update message corresponds to result of the same event in the trading system. By comparing timestamps (tags 779, 273 and 9412) in updates for MSR, OLR, TLR messages you can determine, which updates contain most recent data.
For massive events per an instrument like hitting multiple trades by single order, MSR feed is statistically published earlier than bulk OLR and TLR data in the majority of cases. However, the MSR precedence is not guaranteed. Using LastUpdateTime you can determine that MSR is definitely published ahead of OLR and TLR data and use MSR data for preliminary estimate of OLR and TLR changes.
For example, consider you receive MSR update with 779=170125080709000030 and with best sell price jumping up by 5 price steps. You have OLR state as of 273=2017-01-25 08:07:08 и 9412=999900. To get new OLR state estimate for a time of 273=2017-01-25 08:07:09 and 9412=30 you can remove all sell orders with prices less than new best sell price. You can correct estimate later after receiving OLR update. You can also estimate total trades count, volume, and last trade data for an instrument by MSR update if tag 779 value is greater than the last received TLR feed timestamp in tags 273 and 9412.
- For channels where add, change and delete MDUpdateActions are possible (Orders) the correct state is achieved after processing the whole set of repeating group entries in the message.
- FAST message length is limited by the network MTU size, current limitation is 1300 bytes. For massive updates, this results in splitting the data per several FAST messages.

Tag	Field name	Req'd	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'X'
779	LastUpdateTime	N	uInt64		<p>For MSR channel the meaning of the field is a timestamp of the last processed transaction in the trading system, for which an update is generated as a change from the previous state.</p> <p>While comparing data from MSR, OLR, TLR incremental updates you can now detect, which update contains most recent data by comparing tag 779 value of MSR channel to timestamps from tags 273 and 9412 in OLR and TLR channels.</p> <p>Field format is yyMMDDHHmmSSuuuuuu with fixed length of 18 digits, where yy – last pair of digits in the year number (17 for year 2017), MM – month number, DD – day number, HH – hour number, mm – number of minutes, SS – number of seconds, uuuuuu – fraction of a second rounded to microseconds. Each part of formatting template has fixed length. Zeroes are added where necessary.</p> <p>Example: 52=170125080709000030 is for a timestamp of 2017-01-25 08:07:09.000030 in human-readable format.</p>
268	NoMDEntries	Y	NumInGroup		Number of entries in Market Data message.

=> 279	MDUpdateAction	Y	uInt	'0' (New); '1' (Change); '2' (Delete).	Type of Market Data update action.
=> 269	MDEntryType	Y	char	'0' (Bid); '1' (Offer); '2' (Last Trade in Market statistics feed); '3' (Index Value\Fixings); '4' (Opening Price); '5' (Closing Price); '7' (Trading Session (main or additional session) High Price); '8' (Trading Session (main or additional session) Low Price); '9' (Trading Session (main or additional session) VWAP Price); 'A' (Imbalance), expressed in number of securities 'B' (Trade Volume, expressed in number of securities); 'J' (Empty book); 'N' (Session (main or additional session) high bid); 'O' (Session (main or additional session) low offer); 'Q' (Closing auction clearing Price), the clearing volume (271) is expressed in lots; 'W' (Opening/Closing auction price); 'c' (Opening/Closing auction volume), expressed in number of securities; 'e' (prevention of uncovered trading for security) 'f' (market in opening auction period/ closing auction buy order); 'g' (market in opening auction period/ closing auction sell order); 'i' (Last bid price); 'j' (Last offer price); 'h' (Open period price); 'k' (Close period price);	Type Market Data entry. Notes: <ul style="list-style-type: none"> • The availability of this field's values depends on market type (FX or Equities), SecBoard code (336) and the Exchange trading rules. • Different feeds have subsets of possible values, depending on the data contents. • Empty Book (269=J) indicates no data for a security. Empty Book message may be generated market-wide, which indicates that you should remove all previously collected data and start over. • Meaning of some values depend on market type (FX or Equities) and corresponding trading rules • Off-book trading boards do not have data in Orderbook Snapshot (OBS) and OrderList snapshot feeds (OLS). • Off-book trading boards may have market statistics data for a Symbol taken from on-book trading boards for this Symbol (market, current, WAP prices, etc.) • The set of field values may be extended following the Trading system updates. It is recommended to allow in your code ignoring unknown values of this field, and linked to such entry values of other fields, until the new field meaning can be supported by your application. • Indexes are published in Market statistics (MSR and MSS) channels for EQ market only. For FX market there are fixing values in 269=3 (also MSR\MSS channels). • Preious trading day values are indicated by additional tag 286

				<p>'l' (Market price 2); on FX market – FX fixing price as calculated between 11:59 and 12:00 Moscow time.</p> <p>'m' (Market price); on FX market – FX fixing price</p> <p>'o' (Official open price);</p> <p>'p' (Official current price);</p> <p>'q' (admitted quote); On FX market: international FX fixing price</p> <p>'r' (Official close price);</p> <p>'v' (Total (for main or additional session) bid volume);</p> <p>'w' (Total (for main or additional session) offer volume);</p> <p>'s' (Dark pool Auction price)</p> <p>'x' (Dark Pool Auction volume), expressed in number of securities</p> <p>'y' (Accrued interest amount per the unit of security at current date, expressed in rubles)</p> <p>'u' (Duration);</p> <p>'z' (Trades list).</p>	
=> 278	MDEntryID	N	String		<p>Unique Market Data Entry identifier. Used, for example, for TRADENO.</p> <p>Notes:</p> <ul style="list-style-type: none"> • For trades (269=z) entries, contains a string with Exchange trade number that is equal to trade numbers in all trading interfaces contains a unique string identifier of price level • For OrderList (OLR, OLS channels), contains a string identifier of Add Order (279=0) update for an order, NOT directly tied to the Exchange Order number in trading interfaces. • MDEntryID value for new Orders will be growing for a number up.

=> 55	Symbol	Y	String		<p>Ticker symbol. The MOEX internal instrument identifier, SecCode.</p> <p>Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.</p>
=> 83	RptSeq	Y	int		<p>Sequence number of a message within report series. Incremented by one for each update entry and for security status updates.</p>
=> 270	MDEntryPx	C	Price		<p>Price of the Market Data Entry.</p> <p>Conditionally required when MDUpdateAction (279) = New(0) and MDEntryType (269) not in ('A', 'B', 'J').</p>
=> 271	MDEntrySize	C	Qty		<p>Quantity represented by the Market Data Entry.</p> <p>Conditionally required when MDUpdateAction (279) = New (0) and MDEntryType (269) in ('0', '1', '2', 'A', 'B', 'Q', 'f', 'g', 'v', 'w').</p> <p>Note: For 269='A','B', this field value is expressed in number of securities. For all other values of tag 269, this field value is expressed in number of lots.</p>
=> 272	MDEntryDate	N	UTCDateOnly		Date of Market Data Entry.
=> 273	MDEntryTime	N	UTCTimeOnly		Time of Market Data Entry.
=> 336	TradingSessioID	N	String		<p>Identifier for trading board. Used to represent SECBOARD.</p> <p>Note: an instrument with a given SecCode may be traded in several trading boards (SecBoard). You should use each Symbol (55)+TradingSessionID (336) combination as an individual security with its own order book and list of trades.</p>

=> 625	TradingSessionSubID	N	String	'NA – No trading 'O – Opening period 'S - Opening auction period 'C – Closing period 'N – Normal trading period 'L – Closing auction period 'T – Discrete auction period 'D' – Dark pool auction period 'E' – Trading at the closing auction price period 'A' - Auction: Order entry phase 'a' - Auction: Trade conclusion phase 'b' - Auction: Bookbuilding phase, orders are locked 'p'- Auction: After auction trade phase	Indicates the trading period For updates and snapshots, Period value indicates a period for an event reported, not necessarily the currently running period. In Trades feeds tag indicates a period when trade was completed
=> 276	QuoteCondition	N	MultipleValueString	'C' (Exchange Best)	Space-delimited list of conditions describing a quote.
=> 277	TradeCondition	N	MultipleValueString	'R' (Opening Price) ; 'AJ' (Official Closing Price); '98' (Minimum value); '99' (Maximum value).	Space-delimited list of conditions describing a trade.
=> 286	OpenCloseSettleFlag	N	MultipleValueString	'4' (Entry from previous business day)	Flag that identifies a market data entry.
=> 40	OrdType	N	Char	'1' (Market)	Order type. Used when MDEntryType (269) = 'f', 'g' Note: Market in Closing Auction orders are activated and published in Order List feed in Closing Auction period. Matching occurs at the end of closing auction. Out of auction periods, market orders are not published in the feed because they never stay active.
=> 451	NetChgPrevDay	N	PriceOffset		Net change from previous day closing price vs. last traded price.
=> 236	Yield	N	Percentage		Yield percentage.

=> 64	SettlDate	N*	LocalMktDate		Specific date of trade settlement (SettlementDate) in YYYYMMDD format Notes: For trades – settlement date of regular trade or negotiated deal. For REPO trades – settlement date of first part of REPO.
=> 44	Price	N	Price		REPO rate for REPO trades
=> 423	PriceType	N	int	'1' percentage	Indicates price type (REPO rate in percentage) for REPO trades.
=> 5292	BidMarketSize	N	Int		Total volume of market buy orders calculated for currently expected auction price, expressed in number of securities.Used in closing auctions
=> 5293	AskMarketSize	N	Int		Total volume of market sell orders, expressed in number of securitiesUsed in closing auctions
=> 5384	AccruedInterestAmt	N	Amt		Amount of accrued interest.
=>5154	CXFlag	N	Boolean	'Y' (Yes); 'N' (No).	Prevention of uncovered trading for security (269='e')
=> 5459	SettlType	N	Char		MOEX settlement code for trades (269=z)
=> 5510	ChgFromWAPrice	N	PriceOffset		Indicates the change from previous day's weighted average price vs. last traded price.
=> 5558	BuyBackPx	N	Price		For REPO deals - REPO value calculated in roubles for the current date (used in Trade List (269=z) feed).
=> 5559	BuyBackDate	N	LocalMktDate		For REPO deals - the date of the second part of REPO (used in Trade List (269=z) feed). Published as REPO buyback duration REPOTERM+<Settledate>

=> 5677	Repo2Px	N	Price		Value of the 2nd (buy-back) REPO leg, expressed in roubles (used in Trade List (269=z) feed).
=> 5791	TotalVolume	N	Amt		Used when MDEntryType (269)= 'f', 'g' Market in auction buy orders have money volume instead of lot quantity. Other orders use lot quantities.
=> 5902	EffectiveTime	N	UTSTimestamp		Order activation time. The order or price level with an activation time specified is not active until that time.
=> 6139	TotalNumOfTrades	N	int		Total number of trades.
=> 6143	TradeValue	N	Amt		Trade Value.
=> 7017	VolumeIndicator	N	char	'0' (No orders) '1' (Less then N* minimum order value) '2' (Greater then N* minimum order value) '3' (Order exists)	Volume indicator of Dark Pool auction active orders. Used when MDEntryType(269)= 'v' or 'w'. N (variable)*- the large order volume factor as determined by the Exchange.
=> 9168	OfferNbOr	N	int		Number of sell orders.
=> 9169	BidNbOr	N	int		Number of buy orders.
=> 9412	OrigTime	N	int		Indicates the microseconds portion of the transaction's registration time at the Matching engine. Should be added to tag's 273 value to get microsecond precision timestamp. The field is available in Orders and Trades channels.
=> 9820	StartTime	N	UTSTimestamp		Auction start time. Used for Dark pool and Discrete auctions
=> 10504	OrderSide	N	char		Side of aggressive order in the TLR feed.
=> 1080	RefOrderID	N	char		In TLR feed: MDEntryID of OLR order entry for an order being hit or taken.
=> 10505	OrderStatus	N	char	'O' (Active); 'T' (Order activation time hasn't come yet).	Describes the current state of order. Orders in T status are not active and not used in matching.
=> 10509	MinCurrPx	N	Price		Minimum current price. Used to determine condition when the short sales should be prohibited.
=> 10510	MinCurrPxChgTime	N	UTCTimeOnly		Time when minimum current price was changed.

