Moscow Exchange FAST protocol specification for OTC trades report system (OTC-monitor)

Version 1.18.0

Moscow 2023

Table of Contents

1. Introduction ... 4
    1.1. Document purpose ... 4
    1.2. Fast Gate — Basic information ... 4
        1.2.1. Data streaming approach ... 4
        1.2.2. Incremental messages ... 4
        1.2.3. FIX format ... 4
        1.2.4. Encoding in the FAST format ... 4
        1.2.5. Data receicing via Multicast ... 4
        1.2.6. Data recovery ... 5
2. Scenarios of client interactions with Market Data Multicast ... 6
    2.1. Connect client ... 6
    2.2. Incremental Feeds A and B Arbitration ... 6
3. System functionality ... 8
    3.1. System architecture ... 8
        3.1.1. Main streams (UDP) ... 8
        3.1.2. Recovery streams (UDP) ... 8
        3.1.3. Instrument Definitions streams (UDP) ... 9
        3.1.4. Messages in streams ... 9
        3.1.5. Sessions for requesting missing messages (TCP) ... 9
    3.2. FAST format — details ... 10
        3.2.1. Stop bit encoding ... 10
        3.2.2. Implicit tagging ... 10
        3.2.3. Fields encoding options ... 10
        3.2.4. FAST-template ... 10
        3.2.5. Decoding overview ... 11
        3.2.6. Message fragmentation ... 11
        3.2.7. Data types ... 11
    3.3. Missing data recovery ... 11
        3.3.1. Recovery missing data using Recovery streams (UDP) ... 12
        3.3.2. Recovering missing data using TCP-connection ... 12
    3.4. Message sequence reset ... 12
    3.5. Message templates ... 13
        3.5.1. OtcMonitorIncrementalRefreshMessage ... 13
        3.5.2. OtcMonitorSnapshotMessage ... 13
        3.5.3. OtcMonitorSecurityDefinition ... 14
4. FIX protocol message specifications ... 15
    4.1. Field groups ... 15
        4.1.1. Standard Message Header ... 15
    4.2. Session layer messages ... 15
        4.2.1. Logon (A) ... 15
        4.2.2. Logout (5) ... 15
        4.2.3. Heartbeat (0) ... 16
        4.2.4. Sequence Reset (4) ... 16
    4.3. Business logic layer messages ... 16
        4.3.1. Security Definition (d) ... 16
        4.3.2. Market Data Request (V) ... 17
        4.3.3. Market Data - Snapshot / Full Refresh (W) ... 17
        4.3.4. Market Data - Incremental Refresh (X) ... 18
5. TCP Recovery (Historical Replay) service limitations ... 20

History of changes

Date	Version	Changes
20.01.2023	1.18.0	1. The FAST specification for the OTC monitor is separated into a separate document. 2. Three new message templates are added for broadcasting data from the OTC-monitor: • OtcMonitorIncrementalRefreshMessage (id="33") • OtcMonitorSnapshotMessage (id="34") • OtcMonitorSecurityDefinition (id="35") 3. New attributes were added to **Market Data - Snapshot / Full Refresh (W)** and **Market Data - Incremental Refresh (X)** messages: • **SettlCurrency** - Currency of monetary obligation • **CFICode** - CFI code of the security with which the trade was made • **TradeVolume** - Trade volume, rub. 4. New attributes were added to **Security Definition** message : • **CFICode** - CFI code of the security with which the trade was made • **InList** - Including of the security in the quotation list of the Exchange.

1. Introduction

1.1. Document purpose

This document overviews the FAST protocol specifications for OTC_monitor.

This document does not cover administrative and technical aspects of network connection. Also, this document does not cover security support aspect.

1.2. Fast Gate — Basic information

The **Fast Gate** system is used for distributing market data in the FAST-format via the UDP protocol in the multicast mode.

This approach combines the FIX protocol structure and message syntax with the FAST protocol dataflow optimization benefits. Also, it provides possibilities for fast and reliable data distribution to multiple clients of the UDP ptotocol.

The FAST (FIX Adapted for STreaming) protocol is a FIX based protocol developed by FIX Market Data Optimization Working Group in order to optimize financial data exchange performance and reduce latency in distributing large amounts of data. Fast Gate uses the protocol version 1.1: https://www.fixtrading.org/packages/fast-specification-version-1-1.

The FAST Gate for the OTC monitor sends the following data:

- instrument descriptions;
- anonymous data on OTC-trades reports.

1.2.1. Data streaming approach

Using of the data streaming approach allows to transmit data from sender to recepient without breaking it into separate messages. The new approach allows to combine several events into a single message which leads to higher data transfer speed and reduce latency time.

1.2.2. Incremental messages

Using of incremental messages allows to significantly reduce amount of transmitted data. Only the data changed due to the market events are transmitted; also, minimal number of commands are used for refreshing data: 'add new record', 'change record', 'delete record'.

1.2.3. FIX format

The **Fast Gate** system uses the FIX mesages format and syntax. Each message consists of header, message body and trailer. Fields are separated with the ASCII symbol — <SOH>.

For more information see sec. 4.

1.2.4. Encoding in the FAST format

The FAST (FIX Adapted for STreaming) protocol is the FIX based protocol developed by FIX Market Data Optimization Working Group in order to optimize financial data exchange performance and reduce latency in distributing large amounts of data.

The following features are used for data compression:

- implicit tagging;
- fields encoding options;
- usage of PMap;
- stop-bit encoding;
- usage of binary encoding method.

In most cases, the FAST format encoding rules are negotiated between counterparties by exchanging XML-templates.

For more information see sec. 3.2.

1.2.5. Data receicing via Multicast

For data distribution, the UDP protocol is used in order to distribute data to more than one client at once.

A single UDP packet may contain several FIX messages in the FAST format. Although, currently the system does not provide a possibility to send more than one FAST-coded message via a single UDP packet. In order to match the restriction, FAST messages are generated in a size not bigger then that of the MTU parameter, i.e. 1500 bytes, which is typical for Ethernet networks.

1.2.6. Data recovery

It is extremely important to clients to be able to recover data instantly in case of any data loss.

Fast Gate provides 2 methods of data recovery:

- recovering big amounts of data by sending snapshots (for example, for the clients connected to the system after the trading session start);
- recovering small amounts of data via TCP-connection (for example, in case of message loss during sending).

2. Scenarios of client interactions with Market Data Multicast

This section covers different scenarios of clients connection to the Market Data Multicast feeds. Also, this section covers loss data recovery procedures details.

2.1. Connect client

When client starts listening to MOEX Market Data Multicast FIX/FAST Platform, 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.).

2. Download the FAST template from ftp.

3. Receive the instruments list from **Instrument Replay** feed. Start listening to the **Instruments Incremental** feed.

4. Start listening to the **Incremental** feeds and queue received data.

5. Start listening to the **Snapshot** feeds. Receive and apply actual market data snapshot. In each Market Data - *Snapshot/Full Refresh (W)* tag *369-LastMsgSeqNumProcessed* is equal to tag *34-MsgSeqNum* of the last message Market Data - *Incremental Refresh (X)* of the appropriate stream included in the snapshot. The refresh number of each instrument within the tag *83-RptSeq* of the message Market Data - *Snapshot/Full Refresh (W)* is equal to number of incremental refresh in the tag *83-RptSeq* which corresponds to *MDEntry* of the last message Market Data - *Incremental Refresh (X)*, included into the snapshot. For each instrument, it is necessary to omit all messages with numbers through *369-LastMsgSeqNumProcessed* tag number and apply all that are left. The procedure can be both sequential or parallel. I.e., you can either receive snapshots for all instruments and then process the accumulated data or you can process data after receiving each snapshot.

6. Stop listening to the **Snapshot** feeds.

7. Continue receiving and normal processing incremental data.

2.2. 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:

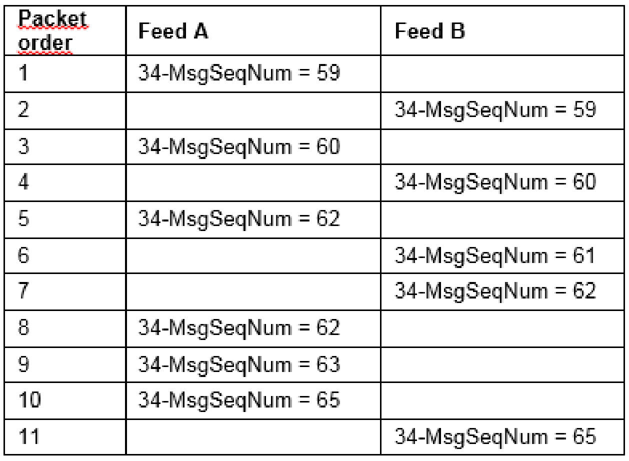
1. Listen feeds A and B.

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:



Packet order	Feed A	Feed B
1	34-MsgSeqNum = 59	
2		34-MsgSeqNum = 59
3	34-MsgSeqNum = 60	
4		34-MsgSeqNum = 60
5	34-MsgSeqNum = 62	
6		34-MsgSeqNum = 61
7		34-MsgSeqNum = 62
8	34-MsgSeqNum = 62	
9	34-MsgSeqNum = 63	
10	34-MsgSeqNum = 65	
11		34-MsgSeqNum = 65

Messages are 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 is missed.

3. System functionality

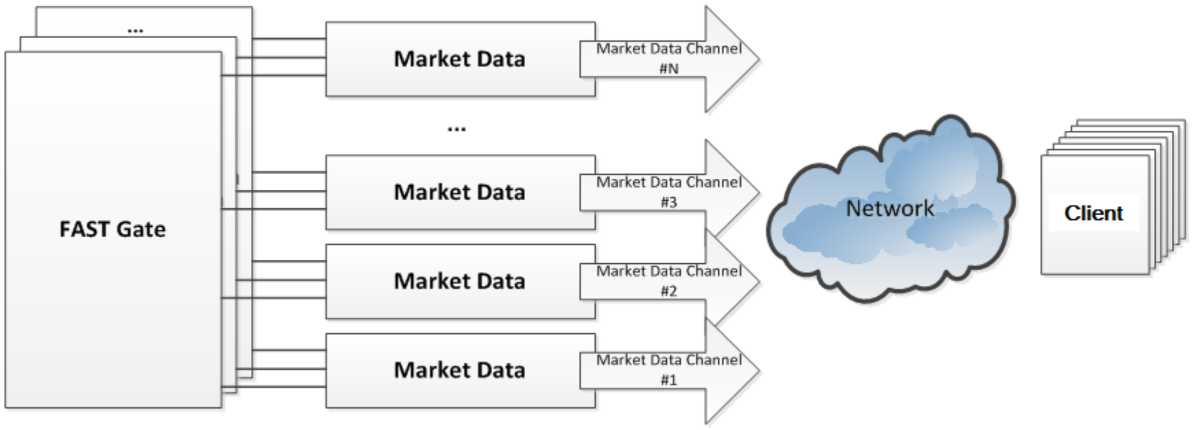
3.1. System architecture

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 one of UDP Channels.

Following feeds are used in the system:

1. Basic:
   - Market Data Incremental Refresh feeds;
   - Instrument Definition feed;
   - Data distribution feed for instrument status change and Trading System connection status.
2. Recovery feeds:
   - Market Recovery feed;
   - TCP Replay session.

Data are distributed via group of feeds, each of that contain data for financial instrument group. The instruments are grouped by the Trading System according to particular parameters. The dedicated Market Data Multicast instance is responsible for distribution in each Feed. A single Market Data Multicast instance is responsible for a single Feed data distribution.



Pic. 1. Market Data distribution feeds

Each feed is a) a bunch of several UDP-feeds with continuous data distribution; b) TCP-port which is used for requesting messages missed in the UDP-feed.

All streams are transmitted using the UDP multicast protocol and every stream is transmitted using a dedicated multicast address. The A and B streams transmit the same data in order to decrease the probability for missing UDP-packets.

Apart from transmitting data in UDP streams, Market Data multicast can accept incoming TCP connections for letting clients request missing data. Clients can request missing messages using one of the next UDP streams (data are available for a period of time specified in the configuration file (not earlier than from beginning of the day), number of messages to be sent at one is limited, number of requests per day is limited, too. All limits are specified in the system configuration file.

3.1.1. Main streams (UDP)

The OTC trades data are distributed in the OTC-TRADES main streams (incr) in the multicast mode with the UDP protocol.

The data are distributed as FIX-messages Market Data - Incremental Refresh (X) coded in the FAST format. Each message may contain refresh data for several financial instruments.

3.1.2. Recovery streams (UDP)

The Recovery (snap) streams in the multicast mode with the UDP protocol are used to periodically distribute the current snapshot of the corresponding data as FIX-messages Market Data - Snapshot/Full Refresh (W) coded in the FAST format. Each message contains data for a single instrument only.

It is not necessary for clients to be constantly connected to these streams. After receiving the missing data, it is recommended to disconnect from these streams.

3.1.3. Instrument Definitions streams (UDP)

The Instrument Replay (inst replay) streams are used to periodically distribute the trading session status and descriptions of financial instruments as TradingSessionStatus (h) and Security Definition (d) FIX messages coded in the FAST format. Each message contains description for a single financial instrument. The TradingSessionStatus (h) message is broadcast in FUT-INFO and OPT-INFO streams. In the Instrument Replay stream, the SequenceReset(4) message marks the start of broadcasting a new snapshot. A spashot consists of trading instruments descriptions in the form of SecurityDefinition (d) messages and a trading session status TradingSessionStatus (h). Upon receipt of a full snapshot, it should be considered that all instruments descriptions for a given trading session were received.

OTC-instruments descriptions are transmitted in the OTC-ISSUES stream.

3.1.4. Messages in streams

This section describes which messages are transmitted in each data stream.

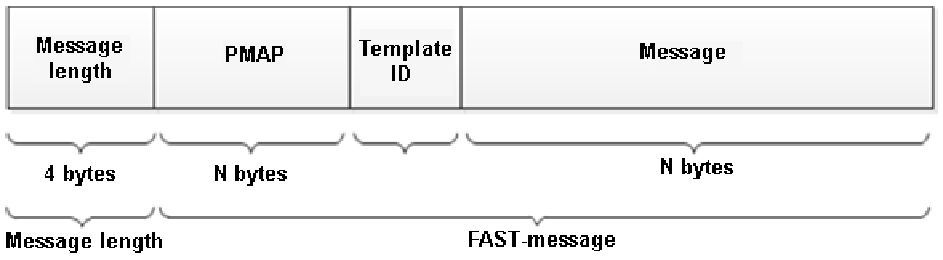
Stream name	Stream type	Message template name
OTC-ISSUES	Instrument Replay	Heartbeat (id="6") SequenceReset (id="7") OtcMonitorSecurityDefinition (id="35")
OTC-TRADES	Incremental	Heartbeat (id="6") SequenceReset (id="7") OtcMonitorIncrementalRefreshMessage (id="33")
OTC-TRADES	Snapshot	Heartbeat (id="6") SequenceReset (id="7") OtcMonitorSnapshotMessage (id="34")
OTC-TRADES	Historical Replay	**From client to gateway:** Logon (FIX MessageType="A") Logout (FIX MessageType="5") Market Data Request (FIX MessageType="V") **From gateway to client:** Heartbeat (id="6") OtcMonitorIncrementalRefreshMessage (id="33") Logon (id="1000") Logout (id="1001")

3.1.5. Sessions for requesting missing messages (TCP)

This service allows to request the resend of missing messages within a specified range of numbers.

The request contains a range of message (numbers) to resend. The request is sent as the Market Data Request (V) FIX-message using the client-initiated TCP-connection. The respond messages are sent to the client as FIX-messages coded in the FAST format using the same TCP-connection. Upon completion of sending, Market Data Multicast closes this TCP-connection. Please note, that maximum number of messages to resend is limited.

The first 4 bytes of each message transmitted in a TCP stream contain its length.



Pic. 2. Message structure in TCP stream

When all FAST messages have been sent out, the gateway sends the message Logout to the FAST client, expecting the message Logout from the client in respond. Finishing FIX session also causes TCP session to close.

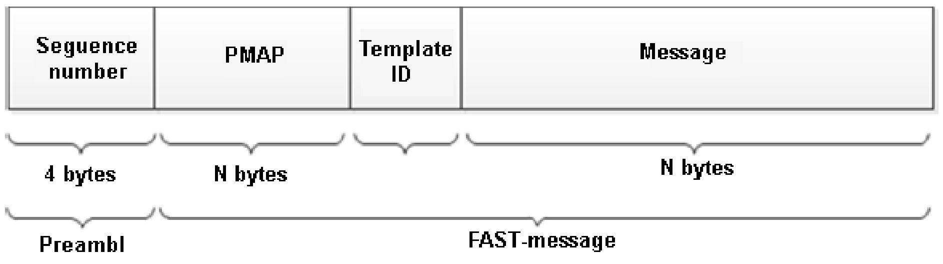
Please also note, that this service should be used only when all other methods are unavailable. This service does not provide high performance and is not available for streams containing aggregated Book-order data.

3.2. FAST format — details

All messages sent by MOEX Market Data Multicast are in the FIX-format coded in the FAST (FIX Adapted for STreaming) protocol. The FAST protocol was developed by FIX Market Data Optimization Working Group in order to optimize financial data flow via distributing bigger amounts of data with less latency.

A specific feature of data distribution via the MOEX Market Data Multicast streams is, that there is a 4-bytes preamble added before every FAST-message. The preamble contains the 34-th tag (SeqNum) value. The 34-th tag is located right after the preamble.

It allows to receive the message sequence number (both when processing messages from the A and B streams and in case of missing messages) without decoding the FAST-message itself; this leads to time saving during processing of streams.



Pic. 3. Message structure

3.2.1. Stop bit encoding

Encoding stop bit is a constitutive procedure of FAST. The coding allows to exclude redundancy on the data field link layer using the stop bit instead of the standard byte separator. In FAST, stop bit is used instead of the standard FIX-separator (<SOH> byte); therefore, 7 bits of every byte are used for data transmission while the 8th bit indicates the field end.

3.2.2. Implicit tagging

According to the FIX protocol standards, every message is as: **Tag = Value <SOH>**, where:

Tag — number of the field, which is now transmitted;

Value — actual value in this field;

<SOH> – ASCII symbol, used as a separator.

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 allows to avoid this redundancy by using a template which describes the whole message structure. This method is called 'implicit tagging', as FIX tags become implicit parts of the transmitted data. FAST-template exchanges the 'Tag=Value' syntax with 'implicit tagging' according to the following rules:

- tags numbers are not transmitted in message but specified in the template;
- sequence of fields in the message is alike to one of the tags in the template;
- the template specifies a structured bunch of fields with their operators.

3.2.3. Fields encoding options

FAST operates as a state machine, which must 'know' all values to store in memory each moment of time. FAST compares the current field value with the previous one and decides how to act:

- use the constant specified in the template as a new value;
- use the default value (in case of absence of a new field value).

3.2.4. FAST-template

A FAST template corresponds to the FIX message type and uniquely identifies order of fields in each message.

The template also includes syntax indicating the type of field and transfer decoding to apply. Each FAST message contains template ID which is used for decoding. The message templates are available at: ftp://ftp.moex.com/pub/FAST/Spectra/ .

3.2.5. Decoding overview

Below is the order of decoding procedure:

1. Transport. A client receives an encoded FAST message.
2. Packet decoding:
   - identification of a template;
   - withdrawal of binary encoded bits;
   - determining correspondences between the received bits and template fields.
3. Fields decoding: using operators to determine value according to the template.
4. Generation of FIX-message.
5. Processing the FIX-message.

3.2.6. Message fragmentation

In order to prevent UDP packets from exceeding MTU size of 1500 bytes (typical for Ethernet networks), messages are fragmented into several parts.

If the message Market Data - Snapshot / Full Refresh (W) does not contain the tag 893-LastFragment, it means that snapshot was transmitted as a single message. All fragmented messages except the last one contain the tag 893-LastFragment = 0. The last fragmented messages contains the tag 893-LastFragment = 1. Therefore, receiving a message with the tag 893-LastFragment = 1 indicates that snapshot has been completely transmitted.

If the message Market Data - Incremental Refresh (X) does not contain the tag 893-LastFragment, it means that messages have not been fragmented, and the book is consistent after processing the message. All fragmented messages except the last one contain the tag 893-LastFragment = 0. The last fragmented messages contains the tag 893-LastFragment = 1. Therefore, receiving a message with the tag 893-LastFragment = 1 indicates that the book is consistent.

3.2.7. Data types

A field within a FAST template will have one of the standard Data Types indicating the required decoding action: ASCII string, Unicode string, Signed Integer, Unsigned Integer and Decimal. Decimal exponent and mantissa will be encoded as a single, composite field.

FAST does not natively support timestamps. FAST gate will convert the timestamp to an integer value depending on the field type. The decoding application should convert the integer to the FIX UTC format after decoding. Time is always displayed in UTC.

Samples of timestamps encoding:

FIX Type	FIX Pattern	Sample FIX value	Sample FAST value	FAST field type
UTCTimeOnly	HH:MM:SS.sssssssss (nanoseconds)	18:44:24.123456789	184424123456789	uInt64
	HH:MM:SS.sssssssss (nanoseconds)	07:12:13.012345678	71213012345678	uInt64
UTCDateOnly	YYYYMMDD	20080812	20080812	uInt32
UTCTimestamp	YYYYMMDD-HH:MM:SS.sss	20080812-18:23:54.213	20080812182354123	uInt64

3.3. Missing data 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 h as 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 recommended only for emergency recovering of 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.

3.3.1. Recovery missing data using Recovery streams (UDP)

This recovery method is preferable to use for 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 in the corresponding feed. 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 last Market Data - Incremental Refresh (X) message. Thus, tag MsgSeqNum(34) shows the gap at the messages level, tag RptSeq(83) shows 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 actual.

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, market data becomes actual.

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

If a message does not contain the tag 893-LastFragment, it means that snapshot was transmitted as a single message. Otherwise, the last fragmented messages contains the tag 893-LastFragment = 1. Therefore, receiving a message with the tag 893-LastFragment = 1 indicates that snapshot has been completely transmitted.

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

Consequence of recovery is equal to that described in sec. 2.1 (steps 4 - 7).

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

3.3.2. Recovering missing data using TCP-connection

If there any market data missing in incremental streams Indexes, Trades and ORDERS-LOG (anonymous orders and trades), 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 only be used if other options are unavailable or for small - scale data recovery. Number of messages which can be requested by client during TCP connection is limited to 1000.

To request missing data, you should do the following:

1. Establish TCP connection with MOEX Market Data Multicast.
2. Send FIX message Logon(A) with sequence numder 1 to server. After successful authorization server sends the FAST - encoded Logon(A) message.
3. Send Market Data Request (V) message with:
   a. Range of sequence numbers - ApplBegSeqNum(1182) and ApplEndSeqNum (1183) tags.

If request is correct, server sends FAST messages according to requested sequence numbers.

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

After server responses, the connection is closed.

Server will process only first user request, second and others will be ignored. If the server does not receive Market Data Request within an established timeout interval after logon, the connection is closed.

Recovery channel has 1 second incoming request timeout.

3.4. Message sequence reset

Every 24 hours, the Fast Gate is being cleaned up from the last day trading session messages, and its message sequences are being reset. When the message sequences have been reset, a message 'Sequence Reset' with a new value in the field 'NewSeqNo' will be transmitted in the (incr) streams. Upon receiving the message 'Sequence Reset', the client is to set the message number value to that transmitted in the message 'NewSeqNo', and reset 'RptSeq' numbers.

Below is the break time schedule for Fast Gate. In the end of each break time, message sequence numbers will be reset:

- OTC Monitor - 0:00 AM (Moscow Time) till 00:02 AM (Moscow Time);
- Test environment for OTC Monitor - 0:00 AM (Moscow Time) till 00:02 AM (Moscow Time);

For all the main (incr) streams the message sequence number will be set to 1, and 'RptSeq' number will be set to 1. Also, the message pair 'Sequence Reset' will be transmitted:

- Sequence Reset: MsgSeqNum=N NewSeqNo[36]=1

- Sequence Reset: MsgSeqNum=N NewSeqNo[36]=N

After those messages have been transmitted, the FAST-messages containing trading data of the last evening trading session, with numbers from 1 till N-1 inclusive, will become available through the TCP Recovery service. The initial 'RptSeq' number value can be obtained with one of the following methods:

- request and process messages with numbers from 1 till N-1 available through the TCP Recovery service;
- connect to Recovery (UDP) stream, in accordance with information provided in section 3.3.1 Recovery missing data using Recovery streams (UDP).

3.5. Message templates

There are three certain message templates used for OTC-monitor data:

- **OtcMonitorIncrementalRefreshMessage (id="33")** - see sec. 3.5.1
- **OtcMonitorSnapshotMessage (id="34")** - see sec. 3.5.2
- **OtcMonitorSecurityDefinition (id="35")** - see sec. 3.5.3

3.5.1. OtcMonitorIncrementalRefreshMessage

This template is used for data refresh purpose. Also, it is used by the service TCP Recovery.

```
<template name="OtcMonitorIncrementalRefreshMessage" id="33">
        <string name="ApplVerID" id="1128">
            <constant value="9"/>
        </string>
        <string name="MessageType" id="35">
            <constant value="X"/>
        </string>
        <string name="SenderCompID" id="49">
            <constant value="MOEX"/>
        </string>
        <uInt32 name="MsgSeqNum" id="34"/>
        <uInt64 name="SendingTime" id="52"/>
        <uInt32 name="LastFragment" id="893" presence="optional"/>
        <sequence name="MDEntries">
            <length name="NoMDEntries" id="268"/>
            <uInt32 name="MDUpdateAction" id="279"/>
            <string name="MDEntryType" id="269"/>
            <string name="Symbol" id="55"/>
            <string name="SecurityGroup" id="1151"/>
            <uInt32 name="RptSeq" id="83"/>
            <int64 name="MDEntryID" id="278"/>
            <string name="MDEntryPx" id="270"/>
            <int64 name="MDEntrySize" id="271"/>
            <uInt32 name="MDEntryDate" id="272" presence="optional"/>
            <uInt64 name="MDEntryTime" id="273"/>
            <string name="Currency" id="15"/>
            <uInt64 name="Revision" id="20018" presence="optional"/>
            <string name="OrderSide" id="10504"/>
            <string name="SettlCurrency" id="120"/>
            <string name="CFICode" id="461"/>
            <string name="TradeVolume" id="1020"/>
        </sequence>
    </template>
```

3.5.2. OtcMonitorSnapshotMessage

This template is used for distributing snapshots.

```
<template name="OtcMonitorSnapshotMessage" id="34">
        <string name="ApplVerID" id="1128">
            <constant value="9"/>
        </string>
        <string name="MessageType" id="35">
            <constant value="W"/>
        </string>
        <string name="SenderCompID" id="49">
            <constant value="MOEX"/>
        </string>
        <uInt32 name="MsgSeqNum" id="34"/>
        <uInt64 name="SendingTime" id="52"/>
        <uInt32 name="LastFragment" id="893" presence="optional"/>
        <uInt32 name="RptSeq" id="83"/>
        <uInt32 name="TotNumReports" id="911"/>
        <uInt32 name="LastMsgSeqNumProcessed" id="369"/>
        <string name="Symbol" id="55"/>
        <string name="SecurityGroup" id="1151"/>
        <sequence name="MDEntries">
            <length name="NoMDEntries" id="268"/>
            <uInt32 name="MDUpdateAction" id="279"/>
            <string name="MDEntryType" id="269"/>
            <int64 name="MDEntryID" id="278"/>
            <string name="MDEntryPx" id="270"/>
            <uInt32 name="MDEntryDate" id="272" presence="optional"/>
            <uInt64 name="MDEntryTime" id="273"/>
            <int64 name="MDEntrySize" id="271"/>
            <string name="Currency" id="15"/>
            <string name="OrderSide" id="10504"/>
            <string name="SettlCurrency" id="120"/>
            <string name="CFICode" id="461"/>
            <string name="TradeVolume" id="1020"/>
        </sequence>
    </template>
```

3.5.3. OtcMonitorSecurityDefinition

This template is used for distributing information about the instruments.

```
<template name="OtcMonitorSecurityDefinition" id="35">
    <string name="ApplVerID" id="1128">
        <constant value="9"/>
    </string>
    <string name="MessageType" id="35">
        <constant value="d"/>
    </string>
    <string name="SenderCompID" id="49">
        <constant value="MOEX"/>
    </string>
    <uInt32 name="MsgSeqNum" id="34"/>
    <uInt64 name="SendingTime" id="52"/>
    <!-- Total count of SecurityDefinition messages -->
    <uInt32 name="TotNumReports" id="911"/>
    <string name="Symbol" id="55"/>
    <string name="SecurityDesc" id="107" presence="optional" charset="unicode"/>
    <!-- Unique among all instruments; primary key -->
    <uInt64 name="SecurityID" id="48"/>
    <uInt32 name="SecurityIDSource" id="22">
        <constant value="8"/>
    </uInt32>
    <string name="SecurityAltID" id="455"/>
    <string name="SecurityAltIDSource" id="456"/>
    <string name="CFICode" id="461"/>
    <string name="MarketID" id="1301">
        <constant value="MOEX"/>
    </string>
    <string name="MarketSegmentID" id="1300"/>
    <sequence name="MDFeedTypes">
        <length name="NoMDFeedTypes" id="1141"/>
        <string name="MDFeedType" id="1022"/>
        <uInt32 name="MarketDepth" id="264" presence="optional"/>
        <uInt32 name="MDBookType" id="1021" presence="optional"/>
    </sequence>
    <sequence name="InstrumentAttributes">
        <length name="NoInstrAttrib" id="870"/>
        <int32 name="InstrAttribType" id="871"/>
        <string name="InstrAttribValue" id="872" charset="unicode"/>
    </sequence>
    <decimal name="UnderlyingQty" id="879" presence="optional"/>
    <string name="UnderlyingCurrency" id="318" presence="optional"/>
    <string name="InList" id="20052"/>
</template>
```

4. FIX protocol message specifications

The protocol message specifications description below is based on the standard FIX protocol specification v. 5.0 SP2 (https://www.fixtrading.org/standards/fix-5-0-sp-2). It is recommended for users to read some general information about the protocol before commencing with this specification.

Each field is described below:

- **Tag** – the unique field ID, used for generating a FIX message.
- **Field** – the field name, not used for generating FIX messages and described for your reference only.
- **Mandatory** – a field attribute: specifies whether the field in message is mandatory or optional.
  - Y - mandatory field;
  - N - optional field;
  - C - mandatory, if meets the condition (see 'Details').
- **Details** – detailed description of the field.
- **Allowable values** - additional limitations.

The "*****" symbol - flag of difference from the standard FIX protocol.

4.1. Field groups

Many messages contain the same fields. For example, the 'Standard Message Header' group fields contain some administrative information and are mandatory for every message.

4.1.1. Standard Message Header

The standard header, mandatory for every message.

Tag	Field	Mandatory	Details	Available values
34	MsgSeqNum	Y	Message sequence number	
35	MsgType	Y	Message type	
49	SenderCompID	Y	Message sender ID	• 'MOEX' - Moscow Exchange • 'ETSC' - Kazakhstan Exchange (ETS)
52	SendingTime	Y	Message sending time	
1128	ApplVerID	Y	FIX protocol version ID	"9" (FIX50SP2)

4.2. Session layer messages

4.2.1. Logon (A)

A FIX message which is used to initiate a session establishment to the service TCP Recovery.

Tag	Field	Mandatory	Details
8	BeginString	Y	Allowable values: 'FIX.4.4' and 'FIXT.1.1'.
9	BodyLength	Y	Message length.
35	MsgType	Y	'A'
553	Username	N	Any string
554	Password	N	Any string
10	CheckSum	Y	Checksum.

A FAST message which is used to confirm a session establishment to the service TCP Recovery.

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type 'A'.

4.2.2. Logout (5)

A FIX message which is used to initiate a session closure with the service TCP Recovery.

Tag	Field	Mandatory	Details
8	BeginString	Y	Allowable values: 'FIX.4.4' and 'FIXT.1.1'.
9	BodyLength	Y	Message length.
35	MsgType	Y	'5'
10	CheckSum	Y	Checksum.

A FAST message which is used to confirm a session closure with the service TCP Recovery.

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type '5'.
58	Text	N	Reason for ending the session. In case of refusal to process the request, a description of the error is transmitted in the field. If the request is successfully processed, 'nullValue' is transmitted.

4.2.3. Heartbeat (0)

The message **HeartBeat** is sent by FastGate when there were no messages sent in the stream within a 30 seconds time interval.

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type '0'.

4.2.4. Sequence Reset (4)

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type '4'.
36	NewSeqNo	Y	New sequence number.

4.3. Business logic layer messages

This section describes messages of OTC-monitor streams.

The following FIX messages are supported:

- **Security Definition** – Information on instrument.
- **Market Data Request** - Missed data request.
- **Market Data - Snapshot / Full Refresh** – Data snapshot.
- **Market Data - Incremental Refresh** – Data refresh.

4.3.1. Security Definition (d)

Information on instrument.

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type 'd'
1301	MarketId*	Y	Exchange MIC: 'MOEX' - Moscow Exchange
1300	MarketSegmentId*	Y	'Q' - OTC-trades
48	SecurityId	Y	Instrument unique ID
22	SecurityIdSource	Y	'8' - Exchange Symbol
55	Symbol	Y	Security code
107	SecurityDesc	N	Name of the issuer of the security
455	SecurityAltID*	Y	Instrument symbol code
456	SecurityAltIDSource*	Y	Class for SecurityAltID (455): • '4' - ISIN number
461	CFICode	Y	Financial instrument class according to ISO-10962.
870	NoInstrAttrib	Y	=6

Tag	Field	Mandatory	Details
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=204 State registration number
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=200 Total number of securities by issuer, in units.
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=205 Full name of the issuer of the security or other person liable under the security (full name of the management company)
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=206 Name of the mutual investment fund
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=207 Type, category of security
=> 871 => 872	InstrAttribType InstrAttribValue	Y Y	=208 Kind of security
879	UnderlyingQty	N	Security nominal value
318	UnderlyingCurrency	N	Code of currency of the security nominal value
20052	InList	Y	Allowable values: • **Y** - the security is included in the quotation list of the Exchange; • **N** - the security is not included in the quotation list of the Exchange, or is included in the quotation list of the Exchange, but is a security of a foreign issuer, the listing of which was carried out without concluding an agreement with the issuer

***** - differs from the standard FIX protocol.

4.3.2. Market Data Request (V)

A FIX message which is used to request missing data in the session to the service TCP Recovery.

Tag	Field	Mandatory	Details
8	BeginString	Y	Allowable values: • FIX.4.4 • FIXT.1.1
9	BodyLength	Y	Message length
35	MsgType	Y	"V"
262	MDReqId	Y	Request ID
1182	ApplBegSeqNum	N	Sequence number of the first requested message.
1183	ApplEndSeqNum	N	Sequence number of the last requested message. If a single message is requested, then ApplBegSeqNum(1182)=ApplEndSeqNum(1183). If all messages are requested (no more than total messages sent) after a particular message number, then ApplEndSeqNum(1183)=0(infinity).
10	CheckSum	Y	Checksum

4.3.3. Market Data - Snapshot / Full Refresh (W)

Data snapshot.

Tag	Field	Mandatory	Details
<Standard Message Header>		Y	Message type 'W'
893	LastFragment	N	Indicates the last message in the message group for the instrument. Allowable values: • **0** – not the last message • **1** – the last message

Tag	Field	Mandatory	Details
			The field is not mandatory. If a message does not contain this field, it means that the packet with message has not been fragmented
83	RptSeq	Y	The 'RptSeq' number of the last incremental update included in the current market data snapshot for instrument
911	TotNumReports	Y	The number of messages in the snapshot, which have 'LastFragment '= 1
369	LastMsgSeqNumProcessed	Y	The 'MsgSeqNum' of the last message sent into incremental feed at the time of the current snapshot generation
55	Symbol	Y	Security code
1151	SecurityGroup	Y	= 'OTC'
268	NoMDEntries	Y	Number of 'MDEntry' records in the current message
=>279	MDUpdateAction	Y	Incremental refresh type: • '0' - New • '1' - Change • '2' - Delete
=>269	MDEntryType	Y	Record type: '2' - Trade
=>278	MDEntryID	Y	Registration identifier of the trade assigned by the OTC-monitor system
=>270	MDEntryPx	Y	The price of one security
=>15	Currency	Y	Currency code. Always = "RUB"
=>271	MDEntrySize	Y	Quantity of securities
=>272	MDEntryDate	N	Trade date
=>273	MDEntryTime	Y	Trade time
=>10504	OrderSide	Y	Trade direction (Buy 'B' / Sell 'S') Allowable values: • ‘1’ – buy order (Buy); • ‘2’ – sell order (Sell).
=>120	SettlCurrency	Y	Currency of monetary obligation
=>461	CFICode	Y	CFI code of the security with which the trade was made
=>1020	TradeVolume	Y	Trade volume, rub.

***** - differs from the standard FIX protocol.

4.3.4. Market Data - Incremental Refresh (X)

Data refresh.

Tag	Поле	Наличие	Описание
<Standard Message Header>		Y	Message type 'X'
893	LastFragment	N	Indicates the last message in the message group for the instrument. Allowable values: • **0** – not the last message • **1** – the last message The field is not mandatory. If a message does not contain this field, it means that the packet with message has not been fragmented
268	NoMDEntries	Y	Number of 'MDEntry' records in the current message
=>279	MDUpdateAction	Y	Incremental refresh type: • '0' - New • '1' - Change • '2' - Delete
=>269	MDEntryType	Y	Record type: '2' - Trade

Tag	Поле	Наличие	Описание
=>55	Symbol	Y	Security code
=>1151	SecurityGroup	Y	= 'OTC'
=>83	RptSeq	Y	Incremental refresh sequence number
=>278	MDEntryID	Y	Registration ID of the trade that was deleted or changed. When the report about trade is changed, it retains the same Registration ID that was assigned by the OTC Monitor system when adding the trade report.
=>270	MDEntryPx	Y	The price of one security
=>271	MDEntrySize	Y	Quantity of securities
=>272	MDEntryDate	N	Trade date
=>273	MDEntryTime	Y	Trade time
=>15	Currency	Y	Currency code. Always = 'RUB'
=>20018	Revision	N	Service field of the replication subsystem
=>10504	OrderSide	Y	Trade direction (Buy 'B' / Sell 'S'). Allowable values: • ‘1’ – buy order (Buy); • ‘2’ – sell order (Sell).
=>120	SettlCurrency	Y	Currency of monetary obligation
=>461	CFICode	Y	CFI code of the security with which the trade was made
=>1020	TradeVolume	Y	Trade volume, rub.

***** - differs from the standard FIX protocol.

5. TCP Recovery (Historical Replay) service limitations

The following limitations are applied to the TCP Recovery service for streams OTC-TRADES, ORDERS-LOG in order to lower the load:

Parameter	Value	Details
Maximum active connections, per market, per instance, per IP address	2	You can establish no more than indicated number active TCP connection from single IP address. An attempt to make more connections will be rejected
Maximum connections count, per market, per instance, per day, per IP address	1000	You can make no more than indicated number of tcp connections per IP address per day. Extra connection attempts will be rejected
Maximum number of messages to request	1000	TCP replay request is rejected if a number of requested messages is greater than indicated value
Marketdata request timeout, seconds	1	Connection is terminated with logout message if marketdata request is not received within indicated number of seconds since logon message. TCP session is terminated if no confirming logout is received after server-side logout.