Moscow Exchange
FIX-protocol for OTCT board

User guide

Moscow Exchange
Version 1.0
Apr 16th 2019

Contetnts

1.Overview

Document purpose

The document below describes the FIX protocol provided by the Moscow Exchange for connection to OTCT board of FX Market. The description is based on the standard FIX protocol (Financial Information Exchange, http://www.fixtrading.org, version 4.4) specification. It is assumed that users have basic knowledge about FIX standard. The specification does not contain neither technical nor administrative details on network connection or security protection methods.

OTCT FIX Server supports only messages, component blocks and fields that are described in this document.

Note that all fields which are required or conditionally required by FIX 4.4 standard but absent in MOEX Interface specification are optional and will be ignored by MOEX. All field values, which are valid according to FIX 4.4 standard but are not described in this document, will be considered as invalid and messages with such values will be rejected.

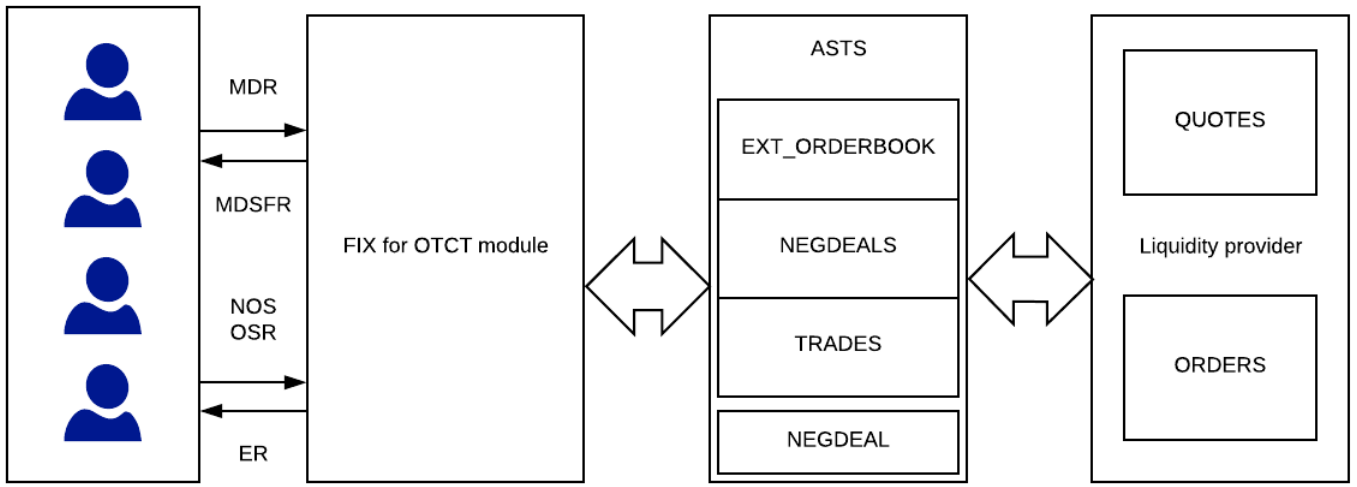
Each message or component block is represented as table, where each row is a message field or component block. The following characteristics are described for each field:

- Tag – unique field identifier.
- Name – field name.
- Required – shows whether the field is required or not in appropriate message or component block.
  - ‘Y’ – tag is required (mandatory);
  - ‘N’ – tag is not required (optional);
  - ‘C’ – tag is conditionally required.
  - ‘Y*’ – tag is required by MOEX, but not required by the standard FIX 4.4 protocol;
  - ‘N*’ – tag is not required by MOEX but required by the standard FIX 4.4 protocol;
  - ‘C*’ – tag is conditionally required by MOEX, but not required by the standard FIX 4.4 protocol.
- Type – field type.
- Valid values – list of valid tag values;
- Comments – comments, additional information for the tag.

Service description

Base service functionality:

- User authorization \ authentication in Trading System
- Order sending and validation
- Order status request
- Market data part: Orderbook
- Subscription management



MDR - Market Data Request
MDSFR - Market Data Snapshot Full Refresh
NOS - New Order Single
ER - Execution Report
OSR - Order Status Request

2 FIX components block

Standard Message Header

A standard header precedes each administrative or application message. The header identifies the message type, length, destination, sequence number, origination point and time.

Tag	Field name	Required	Type	Valid values	Comments
8	BeginString	Y	String (7)	'FIX.4.4'	Identifies beginning of new message and protocol version. Always unencrypted, must be first field in message.
9	BodyLength	Y	Length		Message length, in bytes, forward to the CheckSum field. Always unencrypted, must be second field in message.
35	MsgType	Y	String (10)		Defines message type. Always unencrypted, must be third field in message.
49	SenderCompID	Y	String (12)		Assigned value used to identify firm sending message. Always unencrypted. If this message is sent **to** MOEX, then it should contain USERID assigned to a trader by MOEX. If this message is sent from MOEX, then it will contain the MOEX server identifier. This parameter is given by MOEX
56	TargetCompID	Y	String		Assigned value used to identify receiving firm. Always unencrypted. If this message is sent from MOEX, then it will contain USERID assigned to a trader by MOEX. If this message is 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.
43	PossDupFlag	N	Boolean	'Y' (Possible duplicate) 'N' (Original transmission)	Indicates possible retransmission of message with this sequence number. Required for retransmitted messages.
97	PossResend	N	Boolean	‘Y’ (Possible resend) ‘N’ (Original transmission)	Indicates that message may contain information that has been sent under another sequence number. Milliseconds part contains exactly 3 symbols.
52	SendingTime	Y	UTCTimestamp		Time of message transmission (expressed in UTC). Field format is YYYYMMDD-HH:MM:SS.sss.

122	OrigSendingTime	N	UTCTimestamp		Original time of message transmission when transmitting messages as the result of resend request (expressed in UTC). Field format is YYYYMMDD-HH:MM:SS.sss. Required for message resend as a result of a resend request.

Standard Message trailer

A standard trailer terminates each message, administrative or application. The trailer is used to segregate messages and contains the three-digit character representation of the Checksum value.

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

Parties

The <Parties> component block is used to define parties of order or trade. For each party the following fields should be defined: PartyID, PartyIDSource, PartyRole.

- PartyID (448) = <firm id>, PartyIDSource (447) = ‘D’, PartyRole (452) = ‘1’ – specifies firm;
- PartyID (448) = <client code>, PartyIDSource (447) = ‘D’, PartyRole (452) = ‘3’ – specifies client;

Other PartyRole values which you could meet in Execution Report (8) from the server side

- PartyID (448) = <user id>, PartyIDSource (447) = ‘D’, PartyRole (452) = ‘12’ – specifies user (trader, broker);
- PartyID (448) = <counterparty firm id>, PartyIDSource (447) = ‘D’, PartyRole (452) = ‘17’ - specifies counterparty firm.

Tag	Field name	Required	Type	Valid values	Comments
453	NoPartyID	Y*	NumInGroup		Number of repeating PartyID group entries.
448	=> PartyID	C	String (12)		Party identifier/code. Required if PartyIDSource is specified. Required if NoPartyIDs > 0. User, firm, client can be defined as party of order/trader. Note: this field must contain MOEX CLIENTCODE value that is assigned by broker to a client. For own broker's accounts this value is ignored and not returned in execution reports.
447	=> PartyIDSource	C	char	'D' (Proprietary/Custom code)	Identifies class or source of the PartyID (448) value. Required if PartyID (448) is specified. Note: applicable values depend upon PartyRole (452) specified. Only constant value 'D' is used.
452	=> PartyRole	C	int	'1' (Executing Firm); '3' (Client ID); '12' (Executing Trader); '17' (Contra Firm).	Identifies the type or role of the PartyID (448) specified. '1' is used for firm; '3' is used for client in New Order – Single (D) message; '12' is used for user (trader, broker). '17' is used for counterparty.

Instrument

The <Instrument> component block describes security which is traded the exchange.

Tag	Field name	Required	Type	Valid values	Comments
55	Symbol	Y	String (12)		Ticker symbol. The MOEX internal instrument identifier, SecCode
460	Product	N	int	'4' (Currency)	Indicates the type of product the security is associated with.

3.FIX session-level messages

The session level messages are used to establish, close (terminate), support FIX session, manage session status and opportunity to reestablish it after failure.

Logon (A)

The logon message is used to initiate FIX session and to confirm the establishing of it. The logon message must be the first message sent by the application requesting to initiate a FIX session. The Logon reply normally can take up to 3 seconds to receive. Please allow 3 seconds waiting time for reply

Note: SenderCompID cannot be used for more than single connection to any MOEX service.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'A'
98	EncryptMethod	Y	int	'0' (None)	Method of encryption. Always unencrypted. Note: Encryption is not supported by MOEX.
108	HeartBtInt	Y	int		Heartbeat interval (seconds). Values must fit in limits between 1 and 60. If HeartBtInt field value is out of range, then Logout (35=5) message is sent with text description of error.
141	ResetSeqNumFlag	N	Boolean	'Y' (Yes) 'N' (No)	Indicates if the both sides of the FIX session should reset sequence numbers. Default value is 'Y'
554	Password	Y*	String(8)		User password. The maximum length is 8 characters. Required field at MOEX
1409	SessionStatus	N	Char	'0' (Session active) '5' (Wrong password or user ID) '7' (Logons are not allowed \ UserID already in use)	Status of the request to change the password. Required if the server generates the message.
6936	LanguageID	N	Char	'R' (Russian) 'E' (English)	Language of Trading System messages.

<Standard Message Trailer>	Y			

Logout (5)

The logout message initiates or confirms the termination of a FIX session.

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

Heartbeat (0)

The Heartbeat monitors the status of the communication link and identifies when the last of a string of messages was not received.

Heartbeats issued as the result of Test Request must contain the TestReqID transmitted in the Test Request message. This is useful to verify that the Heartbeat is the result of the Test Request and not as the result of a regular timeout.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '0'
112		N	String		Identifier included in Test Request (1) message to be returned in resulting Heartbeat (0).
<Standard Message Trailer>		Y			

Test Request (1)

The test request message forces a heartbeat from the opposing application. The test request message checks sequence numbers or verifies communication line status. The opposite application responds to the Test Request with a Heartbeat containing the TestReqID.

The TestReqID verifies that the opposite application is generating the heartbeat as the result of Test Request and not a normal timeout. The opposite application includes the TestReqID in the resulting Heartbeat. Any string can be used as the TestReqID (one suggestion is to use a timestamp string).

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '1'
112		Y	String		Identifier included in Test Request (1) message to be returned in resulting Heartbeat (0).
<Standard Message Trailer>		Y			

Resend Request (2)

Receiving application sends the resend request to initiate the retransmission of messages. This function is utilized if a sequence number gap is detected, if the receiving application lost a message, or as a function of the initialization process.
The resend request can be used to request a single message, a range of messages or all messages subsequent to a particular message.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '2'
7		Y	SeqNum		Message sequence number of first message in range to be resent.
16		Y	SeqNum		Message sequence number of last message in range to be resent. If request is for a single message BeginSeqNo (7) = EndSeqNo (16). If request is for all messages subsequent to a

				particular message, EndSeqNo (16) = '0' (representing infinity).
<Standard Message Trailer>	Y			

Sequence Reset (4)

The Sequence Reset message has two modes: Gap Fill mode and Reset mode.
Gap Fill mode is used in response to a Resend Request when one or more messages must be skipped over for the following reasons:

- During normal resend processing, the sending application may choose not to send a message (e.g. an aged order).
- During normal resend processing, a number of administrative messages are skipped and not resent (such as Heart Beats, Test Requests).

Gap Fill mode is indicated by GapFillFlag (tag 123) field = "Y". If the GapFillFlag field is present (and equal to "Y"), the MsgSeqNum should conform to standard message sequencing rules (i.e. the MsgSeqNum of the Sequence Reset GapFill mode message should represent the beginning MsgSeqNum in the GapFill range because the remote side is expecting that next message sequence number).

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '4'
123		N	Boolean	Y' (Gap Fill message, MsgSeqNum field valid) 'N' (Sequence Reset, ignore MsgSeqNum)	Indicates that the Sequence Reset (4) message is replacing administrative or application messages which will not be resent.
36		Y	SeqNum		New sequence number.
<Standard Message Trailer>		Y			

Reject (3)

The reject message should be issued when a message is received but cannot be properly processed due to a session-level rule violation. An example of when a reject may be appropriate would be the receipt of a message with invalid basic data (e.g. MsgType=&) which successfully passes CheckSum and BodyLength checks. Generation and receipt of a Reject message indicates a serious error that may be the result of faulty logic in either the sending or receiving application.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '3'
45		Y	SeqNum		MsgSeqNum (34) of rejected message.
371		N	int		The tag number of the FIX field being referenced.
372		N	String(10)		The MsgType (35) of the FIX message being referenced.
373		N	int	'0' (Invalid tag number) '1' (Required tag missing) '2' (Tag not defined for this message type) '3' (Undefined tag) '4' (Tag specified without a value) '5' (Value is incorrect (out of range) for this tag) '6' (Incorrect data format for value) '7' (Decryption problem) '8' (Signature problem) '9' (CompID problem) '10' (SendingTime accuracy problem) '11' (Invalid MsgType) '12' (XML validation error) '13' (Tag appears more than once) '14' (Tag specified out of required order) '15' (Repeating group fields out of order) '16' (Incorrect NumInGroup count for repeating group) '17' (Non "data" value includes field delimiter) '99' (Other)	Code to identify reason for reject.
58		N	String		Message to explain reason for rejection.
<Standard Message Trailer>		Y			

4.FIX session establishment and termination scenario

Establish connection

The FIX client (initiator) sends a Logon message with SenderCompID and Password for Trading System in order to establish connection with server (acceptor). The acceptor will authenticate the identity of the initiator by examining the Logon message. The Logon message will contain the data necessary to support the previously agreed upon authentication method. If the initiator is successfully authenticated, the acceptor responds with a Logon message. If authentication fails, the session acceptor shuts down the connection and sending message to indicate the reason of failure. The session initiator may begin to send messages immediately following the Logon message, however, the acceptor may not be ready to receive them. The initiator must wait for the confirming Logon message from the acceptor before declaring the session fully established.

After the initiator has been authenticated, the acceptor will respond with a confirming Logon message. The initiator side will use the Logon message being returned from the acceptor as confirmation that a FIX session has been established. The confirming Logon message from MOEX normally can take up to 3 seconds to receive. Please allow 3 seconds waiting time for reply.

After authentication, the initiator and acceptor must synchronize their messages through interrogation of the MsgSeqNum field before sending any queued or new messages. A comparison of the MsgSeqNum in the Logon message to the internally monitored next expected sequence number will indicate any message gaps. Likewise, the initiator can detect gaps by comparing the acknowledgment Logon message's MsgSeqNum to the next expected value. The section on message recovery later in this document deals with message gap handling.

Comments: FIX client should send Logon message with MsgSeqNum (34) = 1 each new day. FIX client should send Logon message with MsgSeqNum (34) = sequence number of the last message in out log + 1 establishing the next session on the same day.

If FIX client sends to server a Logon (A) message with ResetSeqNumFlag='Y', then it will not receive Execution Reports (8) for events which took place before the session is established.

Resend messages mechanism

During initialization, or in the middle of a FIX session, message gaps may occur which are detected via the tracking of incoming sequence numbers. The following section provides details on how to recover messages.

As previously stated, each FIX participant (FIX client or OTCT FIX server) must maintain two sequence numbers for each FIX session, each for incoming and outgoing messages. Each message is assigned a unique sequence number, which is incremented after the message. Likewise, every received message has a unique sequence number and the incoming sequence counter is incremented after each message.

When the incoming sequence number does not match the expected number corrective processing is required.
If the incoming message has a sequence number less than expected and the PossDupFlag is not set, it indicates a serious error. We strongly recommend to terminate session and manual intervention be initiated.

If the incoming sequence number is greater than expected, it indicates that messages were missed and retransmission of the messages is requested via the Resend Request (2) message.

Each side of connection expects to receive message with sequence number, which is equal to sequence number of the last message in his out log + 1. In this case, the side, which detects gaps, should send Resend Request (2) message with a range of missed messages.

The resend request can be used to request a single message, a range of messages or all messages subsequent to a particular message.

- To request a single message: BeginSeqNo = EndSeqNo;
- To request a range of messages: BeginSeqNo = first message of range, EndSeqNo = last message of range;
- To request all messages subsequent to a particular message: BeginSeqNo = first message of range, EndSeqNo = 0 (represents infinity).

Session status management

The Heartbeat monitors the status of the communication link and identifies cases when the last message of a string was not received.
During periods of message inactivity, FIX applications will generate Heartbeat messages at regular time intervals. The heartbeat monitors the status of the communication link and identifies incoming sequence number gaps. The session initiator using the HeartBtInt field in the Logon message declares the Heartbeat Interval. The heartbeat interval timer should be reset after every message is transmitted (not just heartbeats). The HeartBtInt value should be agreed upon by the two firms and specified by the Logon initiator and echoed back by the Logon acceptor. Note that the same HeartBtInt value is used by both sides, the Logon “initiator” and Logon “acceptor”.

When either end of a FIX connection has not sent any data for [HeartBtInt] seconds, it will transmit a Heartbeat message. When either end of the connection has not received any data for (HeartBtInt + “some reasonable transmission time”) seconds, it will transmit a test request message. If there is still no heartbeat message received after (HeartBtInt + “some reasonable transmission time”) seconds then the connection should be considered lost and corrective action be initiated. If HeartBtInt is set to zero no regular heartbeat messages will be generated. Note that a test request message can still be sent independent of the value of the HeartBtInt, which will force a Heartbeat message.

Heartbeats issued as the result of Test Request must contain the TestReqID transmitted in the Test Request message. This is useful to verify that the Heartbeat is the result of the Test Request and not as the result of a regular timeout.

Reset sequence numbers

MOEX automatically resets sequence numbers (MsgSeqNum) at the start of each day. It means that sequence numbers of messages should start from 1 each new day.

FIX client (initiator) may request to reset sequence number of messages (MsgSeqNum (34)) during a trading day. In this case, it is recommended for the initiator to send a TestRequest and wait for a Heartbeat in response to ensure there are no sequence number gaps. Once the Heartbeat has been received, the initiator should send a Logon with ResetSeqNumFlag set to Y and with MsgSeqNum of 1. The acceptor should respond with a Logon with ResetSeqNumFlag set to Y and with MsgSeqNum of 1. At this point new messages from either side should continue with MsgSeqNum of 2. It should be noted that once the initiator sends the Logon with the ResetSeqNumFlag set, the acceptor must obey this request and the message with the last sequence number transmitted "yesterday" may no longer be available.

In case OTCT FIX server cannot correctly resend missed messages via Sequence Reset – Gap Fill mode, for example after an unrecoverable application failure, it may request to increase sequence number of messages via sending Sequence Reset (2) message with GapFillFlag (123) = N (Sequence Reset) and NewSeqNo (36) = <new sequence number>. Note that the use of Sequence Reset – Reset may result in the possibility of losing messages.

Close (Terminate) FIX session

In order to close FIX session FIX client should send Logout (5) message.

The logout message initiates or confirms the termination of a FIX session. Disconnection without the exchange of logout messages should be interpreted as an abnormal condition. Before actually closing the session, the logout initiator should wait for the opposite side to respond with a confirming logout message. This gives the remote end a chance to perform any Gap Fill operations that may be necessary. The session may be terminated if the remote side does not respond in an appropriate timeframe.

After sending the Logout message, the logout initiator should not send any messages unless requested to do so by the logout acceptor via a ResendRequest.

Reestablish session after failure

There are certain mechanisms of FIX session reestablishment:

1) In case connection was broken but FIX client didn't lose its logs the following steps should be taken in order to reestablish FIX session:
   a) Send Logon (A) message with sequence number (MsgSeqNum (34)) = sequence number of the last message in out log + 1;
   b) If OTCT FIX server confirms logon and sends Logon (A) message with sequence number greater than expected, then send Resend Request (2) message with a range of missed messages;
   c) In this case OTCT FIX server resends all missed messages to FIX client.
2) In the case of serious failure when FIX client lost his logs the following steps should be taken in order to reestablish FIX session:
   a) The first way:
      i) Send Logon (A) message with sequence number (MsgSeqNum (34)) = 1 and ResetSeqNumFlag (141) = 'Y';
      ii) If OTCT FIX server confirms logon and sends Logon (A) message with MsgSeqNum (34)) = 1 and ResetSeqNumFlag (141) = 'Y', then send Order Status Request (H) for each order in question.
   b) The second way:
      i) Send Logon (A) message with sequence number (MsgSeqNum (34)) = 1;

ii) If OTCT FIX server confirms logon and sends Logon (A) message with Text (58) =”MsgSeqNum too low, expecting X but received Y” send Logon (A) message with sequence number (MsgSeqNum (34)) = X;
iii) Send Resend Request (2) message with a range of missed messages;
iv) In this case OTCT FIX server resends all missed messages to FIX client.

3) In order to get order status for particular order Order Status Request (H) message with ClrOrdID or OrderID fields should be sent.

5.Messages from Client to Server

Order Status Request (H)

The Order Status Request (H) message is used by FIX client to request current order status.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = H
37	OrderID	C*	String		The OrderID of the order whose status is being requested.
11	ClOrdID	N*	String (20)		Arbitrary string (the maximum length is 20 characters). Tag is required under the standard protocol FIX 4.4, but MOEX does not support getting information about the status of orders on the user identifier of the order.
55	Symbol	Y	String (12)		Ticker symbol. The MOEX internal instrument identifier, SecCode
54	Side	Y	char	'1' (Buy) '2' (Sell)	Side of the order. In case of swap order (currency market) Side is set on the base of spot with the later settlement date.
<Standard Message Trailer>		Y			

New Order Single (D)

New Order – Single (D) message is used to submit new order.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = D
11	ClOrdID	C*	String (20)		Unique identifier of the order as assigned by institution or by the intermediary (CIV term, not a hub/service bureau) with closest association with the investor. Note: this field's value is sent in the BROKERREF field of native MOEX Order message. Please consult your broker how to format this field according to broker preferences.
1	Account	Y*	String (12)		Account mnemonic as agreed between buy and sell sides, e.g. broker and institution or investor/intermediary and fund manager. Is used to represent trading account.
«Parties»		N			Parties of the order. Usually contains client code. Note: It's recommended to define client code only for broker's client accounts, otherwise client code will be ignored by trading system and won't be returned in execution Reports, except the first message for this order.
38	OrderQty	Y*	Qty(10)		Quantity ordered, expressed in number of lots. Lot size is different for different Symbol + Board combinations and should be determined from the marketdata feeds.
«Instrument»					This block contains Symbol (55) field for a security of the order (SECCODE in native interface). Note: FIX gateway checks that tags 336 and 55 combination points to existing security. If there is no match, the order is rejected with 'Unknown Security' error message.
40	OrdType	Y	char	'2' (Limit)	Order type.
44	Price	Y	Price (9)		Order price. Expressed in 'per unit of quantity' or 'percentage' (for bonds). Required for limit OrdTypes. Maximum allowed length of Price field is 10 characters, including decimal point. Orders with price that does not fit in minimal price steps levels will be rejected

54	Side	Y	char	'1' (Buy) '2' (Sell)	Side of the order. In case of swap order (currency market) Side is set on the base of spot with the later settlement date.
59	TimeInForce	Y	char	'3' (Immediate or Cancel)	Specifies how long the order remains in effect.
60	TransactTime	Y	UTCTimestamp		Time this order request was initiated/released by the trader, trading system, or intermediary. Required by FIX protocol but not processed at MOEX.
386	NoTradingSessions	Y*	NumInGroup	'1'	Specifies the number of repeating TradingSessionIDs. TradingSessionIDs group should contain only one element of group. Note: tags 386 and 336 compose a group and should be placed exactly in the order 386, then 336, and not separated by other tags.
336	=> TradingSessionID	Y*	String(4)		Identifier for Trading Session which contains MOEX security board (SECBOARD).
<Standard Message Trailer>		Y			

Market Data Request (V)

Market Data Request (MsgType = 'V') is used to subscribe for- or unsubscribe from- market data.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = 'V'
262	MDReqID	Y	String		New unique identifier of Market Data Request message should be generated.
263	SubscriptionRequestType	Y	char	'0' (Snapshot) '1' (Snapshot + Updates (Subscribe)) '2' (Unsubscribe)	Request type
264	MarketDepth	Y	char	'0' (Full Book)	Request depth.
265	MDUpdateType	N	char	'0' (Full Refresh)	Update type.
386	NoTradingSessions	Y	NumInGroup	'1'	Specifies the number of repeating TradingSessionIDs.

336	TradingSessionID	Y	String	‘OTCT’	Identifier for Trading Session which contains MOEX security board (SECBOARD).
146	NoRelatedSym	Y	NumInGroup		Specifies the number of repeating Instruments
=> <Instrument>		Y			
267	NoMDEntryTypes	Y	NumInGroup		Specifies the number of repeating entry types
269	=> MDEntryType	Y	char	‘0’ (Buy quotes); ‘1’ (Sell quotes);	Order type
<Standard Message Trailer>		Y			

6.Messages from Server to Client

Execution Report (8)

The Execution Report (8) message is used to:

- Confirm the receipt of an order;
- Relay order status information;
- Relay fill information on working orders;
- Reject orders or incorrect order status request or order cancel request.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			MsgType = '8'
11	ClOrdID	Y	String(20)		Unique identifier for Order within a day for a given SenderCompID as assigned by the buy-side (institution, broker, intermediary etc.). This field contains value of BROKERREF field of MOEX trading system NEGDEALS table. The value follows the value of this tag in the originating New Order Single message Note: ClOrdID can be nonunique in the case when order was not placed via OTCT FIX
6	AvgPx	Y	Price	0	Calculated average price of all fills on this order. Always '0' for MOEX.
1	Account	Y*	String		Account mnemonic as agreed between buy and sell sides, e.g. broker and institution or investor/intermediary and fund manager. Is used to represent trading account. Corresponds to the ACCOUNT field of MICEX Bridge API.
14	CumQty	Y	Qty		Currently executed shares for chain of orders.

17	ExecID	Y	String		Unique identifier of execution message as assigned by sell-side (broker, exchange, ECN)
31	LastPx	C	Price		Price of this (last) fill. Required if ExecType (150) = Trade.
32	LastQty	C	Qty		Quantity (e.g. shares) bought/sold on this (last) fill. Required if ExecType (150) = Trade.
37	OrderID	Y	String		OrderID assigned by MOEX. Unique across the given market history.
38	OrderQty	Y*	Qty(10)		Quantity ordered, expressed in number of lots.
39	OrdStatus	Y		'0' (New) '1' (Partially filled) '2' (Filled) '4' (Canceled) '8' (Rejected)	Describes the current state of a CHAIN of orders, same scope as OrderQty, CumQty, LeavesQty, and AvgPx.
103	OrdRejReason	N	Int	‘99’ Other	Reject reason
44	*Price*	Y	Price		Order price expressed in 'per unit of quantity'. Required if specified on the order.
54	Side	Y	Char	'1' \ 'B' (Buy) '2' \ 'S' (Sell)	Side of order.
55	Symbol	Y	String (12)		Ticker symbol. The MOEX internal instrument identifier, SecCode
58	Text	N	String		Free format text string.
60	TransactTime	*C**	UTCTimestamp		The time at which the transaction was registered
150	ExecType	Y	char	'0' (New) '4' (Canceled) '8' (Rejected)	Describes the purpose of the execution report.

				'F' (Trade) 'I' (Order status)	
151	LeavesQty	Y	Qty		Quantity open for further execution (order balance). LeavesQty is always “0” in spot Execution Reports of swap orders. LeavesQty is always “0” if OrdStatus (39) in (‘4’, ‘8’)'. Else LeavesQty = OrderQty – CumQty.
103	OrdRejReason	N	Int	2 (Exchange closed) 3 (Order exceeds limit) 5 (Unknown Order) 6 (Duplicate Order) 13 (Incorrect quantity) 15 (Unknown account(s)) 99 (Other)	Code to identify reason for order rejection.
336	TradingSessionID	Y*	String	‘OTCT’	Identifier for Trading Session which contains MOEX security board (SECBOARD).
790	OrdStatusReqID	N	String		Can be used to uniquely identify a specific Order Status Request message.
5459	OptionSettlType	N	String		MOEX settlement code for a trade
<Standard Message Trailer>		Y			

Market Data Snapshot (W)

The Market Data messages are used as a response to a Market Data Request (V) message.

Tag	Field name	Required	Type	Valid values	Comments
<Standard Message Header>		Y			Type cYYбщеNия = 'W'

55	Symbol	Y	String (12)		Ticker symbol. The MOEX internal instrument identifier, SecCode
262	MDReqID	Y	String		Identifier of corresponding Market Data Request message is returned.
336	TradingSessionID	Y*	String	‘OTCT’	Identifier for Trading Session which contains MOEX security board (SECBOARD).
268	NoMDEntries	Y	NumInGroup		Specifies the number of entry types in repeating group
=>269	MDEntryType	Y	char	‘0’ (Buy quote); ‘1’ (Sell quote);	Market Data entry type
=>270	MDEntryPx	Y	Price		Price of OrderBook entry.
=>271	MDEntrySize	Y	Qty		Quantity of OrderBook entry.