Drop Copy Gateway FIX Specification

Please respond to:

newtradingplatform@lme.com

Table of Contents

Document History

Version	Date	Change Description
1.0	31/12/2019	Initial draft
1.1	09/04/2020	Updated following internal review
1.2	28/01/2022	Internal review
1.3	24/03/2023	Internal review
1.4	23/06/2023	3.9.1 added RelatedHighPrice (1819) and RelatedLowPrice (1820) for unsolicited cancellation
1.5	13/10/2023	1.1.2 password encryption example added 3.2 timestamp precision 3.6.7 and 3.7.1 Text (58) is conditionally required if reject reason is Other 3.8.1 301 description
1.6	15/03/2024	1.1.4.1 password reuse policy 1.2 duplicate connection termination removed 3.6.1 EncryptedPassword (1402) and EncryptedNewPassword (1404) length 450 3.8.1 LEI usage 3.9.1 ExecID description
1.7	19/07/2024	1.1.2 public key location 3.2 added String data type 3.8.1 PartyID format where Alphanumeric changed to String 3.9.1 corrected cross-reference
1.7.1	16/09/2024	2.8 Technical Halt
1.8	29/10/2024	2.8 Technical Halt 3.2 String data type 3.9.1 removed pending OrdStatus values
1.8.1	04/02/2025	3.2 special character usage

Preface

This document describes the LME implementation of the FIX protocol based on FIX 5.0 SP2 Specification with relevant extension packs.

The document assumes the reader has an understanding of the FIX protocol, see http://www.fixprotocol.org/.

This document should be read in conjunction with related materials on LME.com for LMEselect v10.

Delivery Phasing

This document covers all the functionality available in LMEselect 10 however functionality will be delivered in phased releases.

Functionality that will be included in a later release is specified in the following table and shown throughout the document in *dark grey italics*. The initial release will contain all functionality that is **not** specified in the table.

Function	Reference
Order types: • **Market** • **Stop Market** • **Iceberg** • **Post Only** • **One Cancels Other**	3.9.1 Execution Report (8)
Order validity condition: • **Fill or Kill (FOK)**	
Speed Bumps	
Self Execution Prevention	
Mass Quote	

1 Session Management

1.1 Authentication

1.1.1 Comp ID

A FIX session is established by sending a Logon (35=A) request which includes the sender and the target in the Message Header:

- SenderCompID (49) – the party initiating the session.
- TargetCompID (56) – the acceptor of the session as per configuration.

For messages sent to the Exchange, the client should use the session CompID allocated by the Exchange to populate SenderCompID (49). A single client may have multiple connections to the gateway i.e. multiple FIX sessions, each with its own Comp ID.

The TargetCompID (56) in messages sent to the Exchange is environment specific as follows:

Production:

- DCLME

Member Test environments:

- DCLMEMTA
- DCLMEMTB.

1.1.2 Password Encryption

The client should specify their password in EncryptedPassword (1402) in the Logon request.

To encrypt the password, the client is expected to use a 2048-bit RSA (http://en.wikipedia.org/wiki/RSA_(algorithm)) public key circulated by the Exchange on https://www.lme.com/Trading/Systems/LMEselect. The binary output of the RSA encryption must be represented in Big Endian PKCS #1 with padding scheme OAEP (https://en.wikipedia.org/wiki/PKCS_1) and then converted to an alphanumeric value by means of standard base-64 encoding (http://en.wikipedia.org/wiki/Base64) when communicating with the gateway.

Password encryption example:

```
public static String encrypt(String value) throws CrytographyException {
try {
Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-1AndMGF1Padding");
cipher.init(Cipher.ENCRYPT_MODE, publicKey);
byte [] bytes = cipher.doFinal(value.getBytes());
return Base64.getEncoder().encodeToString(bytes);
} catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | IllegalBlockSizeException | BadPaddingException e) {
throw new CrytographyException(e.getMessage());
}
}
…….
```

```
String pubKey = new String(keyBytes, "UTF-8");

pubKey = pubKey.replaceAll("(-+BEGIN PUBLIC KEY-+\\r?\\n|-+END PUBLIC KEY-
+\\r?\\n?)", "");

pubKey = pubKey.replaceAll("(-+BEGIN RSA PUBLIC KEY-+\\r?\\n|-+END RSA
PUBLIC KEY-+\\r?\\n?)", "");

pubKey = pubKey.replaceAll("\\n|\\r","");

KeyFactory keyFactory = KeyFactory.getInstance("RSA");

X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(pubKey.getBytes()));

publicKey = keyFactory.generatePublic(keySpec);
```

1.1.3 Password

The gateway authenticates the participant’s Logon (35=A) request and sends a Logon (35=A) response containing SessionStatus (1409) which indicates whether the logon attempt was successful or not.

Repeated failures in password validation will result in the client account being locked. The participant is expected to contact the Exchange to unlock the client account.

1.1.4 Change Password

Each new Comp ID will be assigned a password by the Exchange. The client is expected to change this password upon initial logon.

A password change can be made in a Logon (35=A) request. The client should specify the new encrypted password in EncryptedNewPassword (1404) and the current encrypted password in EncryptedPassword (1402).

The new password must comply with Exchange’s password policy. The status of the new password (i.e. whether it is accepted or rejected) will be specified in the SessionStatus (1409) response from the gateway. The new password, if accepted, will be effective for subsequent logins.

1.1.4.1 Password Policy

The Exchange requires the password to contain:

- Minimum of 8 characters
- At least one number
- Combination of uppercase and lowercase characters.

Password history is retained and therefore the last 24 passwords cannot be reused.

1.2 Establishing a FIX Session

The client must wait for a successful Logon response before sending additional messages. If additional messages are received from the client before the exchange of Logon messages, the TCP/IP connection with the client will be disconnected.

If a Logon (35=A) attempt fails for the following reasons, the gateway will send a Logout (35=5) or a Reject (35=3) and then terminate the session:

- Password failure
- Comp ID is locked
- Logon is not permitted during this time

For all other reasons, including the following, the gateway will terminate the session without sending a Logout or Reject:

- Invalid Comp ID

Inbound message sequence number will not be incremented if the connection is abruptly terminated due to the logon failure.

If a session level failure occurs due to a message sent by the client which contains a sequence number that is less than what is expected and the PossDup (43) is not set to Y = Yes, then the gateway will send a Logout (35=5) and terminate the FIX session. In this scenario the inbound sequence number will not be incremented.

1.3 Message Sequence Numbers

As outlined in the FIX protocol, the client and gateway will each maintain a separate and independent set of incoming and outgoing message sequence numbers. Sequence numbers should be initialized to 1 (one) at the start of the day and be incremented throughout the session. Either side of a FIX session will track the:

- NextExpectedMsgSeqNum (789) (starting at 1) in Logon (35=A)
- MsgSeqNum (34) in the Message Header (starting at 1); with respect to the contra-party.

The MsgSeqNum (34) in the Message Header is always incremented by the sender, whereas the NextExpectedMsgSeqNum (789) is only updated as a result of an incoming message.

Monitoring sequence numbers will enable parties to identify and react to missed messages and to gracefully synchronize applications when reconnecting during a FIX session.

Any message sent by either side of a FIX session will increment the sequence number unless explicitly specified for a given message type.

If any message sent by one side of a FIX session contains a sequence number that is LESS than the NextExpectedMsgSeqNum (789) then the other side of this session is expected to send a Logout message and terminate the FIX connection immediately, unless the PossDup flag is set to Y = Yes

A FIX session will not be continued to the next trading day. Both sides are expected to initialize (reset to 1) the sequence numbers at the start of each day. At the start of each trading day if the client starts with a NextExpectedMsgSeqNum (789) greater than 1 then the gateway will send a Logout message and terminate the session immediately without any further exchange of messages.

1.4 Heartbeat and Test Request

The client and the gateway will use the Heartbeat (35=0) message to monitor the communication line during periods of inactivity and to verify that the interfaces at each end are available.

The gateway will send a Heartbeat anytime it has not transmitted a message for the heartbeat interval. The client is expected to employ the same logic.

If the gateway detects inactivity for a period longer than 3 heartbeat intervals, it will send a Test Request message to force a Heartbeat from the client. If a response to the Test Request (35=1) is not received within a reasonable transmission time (recommended being an elapsed time equivalent to 3 heartbeat intervals), the gateway will send a Logout (35=5) and break the TCP/IP connection with the client. The client is expected to employ similar logic if inactivity is detected on the part of the gateway.

1.5 Terminating a FIX Session

Session termination can be initiated by either the gateway or the client by sending a Logout (35=5). Upon receiving the Logout request, the contra party will respond with a Logout message signifying a Logout reply. Upon receiving the Logout reply, the receiving party will terminate the connection.

If the contra-party does not reply with either a Resend Request or a Logout reply, the Logout initiator should wait for 60 seconds prior to terminating the connection.

The client is expected to terminate each FIX connection at the end of each trading day before the gateway is shut down. Any open FIX connections will be terminated by the gateway sending a Logout when the service is shut down. Under exceptional circumstances, for example, a slow consumer, the gateway may initiate the termination of a connection during the trading day by sending a Logout.

If, during the exchange of Logout messages, the client or the gateway detects a sequence gap, it should send a Resend Request.

1.6 Re-establishing a FIX Session

If a FIX connection is terminated during the trading day it may be re-established via an exchange of Logon messages.

Once the FIX session is re-established, the message sequence numbers will continue from the last message successfully transmitted prior to the termination as described in 2.7 Transmission of Missed Messages.

1.7 Sequence Reset

Gap-fill mode can be used by one side when skipping session level messages which can be ignored by the other side.

During a FIX session the gateway or the client may use the Sequence Reset (35=4) message in Gap Fill mode if either side wishes to increase the expected incoming sequence number of the other party.

It will not be possible to reset the client sequence number to 1 using the Logon message. Should a reset be required the participant should contact the Exchange.

The client is required to support a manual request by Exchange to initialize sequence numbers prior to the next login attempt.

1.8 Fault Tolerance

After a failure on client side or on gateway side, the client is expected to be able to continue the same session.

If the sequence number is reset to 1 by the gateway, all previous messages will be available for recovery by the client side. Messages that are resent may contain PossResend (97) as described in 2.5 Possible Resends.

1.9 Checksum Validation

The gateway performs a checksum validation on all incoming messages into the input services. Incoming messages that fail the checksum validation will be rejected and the connection will be dropped by the gateway without sending a logout.

Conversely, in case of a checksum validation failure, the client is expected to drop the connection and take any appropriate action before reconnecting.

Messages that fail the checksum validation should not be processed.

2 Recovery

2.1 General Message Recovery

Message gaps may occur which are detected via the tracking of incoming sequence numbers. Recovery will be initiated if a gap is identified when an incoming message sequence number is found to be greater than NextExpectedMsgSeqNum (789) during Logon or the MsgSeqNum (34) at other times.

The Resend Request will indicate the BeginSeqNo (7) and EndSeqNo (16) of the message gap identified and when replying to a Resend Request, the messages are expected to be sent strictly honouring the sequence.

If messages are received outside of the BeginSeqNo and EndSeqNo, then the recovering party is expected to queue those messages until the gap is recovered.

During the message recovery process, the recovering party will increment the Next Expected Sequence number accordingly based on the messages received. If messages applicable to the message gap are received out of sequence then the recovering party will drop these messages.

The party requesting the Resend Request can specify “0” in the EndSeqNo to indicate that they expect the sender to send ALL messages starting from the BeginSeqNo. In this scenario, if the recovering party receives messages with a sequence greater than the BeginSeqNo, out of sequence, the message will be ignored.

Administrative messages such as Sequence Reset, Heartbeat and Test Request which can be considered irrelevant for a retransmission could be skipped using the Sequence Reset message in gap-fill mode. Note that the gateway expects the client to skip Sequence Reset messages when replying to a Resend Request at all times.

When resending messages, the gateway would use either PossDup or PossResend flag to indicate whether the messages were retransmitted earlier.

If PossDup flag is set to Y = Yes, it indicates that the same message with the given sequence number with the same business content may have been transmitted earlier.

In the case where PossResend flag is set to Y = Yes, it indicates that the same business content may have been transmitted previously but under the different message sequence number. In this case business contents needs to be processed to identify the resend. For example, in Execution Reports the ExecID (17) may be used for this purpose.

2.2 Resend Request

The client may use the Resend Request (35=2) message to recover any lost messages. This message may be used in one of three modes:

1. To request a single message. The BeginSeqNo and EndSeqNo should be the same.
2. To request a specific range of messages. The BeginSeqNo should be the first message of the range and the EndSeqNo should be the last of the range.
3. To request all messages after a particular message. The BeginSeqNo should be the sequence number immediately after that of the last processed message and the EndSeqNo should be zero (0).

2.3 Logon Message Processing – Next Expected Message Sequence

The session initiator should supply the NextExpectedMsgSeqNum (789) the value next expected from the session acceptor in MsgSeqNum (34). The session acceptor should validate the logon request including that NextExpectedMsgSeqNum (789) does not represent a gap. It then constructs its logon response with NextExpectedMsgSeqNum (789) containing the value next expected from the session initiator in MsgSeqNum (34) having incremented the number above the logon request if that was the sequence expected.

The session initiator must wait until the logon response is received in order to submit application messages. Once the logon response is received, the initiator must validate that NextExpectedMsgSeqNum (789) does not represent a gap.

In case of gap detection from either party (lower than the next to be assigned sequence) recover all messages from the last message delivered prior to the logon through the specified NextExpectedMsgSeqNum (789) sending them in order, then gap fill over the sequence number used in logon and proceed sending newly queued messages with a sequence number one higher than the original logon.

Neither side should generate a Resend Request (35=2) based on MsgSeqNum (34) of the incoming Logon message but should expect any gaps to be filled automatically by following the Next Expected Sequence processing described above.

Whilst the gateway is resending messages to the client, the gateway does not allow another Resend Request (35=2) from the client. If a new Resend Request is received during this time, the gateway will terminate the session immediately without sending the Logout (35=5) message.

Note that indicating the NextExpectedMsgSeqNum (789) in the Logon (35=A) is mandatory.

2.4 Possible Duplicates

The gateway handles possible duplicates according to the FIX protocol. The client and the gateway use the PossDupFlag (43) field to indicate that a message may have been previously transmitted with the same MsgSeqNum (34).

2.5 Possible Resends

The gateway may use the PossResend (97) field to indicate that an application message may have already been sent under a different MsgSeqNum (34). The client should validate the contents (e.g. ExecID (17)) of such a message against those of messages already received during the current trading day to determine whether the new message should be ignored or processed.

2.6 Gap Fills

The following messages are expected to be skipped using gap-fills when being retransmitted:

1. Logon
2. Logout
3. Heartbeat
4. Test Request

5. Resend Request
6. Sequence Reset

All other messages are expected to be replayed within a retransmission.

2.7 Transmission of Missed Messages

The Execution Report messages generated during a period when a client is disconnected from the gateway will be sent to the client when it next reconnects on the same business day. In the unlikely event the disconnection was due to a gateway outage the messages which will be retransmitted will include a PossResend (97) set to Y = Yes.

2.8 Technical Halt

In the event of a system component failure, a technical halt will be applied and the Market Data service will publish the Trading State = Technical Halt. On receipt of this message, market participants are required to clear their public and private order books of all orders including persisted orders. Order cancellations will not be transmitted by the gateway.

3 Message Definitions

3.1 Supported Message Types

3.1.1 Inbound Messages

- Logon (A)
- Heartbeat (0)
- Test Request (1)
- Resend Request (2)
- Sequence Reset (4)
- Logout (5)

3.1.2 Outbound Messages

- Logon (A)
- Heartbeat (0)
- Test Request (1)
- Resend Request (2)
- Sequence Reset (4)
- Logout (5)
- Reject (3)
- Business Message Reject (j)
- Execution Report (8)

3.2 Data Types

Data Types used are based on the published standard FIX specifications. The field length in characters is shown in brackets. The length of numeric fields is the number of digits in that value and not the size of the value in bytes. If a data type has a specific value this will be provided in the description.

Data Type	Format
UTCTimestamp (27)	Incoming YYYYMMDD-HH:mm:ss.SSSSSS YYYYMMDD-HH:mm:ss.SSSSSSSSS Timestamps will be represented as UTC and accepted to microsecond or nanosecond precision
	Outgoing YYYYMMDD-HH:mm:ss.SSSSSSSSS

Data Type	Format
	Note: Timestamps will be represented as UTC up to microsecond precision with the nanosecond element being represented by trailing zeros.
Price (20)	Can be up to 12 significant digits before the decimal point (with provision for a negative value) and at the most 6 decimal places For example, 1234567891234.567891 -123456789123.456789
String (*n*)	Permitted ASCII characters are A-Z, a-z, 0-9 Note, special characters are also permitted for the following attributes:

Tag / PartyID	Underscore	Hyphen
ClOrdID (11)	✔	✔
OrigClOrdID (41)	✔	✔
PartyID (448) for PartyRole (452)		
‘81’ Broker Client ID	✔	✔
‘11’ Order Origination Trader	✔	X
‘3’ Client ID (Proprietary/Custom)	✔	X
‘24’ Customer Account	✔	X

Note: Text in rejection reasons can contain other ASCII characters and spaces.

3.3 Required Fields

The following conventions are used for fields in the message definitions:

Y	Required by FIX
Y*	Required by LME
C	Conditionally required by FIX
C*	Conditionally required by LME

N	Not required / optional.

3.4 Message Header

Tag	Field Name	Req	Data Type	Description
8	BeginString	Y	String (8)	Always set to FIXT.1.1
9	BodyLength (4)	Y	Length	Message length, in bytes, forward to the CheckSum field. Maximum value 9999
35	MsgType	Y	String (3)	Defines message type.
1128	ApplVerID	N	String (1)	Version of FIX used in the message: 9 = FIX50SP2 Returned by the gateway
49	SenderCompID	Y	String (10)	Identifies the sender of the message, see Comp ID
56	TargetCompID	Y	String (10)	Identifies the receiver of the message see Comp ID
34	MsgSeqNum	Y	SeqNum (9)	Outbound message sequence number. Always incremented by the sender.
43	PossDupFlag	N	Boolean	Indicates whether the message was previously transmitted with the same MsgSeqNum (34). Absence of this field is interpreted as original transmission (N).
97	PossResend	N	Boolean	Indicates whether the message was previously transmitted under a different MsgSeqNum (34). Absence of this field is interpreted as original transmission (N).
52	SendingTime	Y	UTCTimestamp	Time the message was transmitted.
122	OrigSendingTime	C	UTCTimestamp	Conditionally Required for messages sent as a result of a ResendRequest.

Tag	Field Name	Req	Data Type	Description
				If the original time is not available, this should be the same value as SendingTime (52).

3.5 Message Trailer

Tag	Field Name	Req	Data Type	Description
10	CheckSum	Y	String (3)	Standard check sum described by FIX protocol. Always last field in the message; i.e. serves, with the trailing <SOH>, as the end-of-message delimiter. Always defined as three characters.

3.6 Administrative Messages

3.6.1 Logon (A)

The first messages exchanged in a FIX session are the Logon request and the Logon response. The main purposes of the Logon request and response are:

- To authenticate the client.
- To agree on the sequence numbers.

Tag	Field Name	Req	Data Type	Description
98	EncryptMethod	Y	Int	Method for encryption. Valid value is: 0 = None
108	HeartBtInt	Y	Int	Heartbeat interval in seconds.
789	NextExpectedMsgSeqNum	Y*	SeqNum (9)	Next expected MsgSeqNum (34) value to be received. Always updated as a result of an incoming message.
1400	EncryptedPasswordMethod	N	Int	Enumeration defining the encryption method used to encrypt password fields: 101 = RSA

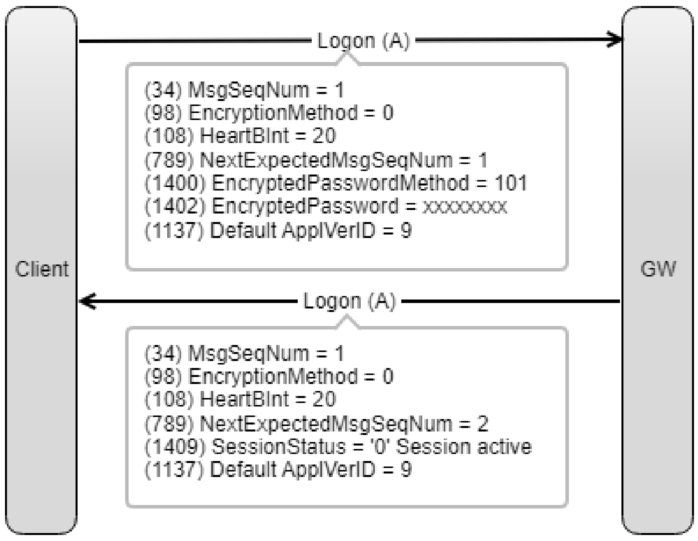
Tag	Field Name	Req	Data Type	Description
1402	EncryptedPassword	Y	Data (450)	Encrypted password – encrypted via the method specified in EncryptedPasswordMethod (1400)
1404	EncryptedNewPassword	N	Data (450)	Encrypted new password – encrypted via the method specified in EncryptedPasswordMethod (1400)
1137	DefaultApplVerID	Y	String (1)	The default version of FIX being used in this session: 9 = FIX50SP2

A Logon message is returned in response to an incoming Logon message to initiate a FIX session. The SessionStatus (1409) indicates whether the logon attempt was successful or not.

Tag	Field Name	Req	Data Type	Description
98	EncryptMethod	Y	Int	Method for encryption. Valid value is: 0 = None
108	HeartBtInt	Y	Int	Heartbeat interval in seconds.
789	NextExpectedMsgSeqNum	Y*	SeqNum (9)	Next expected MsgSeqNum (34) value to be received. Always updated as a result of an incoming message.
1409	SessionStatus	N	Int	Status of the FIX session. Valid values: 0 = Session active 1 = Session password changed
1137	DefaultApplVerID	Y	String (1)	The default version of FIX being used in this session: 9 = FIX50SP2

Example Message Flow

Initial Logon


Client
GW
Logon (A)
(34) MsgSeqNum = 1
(98) EncryptionMethod = 0
(108) HeartBInt = 20
(789) NextExpectedMsgSeqNum = 1
(1400) EncryptedPasswordMethod = 101
(1402) EncryptedPassword = xxxxxxxx
(1137) Default ApplVerID = 9
Logon (A)
(34) MsgSeqNum = 1
(98) EncryptionMethod = 0
(108) HeartBInt = 20
(789) NextExpectedMsgSeqNum = 2
(1409) SessionStatus = '0' Session active
(1137) Default ApplVerID = 9

3.6.2 Heartbeat (0)

Heartbeat (35=0) is sent at the interval specified in HeartBtInt (108) in Logon (35=A). It is also sent in response to a Test Request (35=1).

Tag	Field Name	Req	Data Type	Description
112	TestReqID	C	String (20)	Conditionally required if the heartbeat is a response to a Test Request (1). The value in this field should echo the TestReqID (112) received in the Test Request (1).

3.6.3 Test Request (1)

Test Request (35=1) can be sent by either the client or gateway to verify a connection is active. The recipient responds with a Heartbeat (35=0).

Tag	Field Name	Req	Data Type	Description
112	TestReqID	Y	String (20)	Identifier included in Test Request message to be returned in resulting Heartbeat (0).

3.6.4 Resend Request (2)

Resend Request (35=2) is used to initiate the retransmission of messages if a sequence number gap is detected.

To request a single message. The BeginSeqNo and EndSeqNo should be the same.

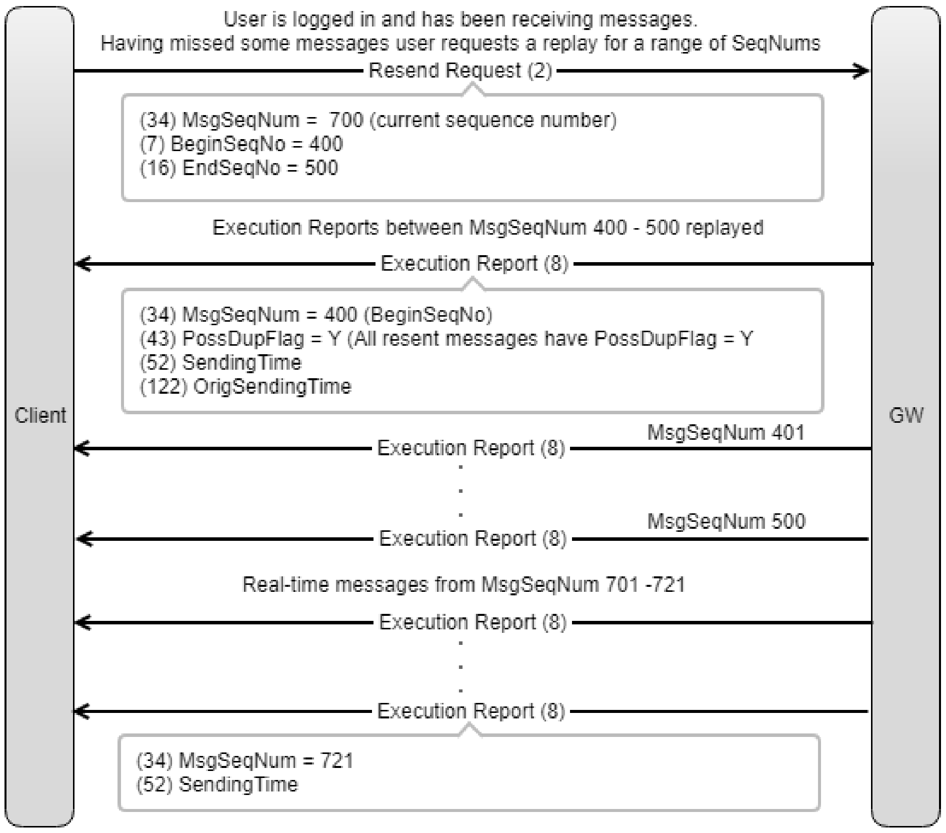
To request a specific range of messages. The BeginSeqNo should be the first message of the range and the EndSeqNo should be the last of the range.

To request all messages after a particular message. The BeginSeqNo should be the sequence number immediately after that of the last processed message and the EndSeqNo should be zero (0).

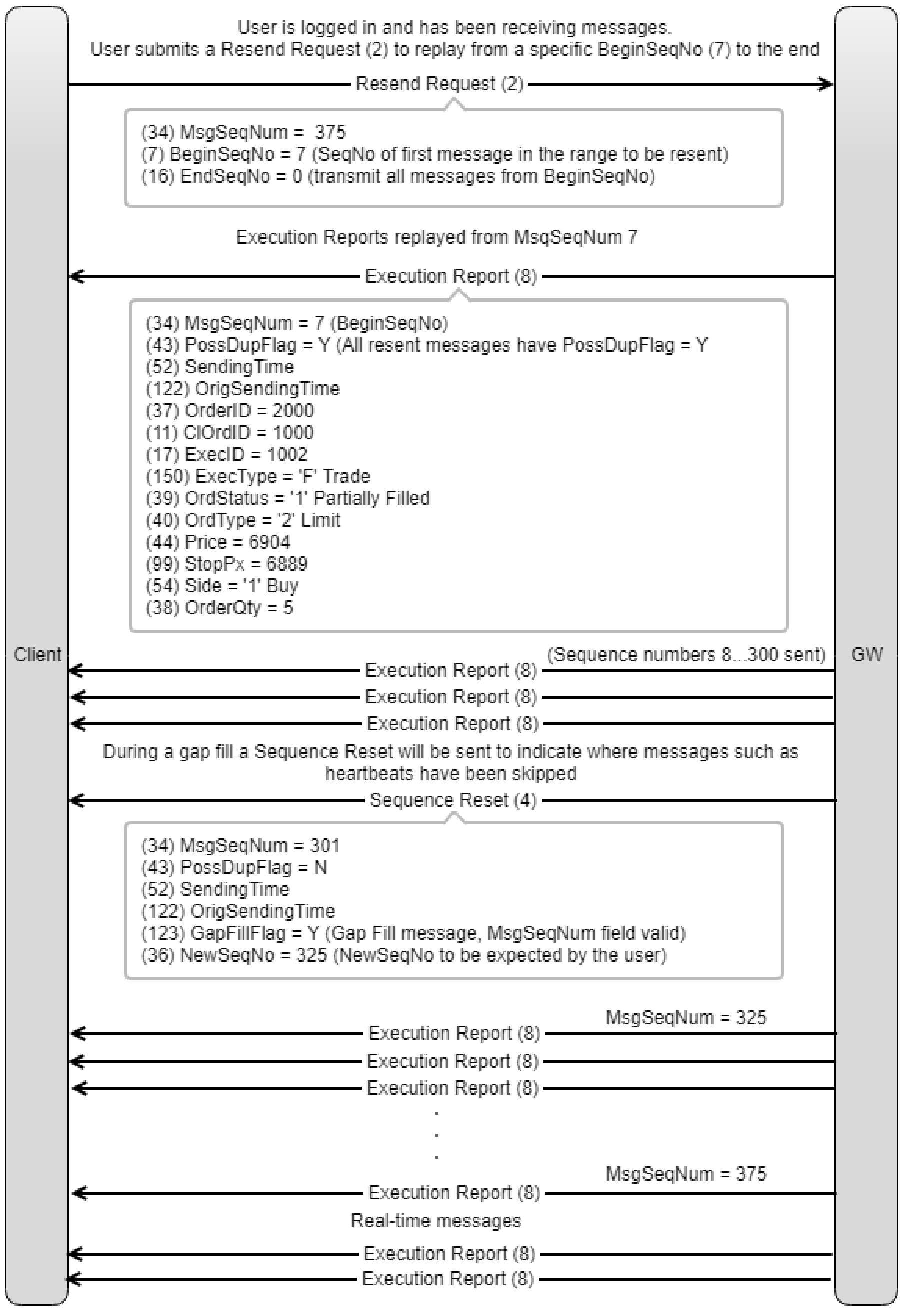
Tag	Field Name	Req	Data Type	Description
7	BeginSeqNo	Y	SeqNum (9)	Message sequence number of the first message in the range to be resent.
16	EndSeqNo	Y	SeqNum (9)	Sequence number of the last message expected to be resent. This may be set to 0 to request the sender to transmit ALL messages starting from BeginSeqNo (7).

Example Message Flow

Resend Request for a range of messages

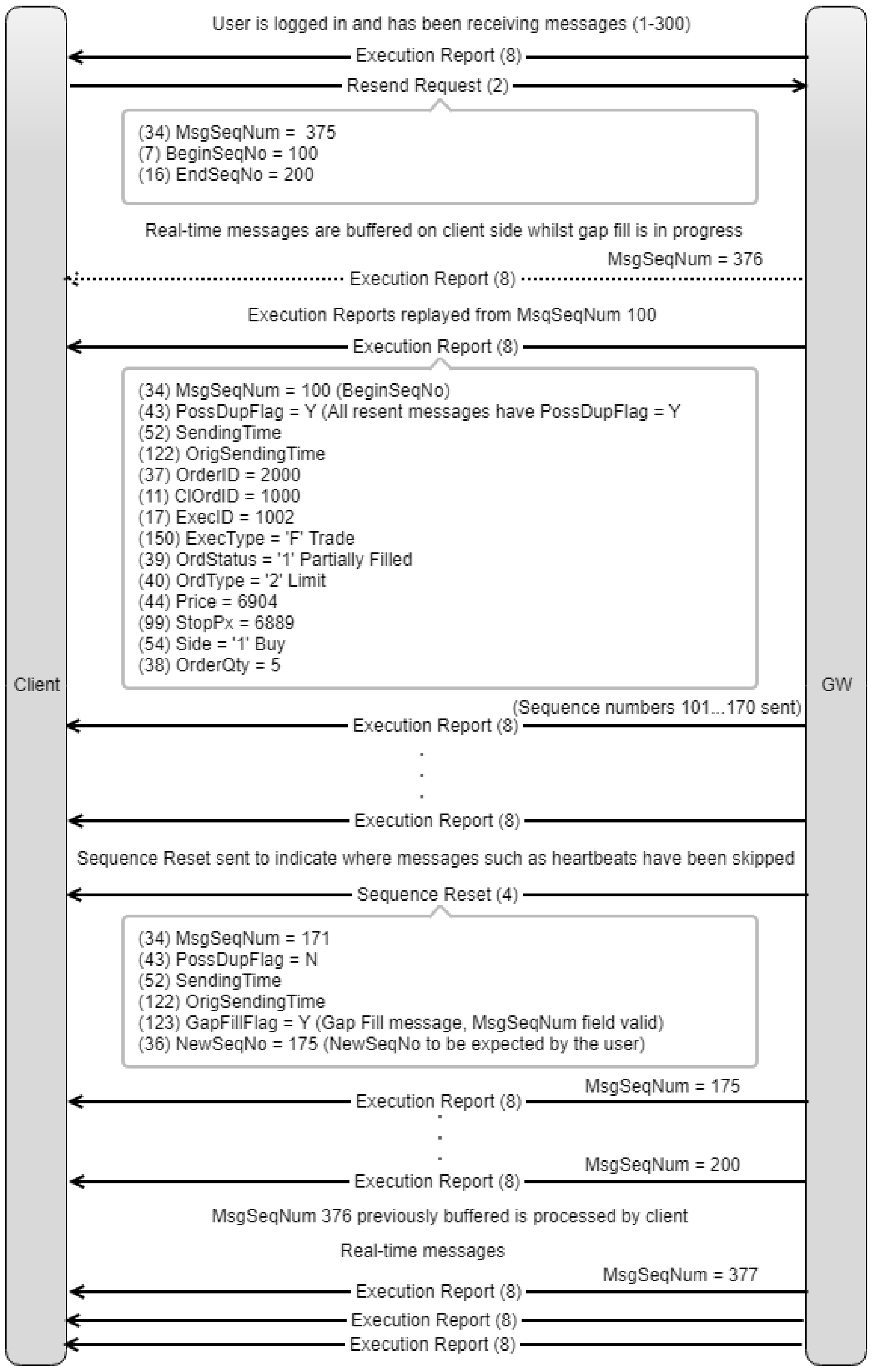

User is logged in and has been receiving messages.
Having missed some messages user requests a replay for a range of SeqNums
Resend Request (2)
(34) MsgSeqNum = 700 (current sequence number)
(7) BeginSeqNo = 400
(16) EndSeqNo = 500
Execution Reports between MsgSeqNum 400 - 500 replayed
Execution Report (8)
(34) MsgSeqNum = 400 (BeginSeqNo)
(43) PossDupFlag = Y (All resent messages have PossDupFlag = Y
(52) SendingTime
(122) OrigSendingTime
Client
GW
Execution Report (8)
MsgSeqNum 401
Execution Report (8)
MsgSeqNum 500
Real-time messages from MsgSeqNum 701 -721
Execution Report (8)
Execution Report (8)
(34) MsgSeqNum = 721
(52) SendingTime

Resend Request for all messages after a particular message


Client
GW
User is logged in and has been receiving messages.
User submits a Resend Request (2) to replay from a specific BeginSeqNo (7) to the end
Resend Request (2)
(34) MsgSeqNum = 375
(7) BeginSeqNo = 7 (SeqNo of first message in the range to be resent)
(16) EndSeqNo = 0 (transmit all messages from BeginSeqNo)
Execution Reports replayed from MsgSeqNum 7
Execution Report (8)
(34) MsgSeqNum = 7 (BeginSeqNo)
(43) PossDupFlag = Y (All resent messages have PossDupFlag = Y
(52) SendingTime
(122) OrigSendingTime
(37) OrderID = 2000
(11) ClOrdID = 1000
(17) ExecID = 1002
(150) ExecType = 'F' Trade
(39) OrdStatus = '1' Partially Filled
(40) OrdType = '2' Limit
(44) Price = 6904
(99) StopPx = 6889
(54) Side = '1' Buy
(38) OrderQty = 5
(Sequence numbers 8...300 sent)
Execution Report (8)
Execution Report (8)
Execution Report (8)
During a gap fill a Sequence Reset will be sent to indicate where messages such as heartbeats have been skipped
Sequence Reset (4)
(34) MsgSeqNum = 301
(43) PossDupFlag = N
(52) SendingTime
(122) OrigSendingTime
(123) GapFillFlag = Y (Gap Fill message, MsgSeqNum field valid)
(36) NewSeqNo = 325 (NewSeqNo to be expected by the user)
MsgSeqNum = 325
Execution Report (8)
Execution Report (8)
Execution Report (8)
MsgSeqNum = 375
Execution Report (8)
Real-time messages
Execution Report (8)
Execution Report (8)

Resend Request - incoming message buffered by Client

A Resend Request is submitted but before gap fill messages have been transmitted an incoming message is received. The client will hold the message until all the gap fill messages have been received and then process the buffered message. All messages should be processed in sequence number order.



3.6.5 Sequence Reset (4)

Sequence Reset (35=4) allows the client or the gateway to increase the expected incoming sequence number of the other party, for example to skip heartbeats on a response to a Resend Request.

In a Gap Fill it is sent as notification of the next sequence number to be transmitted.

Tag	Field Name	Req	Data Type	Description
123	GapFillFlag	N	Boolean	Indicates that the Sequence Reset message is replacing administrative or application messages which will not be resent. Valid values: Y = Gap Fill message, MsgSeqNum (34) field valid. N = Sequence Reset, ignore MsgSeqNum (tag 34). If omitted default value is N.
36	NewSeqNo	Y	SeqNum (9)	Sequence number of the next message to be transmitted.

3.6.6 Logout (5)

Logout (35=5) initiates or confirms the termination of a FIX session. FIX clients should terminate their sessions gracefully by logging out.

If a FIX user has their password reset by LME Market Operations and attempts to login with their previous password, the user will receive a Logout with SessionStatus (1409) = Password change is required.

Tag	Field Name	Req	Data Type	Description
1409	SessionStatus	N	Int	Session status at time of logout. Valid values: 3 = New session password does not comply with policy 4 = Session logout complete 5 = Invalid username or password 6 = Account locked 7 = Logons are not allowed at this time 8 = Password expired 100 = Password change is required 101 = Other
58	Text	C	String (50)	Reason for logout. Conditionally required if SessionStatus (1409) = ‘101’ Other

3.6.7 Reject (3)

Reject (35=3) will be sent when a message is received but cannot be properly processed by the gateway due to a session level rule violation. For example, a message missing a mandatory tag.

Tag	Field Name	Req	Data Type	Description
45	RefSeqNum	Y	SeqNum (9)	Sequence number of the message which caused the rejection.
371	RefTagID	N	Int	If a message is rejected due to an issue with a particular field its tag number will be indicated.
372	RefMsgType	N	String (2)	Message type of the rejected message.
373	SessionRejectReason	N	Int	Code specifying the reason for the session level rejection: Valid values: 0 = Invalid Tag Number 1 = Required Tag Missing 2 = Tag not defined for this message 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 9 = CompID problem 10 = Sending Time Accuracy problem 11 = Invalid Msg Type 13 = Tag appears more than once 15 = Repeating group fields out of order 16 = Incorrect NumInGroup count for repeating group 18 = Invalid/Unsupported Application Version 99 = Other.
58	Text	C*	String (50)	Conditionally required if SessionRejectReason (373) = ’99’ Other. Text specifying the reason for the rejection.

3.7 Other Messages

3.7.1 Business Message Reject (j)

Once an application level message passes validation at FIX Session level it will then be validated at business level. If business level validation detects an error condition then a rejection should be issued.

For example, an application message is received and passes session level validation but the message type is not supported by the system, a Business Message Reject will be returned with a BusinessRejectReason (380) = ‘3’ Unsupported Message Type.

Tag	Field Name	Req	Data Type	Description
45	RefSeqNum	N	SeqNum (9)	Sequence number of the message which caused the rejection.
372	RefMsgType	Y	String (2)	Message type of the rejected message.
379	BusinessRejectRefID	N	String (21)	Client specified unique identifier on the message that was rejected.
380	BusinessRejectReason	Y	Int	Code specifying the reason for the rejection of the message. Valid values: 0 = Other 3 = Unsupported Message Type
58	Text	C*	String (50)	Conditionally required if BusinessRejectReason (380) = ‘0’ Other. Text specifying the reason for the rejection.

3.8 Parties Component Block

Tag	Field Name	Req	Data Type	Description
453	NoPartyIDs	N	NumInGrp (2)	Number of parties specified.
>448	PartyID	Y	String See PartyRole Usage	Party identifier/code. Required if NoPartyIDs (453) > 0.

Tag	Field Name	Req	Data Type	Description

>447	PartyIDSource	Y	Char	Source of the PartyID (448) value. Required if NoPartyIDs (453) > 0.
				
				Valid values:
				P = Client Short Code
				D = Proprietary/Custom
				E = ISO Country Code (i.e. two letter ISO country code)
				N = Legal Entity ID - LEI

>452	PartyRole	Y	Int	Role of the specified PartyID (448). Required if NoPartyIDs (453) > 0. Valid values: 1 = Executing Firm 3 = Client ID 4 = Clearing Firm 7 = Entering Firm 11 = Order Origination Trader 24 = Customer Account 26 = Correspondent broker 36 = Entering Trader 66 = Market Maker 81 = Broker Client ID 122 = Decision Maker 300 = Investment Decision Within Firm 301 = Execution Decision Within Firm 302 = Investment Decision Country 303 = Execution Decision Country 304 = Client Branch Country

3.8.1 PartyRole Usage

PartyRole (452) values used by LME are described below:

PartyRole (452)		PartyID (448) format	PartyIDSource (447)	Description	Usage
1	Executing Firm	Char (3)	D = Proprietary/Custom	Identifier of the executing firm.	N/A for order entry Used for Transaction Reporting and Order Record Keeping
3	Client ID	Int (8 bytes)	P = Client Short Code	Client short code identifier. Required only for Client Orders i.e. AccountType (581) = 1, 8 or 101 where OrderAttributeType (2594) = 0 or 1 has not been specified. Note: PartyID (448) can be set to 0 = No Client for if there is no client where AccountType (581) = 3. PartyID (448) is not valid if populated with either 1, 2 or 3.	Conditional - Mandatory for Client orders Used for Transaction Reporting and Order Record Keeping Up to two instances of PartyRole (452) = '3' Client ID can be specified but PartyIDSource (447) values must be unique.
		String (<=40)	D = Proprietary/Custom	Proprietary or Custom Client ID as assigned by the member. Required only for client orders i.e. AccountType (581) = 1, 8 or 101.	
		String (<=40)	N = Legal Entity ID	LEI.	Optional

PartyRole (452)		PartyID (448) format	PartyIDSource (447)	Description	Usage
					Up to two instances of PartyRole (452) = '3' Client ID can be specified but PartyIDSource (447) values must be unique.
4	Clearing Firm	Char (3)	D = Proprietary/Custom	Identifier of the clearing firm. A 3 character broker code (Member mnemonic). Cannot be entered in requests but is returned in Execution Reports for all fills	N/A for order entry Used for Transaction Reporting and Order Record Keeping
7	Entering Firm	Char (3)	D = Proprietary/Custom	Identifier of the entering firm. A 3 character broker code (Member mnemonic). Required for MiFID as agent relationships are not captured in the LME participant structure.	Optional Used for Transaction Reporting and Order Record Keeping
11	Order Origination Trader	String (<=40)	D = Proprietary/Custom	Order Origination Trader (associated with Order Origination Firm e.g. trader who initiates/submits the order). (Required as could be more than one individual under a FIX Comp ID). Required in New Order Single and will be returned in Execution Reports.	Mandatory for House and Client orders

PartyRole (452)		PartyID (448) format	PartyIDSource (447)	Description	Usage
24	Customer Account	String (<=30)	D = Proprietary/Custom	Identification of the Client Account Code where the AccountType (581) = 1, 8 or 101.	Conditional - Mandatory for Client orders
26	Correspondent broker - Non-executing broker	Char (3)	D = Proprietary/Custom	A 3 character broker code (Member mnemonic).	Optional Used for Order Record Keeping
36	Entering Trader	String (<=10)	D = Proprietary/Custom	Identifier of the trader entering the order. Cannot be entered in requests but will be returned in Execution Reports.	N/A for order entry
66	Market Maker	Char (1)	D = Proprietary/Custom	This should be set to Y if the trader qualifies for a Market Maker initiative. Not validated by the system but should be set correctly on New Single Order requests.	Optional
81	Broker Client ID	String (<=16)	D = Proprietary/Custom	Identifier of the entity in a risk group. Required in New Order Single and will be returned in Execution Reports	Mandatory used for Risk Management
122	Decision Maker	Int (8 bytes)	P = Client Short Code	Decision maker short code, required on client orders to identify the investment decision maker. Also used under the power of representation clause where the investment decision maker may be a third party in	Conditional - Mandatory for Client orders Transaction Reporting

PartyRole (452)		PartyID (448) format	PartyIDSource (447)	Description	Usage
				accordance with Article 8 of Commission Delegated Regulation (EU) …/… 22 on transaction reporting under Article 26 of Regulation EU No 600/2014. Required only for client orders i.e. AccountType (581) = 1, 8 or 101.	
300	Investment Decision Within Firm	Int (8 bytes)	P = Client Short Code	Short code to identify the individual who is responsible for the investment decision.	Optional Used for Transaction Reporting and Order Record Keeping
301	Execution Decision Within Firm	Int (8 bytes)	P = Client Short Code	Short code to identify the execution decision maker with the firm. Required in New Order Single and Order Cancel Replace Requests and will be returned in Execution Reports.	Mandatory for House and Client orders Used for Transaction Reporting and Order Record Keeping
302	Investment Decision Country	Char (2)	E = ISO Country Code	ISO Country Code of the branch responsible for the person making the investment decision.	Optional Used for Transaction Reporting
303	Execution Decision Country	Char (2)	E = ISO Country Code	ISO Country Code of the branch responsible for the person making the execution decision.	Optional

PartyRole (452)		PartyID (448) format	PartyIDSource (447)	Description	Usage
					Used for Transaction Reporting
304	Client Branch Country	Char (2)	E = ISO Country Code	ISO Country Code to identify the branch that received the client order or made an investment decision for a client. Required for client orders i.e. AccountType (581) = 1, 8 or 101	Conditional - Mandatory for Client orders Used for Transaction Reporting

3.9 Application Messages

3.9.1 Execution Report (8)

Execution Report is used to:

- confirm the receipt of an order submitted using New Order Single or *Mass Quote*
- confirm changes to an existing order (i.e. accept cancel and replace requests)
- confirm or convey an order cancellation or expiration
- convey order or trade cancellation by Market Operations
- convey fill information
- convey triggering of a stop order
- *convey speed bump processing*
- convey information about restated persisted orders carried from one trading day to the next
- notify a GCM of trades executed by their NCMs.

ExecType (150) identifies the purpose of the execution report message and OrdStatus (39) conveys the current state of the order.

The Execution Report sent to a Drop Copy user will have the CopyMsgIndicator (797) set to Y = Yes to indicate that the message is a drop copy of another message.

The attributes that can be returned in an Execution Report for each execution type are listed in the Execution Report Matrix. Refer to 4.11.7.1 Execution Report Matrix in the Order Entry Gateway FIX Specification.

Tag	Field Name	Req	Data Type	Description
37	OrderID	Y	String (19)	A unique order identifier set by the trading system. This identifier is not changed by cancel/replace messages; it will remain the same for all chain of orders.
526	*SecondaryClOrdID*	*C**	*String (18)*	*Quote Entry ID in a Mass Quote (22).* *Conditionally required according to Execution Report Matrix.*
11	ClOrdID	Y*	String (18)	Client specified identifier in the message that caused this Execution Report. *For quotes this is mapped to Quote ID in a Mass Quote (22)*
41	OrigClOrdID	C	String (18)	ClOrdID (11) of the previous order (NOT the initial order of the day) as assigned by the

Tag	Field Name	Req	Data Type	Description
				institution. Identifies the previous order in cancel and cancel/replace requests. Conditionally required according to the Execution Report Matrix. *Not applicable for an order from a Mass Quote (22).*
Component Block <Parties>		Y*	See Parties Component Block	
880	TrdMatchID	C*	String (19)	Identifier assigned by the trading system which joins buy and sell half trades. Conditionally required if ExecType (150) = ‘F’ Trade.
17	ExecID	Y	String (19)	Unique identifier assigned by the trading system to the execution message. A copy from Order Entry.
19	ExecRefID	C*	String (19)	Reference identifier used with Trade Cancel execution type. Conditionally required if ExecType (150) = ‘H’ Trade Cancel. *Not applicable for an order from a Mass Quote (22).*
150	ExecType	Y	Char	Describes the specific Execution Report. Valid values: 0 = New 3 = Done 4 = Cancelled 5 = Replaced C = Expired D = Restated F = Trade H = Trade Cancel L = Triggered or Activated by the System

Tag	Field Name	Req	Data Type	Description
39	OrdStatus	Y	Char	Identifies current status of order. Valid values: 0 = New 1 = Partially Filled 2 = Filled 3 = Done for day 4 = Cancelled C = Expired
378	ExecRestatementReason	C*	Int	Conditionally required if ExecType (150) = 'D' Restated. The reason for restatement. Valid values: 1 = GT renewal / restatement 99 = Other. See ExecTypeReason (2431) for speed bump handling. *Not applicable for an order from a Mass Quote (22).*
581	AccountType	Y*	Int	Specifies the type of account associated with the order. Valid values: 1 = Client ISA 3 = House 8 = Joint back office account (JBO) = Gross OSA 101 = Client OSA For contracts assigned to the T4 booking model only 3 = House is valid whereas for the T2 booking model all account types are valid.
1115	OrderCategory	C*	Char	Conditionally required for a trade from an implied order when ExecType (150) = 'F' Trade. Defines the type of interest behind a trade (fill or partial fill). Valid value: 7 = Implied Order

Tag	Field Name	Req	Data Type	Description
Component Block <Instrument>				
48	SecurityID	Y*	Int	Tradable instrument identifier
22	SecurityIDSource	C*	String (1)	Identifies the source of the SecurityID (48): 8 = Exchange Symbol Conditionally required when SecurityID (48) is specified.
End Component Block				
54	Side	Y	Char	Side of the order Valid values: 1 = Buy 2 = Sell
Component Block <OrderQtyData>				
38	OrderQty	Y*	Qty	Total order quantity of the order.
End Component Block				
40	OrdType	Y*	Char	Order type applicable to the order. Valid values: *1 = Market* 2 = Limit *3 = Stop Market* 4 = Stop Limit *11 = Iceberg* *12 = Post-Only* *13 = One Cancels Other Market* *14 = One Cancels Other Limit*
44	Price	C	Price (20)	The order price. Conditionally required if OrdType (40) is 2 = Limit or 4 = Stop Limit.
99	StopPx	C*	Price (20)	The Stop trigger price. Conditionally required if OrdType

Tag	Field Name	Req	Data Type	Description
				(40) = *'3' Stop Market* or '4' Stop Limit. TriggerPriceType (1107) is required if StopPx is specified. *Not applicable for an order from a Mass Quote (22).*
Component Block <TriggeringInstruction>				
1100	TriggerType	C*	Char	Trigger prompt for the stop order elements. Conditionally required if any other Triggering tags are specified. Valid value: 4 = Price Movement *Not applicable for an order from a Mass Quote (22).*
1102	*TriggerPrice*	*C**	*Price (20)*	*Stop order price of the OCO.* *Conditionally required for an OCO.* *Not applicable for an order from a Mass Quote (22).*
1107	TriggerPriceType	C*	Char	Type of price event that triggers the stop order: Valid values: 2 = Last Trade 4 = Best Bid or Last Trade 5 = Best Offer or Last Trade Conditionally required if StopPx (99) or *TriggerPrice (1102)* is specified. *Not applicable for an order from a Mass Quote (22).*
1110	*TriggerNewPrice*	*C**	*Price (20)*	*Limit order price of the stop once triggered.*

Tag	Field Name	Req	Data Type	Description
				Conditionally required if TriggerOrderType (1111) = '2' Limit *Not applicable for an order from a Mass Quote (22).*
1111	*TriggerOrderType*	*C**	*Char*	*Order type of the order once triggered.* *1 = Market* *2 = Limit* *Conditionally required for an OCO.* *Not applicable for an order from a Mass Quote (22).*
End Component Block				
59	TimeInForce	Y*	Char	Specifies how long the order remains in effect. Valid values: 0 = Day 1 = Good Till cancel (GTC) 3 = Immediate or cancel (IOC) *4 = Fill or Kill* 6 = Good Till Date (GTD)
432	ExpireDate	C	LocalMktDate	The expiry date of an order. Conditionally required if TimeInForce (59) = ‘6’ Good Till Date is not specified. Format is YYYYMMDD. *Not applicable for an order from a Mass Quote (22).*
18	ExecInst	C*	MultipleCharValue	Instructions for order handling. If more than one instruction is applicable to an order, this field can contain multiple instructions separated by space. Valid values:

Tag	Field Name	Req	Data Type	Description
				6 = Participate but don't initiate for Post Only orders o = Cancel on connection loss Conditionally required according to Execution Report Matrix. *Not applicable for an order from a Mass Quote (22).*
1057	AggressorIndicator	C*	Boolean	Indicates if a matching order is an aggressor or not in the trade. Y = Aggressor N = Passive Conditionally required if ExecType (150) = 'F' Trade. *Not applicable for an order from a Mass Quote (22).*
528	OrderCapacity	Y*	Char	Designates the capacity of the firm placing the order. Valid values: A (agency) = AOTC P (principal) = DEAL R (riskless principal) = MTCH

529	OrderRestrictions	Y*	MultipleCharValue	Indicates if the order is entered either by an algo trader or a human.
				Valid values:
				D = Non-algorithmic (human)
				E = Algorithmic (algo)

32	LastQty	C	Qty	Conditionally required if ExecType (150) = 'F' Trade. The total volume of this trade.
31	LastPx	C	Price (20)	Conditionally required if ExecType (150) = 'F' Trade. The price of this trade.
151	LeavesQty	Y	Qty	The quantity open for further execution.

Tag	Field Name	Req	Data Type	Description
				If OrdStatus (39) = '4' Cancelled or'C' Expired then LeavesQty (151) could be 0 otherwise LeavesQty (151) will be OrderQty38) - CumQty (14)
14	CumQty	Y	Qty	The quantity of the order that has been executed so far.
60	TransactTime	Y*	UTCTimestamp	Timestamp when the message was generated.
Component Block <DisplayInstruction>				
1138	*DisplayQty*	*C**	*Qty*	*Visible quantity for Iceberg orders.* *Conditionally required for an Iceberg order.* *Not applicable for an order from a Mass Quote (22).*
End Component Block				
58	Text	C*	String (50)	Contains the value supplied in this field on the order. Conditionally required according to Execution Report Matrix. *Not applicable for an order from a Mass Quote (22).*
Component Block <InstrmtLegExecGrp>				
555	NoLegs	C*	NumInGrp (1)	Conditionally required if ExecType (150) = 'F' Trade on a multileg tradable instrument. Number of InstrumentLeg repeating group instances.
Component Block <InstrumentLeg> - Required if NoLegs (555) > 0				
>602	LegSecurityID	C*	Int	Conditionally required if ExecType (150) = F (Trade).

Tag	Field Name	Req	Data Type	Description
				Multileg tradable instrument's individual SecurityID.
>603	LegSecurityIDSource	C*	String (1)	Identifies the source of the SecurityID (48): 8 = Exchange Symbol Conditionally required when LegSecurityID (602) is specified.
>624	LegSide	C*	Char	Conditionally required if ExecType (150) = ‘F’ Trade on a multileg tradable instrument. The side of this individual leg (multileg security). Valid values: 1 = Buy 2 = Sell
End Component Block				
>1366	LegAllocID	C*	String (19)	Strategy leg trade identifier assigned by the trading system which is shared by half trades. Conditionally required if ExecType (150) = ‘F’ Trade on a multileg tradable instrument.
>637	LegLastPx	C*	Price (20)	Conditionally required if ExecType (150) = ‘F’ Trade on a multileg tradable instrument Execution price assigned to the leg of the multileg tradable instrument.
>1418	LegLastQty	C*	Qty	Conditionally required if ExecType (150) = ‘F’ Trade on a multileg tradable instrument. Fill quantity for the instrument leg.
End Component Block				

Tag	Field Name	Req	Data Type	Description
797	CopyMsgIndicator	Y	Boolean	Indicates whether or not this message is a drop copy of another message.
1328	RejectText	C*	String (75)	Identifies the reason for rejection. Conditionally required for unsolicited cancellations
1724	OrderOrigination	C*	Int	Origin of the order Valid value: 5 = Order received from a direct access or sponsored access (the trader has direct electronic access – DEA) Absence of this field indicates DEA = false. Conditionally required according to Execution Report Matrix.
2431	ExecTypeReason	C*	Int	The initiating event for the Execution Report. Conditionally required to report unsolicited cancellation and order status in speed bump processing. Valid values: 4 = Unsolicited order cancellation *101 = Order accepted but speed bump applied* *102 = Order added after speed bump* *103 = Order cancelled whilst in speed bump delay* *104 = Original order is in speed bump enforced delay* *105 = Order updated after speed bump delay* *106 = Amend is in speed bump delay* *107 = Order amended after speed bump delay*

Tag	Field Name	Req	Data Type	Description
				108 = Order rejected after speed bump delay *109 = Unsolicited cancel while in speed bump*
2362	*SelfMatchPreventionID*	*C**	*Int (9)*	*Identifies an order that should not be matched to an opposite order if both buy and sell orders for the trade contain the same SelfMatchPreventionID (2362) and are submitted by the same member.* *Conditionally required according to Execution Report Matrix.*
Component Block <OrderAttributeGrp> - Conditionally required if specified.				
2593	NoOrderAttributes	C*	NumInGrp (1)	Number of order attribute entries.
>2594	OrderAttributeType	C*	Int	The type of order attribute. Conditionally required if NoOrderAttributes (2593) > 0. Valid values: 0 = Aggregated order. In the context of ESMA RTS 24 Article 2(3), when OrderAttributeValue (2595) = Y, it signifies that the order consists of several orders aggregated together. This maps to ESMA RTS value "AGGR". 1 = Pending allocation. In the context of ESMA RTS 24 Article 2(2), when OrderAttributeValue (2595) = Y, it signifies that the order submitter "is authorized under the legislation of a Member State to allocate an order to its client following submission of the order to the trading venue and has not yet allocated the order to its client at the time of the submission of the order". This

Tag	Field Name	Req	Data Type	Description
				maps to ESMA RTS value "PNAL". 2 = Liquidity Provision Order. In the context of ESMA RTS 24 Article 3, when OrderAttributeValue (2595) = Y, it signifies that the order was submitted "as part of a market making strategy pursuant to Articles 17 and 18 of Directive 2014/65/EU, or is submitted as part of another activity in accordance with Article 3" (of RTS 24). 3 = Risk Reduction Order. In the context of ESMA RTS 22 Article 4(2)(i), when OrderAttributeValue (2595) = Y, it signifies that the commodity derivative order is a transaction "to reduce risk in an objectively measurable way in accordance with Article 57 of Directive 2014/65/EU".
>2595	OrderAttributeValue	C*	String	The value associated with the order attribute type specified in OrderAttributeType (2594). Conditionally required if NoOrderAttributes (2593) > 0. Valid value: Y = Yes
End Component Block				
1819	RelatedHighPrice	C*	Price	Upper price limit value
1820	RelatedLowPrice	C*	Price	Lower price limit value