FINANCIAL INFORMATION EXCHANGE (FIX)

FIX APPLICATION LAYER

Business Area: Infrastructure

FIX Latest

As of EP97, August 2019

FIX Global Technical Committee

Table of Contents

DISCLAIMER

THE INFORMATION CONTAINED HEREIN AND THE FINANCIAL INFORMATION EXCHANGE PROTOCOL (COLLECTIVELY, THE "FIX PROTOCOL") ARE PROVIDED "AS IS" AND NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL MAKES ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, AS TO THE FIX PROTOCOL (OR THE RESULTS TO BE OBTAINED BY THE USE THEREOF) OR ANY OTHER MATTER AND EACH SUCH PERSON AND ENTITY SPECIFICALLY DISCLAIMS ANY WARRANTY OF ORIGINALITY, ACCURACY, COMPLETENESS, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SUCH PERSONS AND ENTITIES DO NOT WARRANT THAT THE FIX PROTOCOL WILL CONFORM TO ANY DESCRIPTION THEREOF OR BE FREE OF ERRORS. THE ENTIRE RISK OF ANY USE OF THE FIX PROTOCOL IS ASSUMED BY THE USER.

NO PERSON OR ENTITY ASSOCIATED WITH THE FIX PROTOCOL SHALL HAVE ANY LIABILITY FOR DAMAGES OF ANY KIND ARISING IN ANY MANNER OUT OF OR IN CONNECTION WITH ANY USER'S USE OF (OR ANY INABILITY TO USE) THE FIX PROTOCOL, WHETHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL (INCLUDING, WITHOUT LIMITATION, LOSS OF DATA, LOSS OF USE, CLAIMS OF THIRD PARTIES OR LOST PROFITS OR REVENUES OR OTHER ECONOMIC LOSS), WHETHER IN TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), CONTRACT OR OTHERWISE, WHETHER OR NOT ANY SUCH PERSON OR ENTITY HAS BEEN ADVISED OF, OR OTHERWISE MIGHT HAVE ANTICIPATED THE POSSIBILITY OF, SUCH DAMAGES.

No proprietary or ownership interest of any kind is granted with respect to the FIX Protocol (or any rights therein), except as expressly set out in FIX Protocol Limited's Copyright and Acceptable Use Policy.

REPRODUCTION

FIX Protocol Limited grants permission to print in hard copy form or reproduce the FIX Protocol specification in its entirety provided that the duplicated pages retain the "Copyright FIX Protocol Limited" statement at the bottom of the page.

Portions of the FIX Protocol specification may be extracted or cited in other documents (such as a document which describes one's implementation of the FIX Protocol) provided that one reference the origin of the FIX Protocol specification (http://www.fixtrading.org) and that the specification itself is "Copyright FIX Protocol Limited".

FIX Protocol Limited claims no intellectual property over one's implementation (programming code) of an application which implements the behavior and details from the FIX Protocol specification.

1 Area – Infrastructure

Infrastructure messaging is characterized as messages which are common to the business areas pre-trade, trade and post-trade.

The specific FIX infrastructure messaging categories are:

1. CATEGORY: BUSINESS MESSAGE REJECTS
2. CATEGORY: NETWORK STATUS COMMUNICATION
3. CATEGORY: USER MANAGEMENT
4. CATEGORY: APPLICATION SEQUENCING
5. APPENDIX: COMPONENTS AND MESSAGES

Descriptions of the specific FIX infrastructure application messages follow.

1.1 Business Message Rejects

The BusinessMessageReject(35=j) message can reject an application-level message which fulfills session-level rules and cannot be rejected via any other means. Note if the message fails a session-level rule (e.g. body length is incorrect), a session-level Reject(35=3) message should be issued.

The only exception to this rule is when a transport other than the FIX session protocol (FIX4 or FIXT) is being used (transport independence). An appropriate reject message of the given session protocol (e.g. Reject(35=3) for FIX4 or FIXT session layer) or the BusinessMessageReject(35=j) message should be used instead.

It should **NOT** be used in the following situations:

Situation	***Appropriate Response***
Session-level problem meeting the criteria of the session-level Reject(35=3) message	Use the session-level Reject(35=3) message if the FIX session protocol FIX 4 or FIXT is being used. If FIX4 or FIXT are not being used, use an appropriate reject message of the given session protocol or the BusinessMessageReject(35=j) message.
In response to: – QuoteRequest(35=R)	Use the QuoteRequestReject(35=AG) message
In response to: – Quote(35=S) – QuoteCancel(35=Z) – QuoteStatusRequest(35=a) – QuoteResponse(35=AJ)	Use the QuoteStatusReport(35=AI) message
In response to: – MassQuote(35=i)	Use the MassQuoteAck(35=b) message
In response to: – MarketDataRequest(35=V)	Use the MarketDataRequestReject(35=Y) message
In response to:	Use the StreamAssignmentReport(35=CD) message

Situation	*Appropriate Response*
– StreamAssignmentRequest(35=CC)	
In response to: – StreamAssignmentReport(35=CD)	Use the StreamAssignmentReportACK(35=CE) message
In response to: – SecurityDefinitionRequest(35=c)	Use the SecurityDefinition(35=d) message
In response to: – SecurityTypeRequest(35=v)	Use the SecurityTypes(35=w) message
In response to: – SecurityListRequest(35=x)	Use the SecurityList(35=y) message
In response to: – DerivativeSecurityListRequest(35=z)	Use the DerivativeSecurityList(35=AA) message
In response to: – SecurityStatusRequest(35=e)	Use the SecurityStatus(35=f) message
In response to: – TradingSessionStatusRequest(35=g)	Use the TradingSessionStatus(35=h) message
In response to: – TradingSessionListRequest(35=BI)	Use the TradingSessionList(35=BJ) message
In response to: – PartyDetailsListRequest(35=CF)	Use the PartyDetailsListReport(35=CG) message
In response to: – NewOrderSingle(35=D) – OrderStatusRequest(35=H) – OrderMassStatusRequest(35=AF) – NewOrderCross(35=s) – NewOrderMultileg(35=AB) – NewOrderList(35=E) – ListExecute(35=L)	Use the ExecutionReport(35=8) message
In response to:	Use the OrderCancelReject(35=9) message

Situation	***Appropriate Response***
– OrderCancelRequest(35=F) – OrderCancelReplaceRequest(35=G) – CrossOrderCancelRequest(35=u) – CrossOrderCancelReplaceRequest(35=t) – MultilegOrderCancelReplace(35=AC) – ListCancelRequest(35=K)	
In response to: – ExecutionReport(35=8)	Use the DontKnowTrade(35=Q) message (DK'd) or the ExecutionReportAck(35=BN) message
In response to: – OrderMassCancelRequest(35=q)	Use the OrderMassCancelReport(35=r) message
In response to: – OrderMassActionRequest(35=CA)	Use the OrderMassActionReport(35=BZ) message
In response to: – ListStatusRequest(35=M)	Use the ListStatus(35=N) message
In response to: – BidRequest(35=k)	Use the BidResponse(35=l) message
In response to: – AllocationInstruction(35=J)	Use the AllocationInstructionAck(35=P) message
In response to: – AllocationReport(35=AS)	Use the AllocationReportAck(35=AT) message
In response to: – Confirmation(35=AK)	Use the ConfirmationAck(35=AU) message
In response to: – RegistrationInstructions(35=o)	Use the RegistrationInstructionsResponse(35=p) message
In response to: – TradeCaptureReportRequest(35=AD)	Use the TradeCaptureReportRequestAck(35=AQ) message
In response to: – TradeCaptureReport(35=AE)	Use the TradeCaptureReportAck(35=AR) message

Situation	*Appropriate Response*
In response to: – ConfirmationRequest(35=BH)	Use the Confirmation(35=AK) message
In response to: – SettlementInstructionRequest(35=AV)	Use the SettlementInstructions(35=T) message
In response to: – PositionMaintenanceRequest(35=AL)	Use the PositionMaintenanceReport(35=AM) message
In response to: – RequestForPositions(35=AN)	Use the RequestForPositionsAck(35=AO) message
In response to: – CollateralRequest(35=AX)	Use the CollateralAssignment(35=AY) message
In response to: – CollateralAssignment(35=AY)	Use the CollateralResponse(35=AZ) message
In response to: – CollateralInquiry(35=BB)	Use the CollateralInquiryAck(35=BG) message

Note the only exceptions to this rule are:

1. **in the event a business message is received, fulfills session-level rules, however, the message cannot be communicated to the business-level processing system.** In this situation a BusinessMessageReject(35=j) with BusinessRejectReason(380) = 4 (Application not available) can be issued if the system is unable to send the specific "reject" message listed above due to this condition.

2. **in the event a valid business message is received, fulfills session-level rules, however, the message type is not supported by the recipient.** In this situation a BusinessMessageReject(35=j) with BusinessRejectReason(380) = 3 (Unsupported Message Type) can be issued if the system is unable to send the specific "reject" message listed above because the receiving system cannot generate the related "reject" message.

3. **In the event a business message is received, fulfills session-level rules, but lacks a field conditionally required by the FIX specification.** In this situation a BusinessMessageReject(35=j) with BusinessRejectReason(380) = 5 (Conditionally required field missing) can be issued if the system is unable to send the specific "reject" message listed above. One example of this would be a stop order missing StopPx(99). However, a BusinessMessageReject(35=j) message **MUST NOT** be used to enforce proprietary rules more restrictive than those explicit in the FIX specification, such as a broker requiring an order to contain an Account(1), which the FIX specification considers an optional field.

1.1.1 Key Fields for Application Message References

Messages which can be referenced via the BusinessMessageReject(35=j) message are (the "ID" field BusinessRejectRefID(379) refers to is noted in [ ]) as follows:

1.1.1.1 Pre-Trade

- Indication of Interest - IOI(35=6) [IOIid(23)]
- Advertisement(35=7) [AdvId(2)]

- News(35=B) [Headline(148) or NewsID(1472)]
- Email(35=C) [EmailThreadID(164)]
- MarketDataSnapshotFullRefresh(35=W) [MDReqID(262)]
- MarketDataIncrementalRefresh(35=X) [MDReqID(262)]
- MarketDataRequestReject [MDReqID(262)]
- MarketDefinition(35=BU) [MarketReportID(1394)]
- MarketDefinitionRequest(35=BT) [MarketReqID(1393)]
- MarketDefinitionUpdateReport(35=BV) [MarketReportID(1394)]
- StreamAssignmentReportAck(35=CE) [StreamAsgnRptID(1501)]
- SecurityDefinition(35=d) [SecurityResponseID(322) or SecurityReportID(964)]
- SecurityDefinitionUpdateReport(35=BP) [SecurityResponseID(322) or SecurityReportID(964)]
- SecurityStatus(35=f) [SecurityStatusReqID(324)]
- SecurityTypes(35=w) [SecurityResponseID(322)]
- SecurityList(35=y) [SecurityResponseID(322)]
- SecurityListUpdateReport(35=BK) [SecurityResponseID(322) or SecurityReportID(964)]
- DerivativeSecurityList(35=AA) [SecurityResponseID(322)]
- DerivativeSecurityListUpdateReport(35=BR) [SecurityResponseID(322)]
- TradingSessionStatus(35=h) [TradSesReqID(335)]
- TradingSessionList(35=BJ) [TradSesReqID(335)]
- TradingSessionListUpdateReport(35=BS) [TradSesReqID(335)]
- PartyDetailsListReport(35=CG) [PartyDetailsListReportID(1510)]
- MassQuoteAck(35=b) [QuoteReqID(131) or QuoteID(117)]
- QuoteRequestReject(35=R) [QuoteReqID(131)]
- RFQRequest(35=AH) [RFQReqID(644)]
- QuoteStatusReport(35=AI) [QuoteStatusReqID(649) or QuoteRespID(693) or QuoteID(117) or QuoteMsgID(1166)]
- OrderCancelReject(35=9) [ClOrdID(11)]

1.1.1.2 Trade

- ListStatus(35=N) [ListID(66)]
- ListStrikePrice(35=m) [ListID(66)]
- BidResponse(35=l) [BidID(390)]
- OrderMassCancelReport(35=r) [ClOrdID(11)]
- OrderMassActionReport(35=BZ) [MassActionReportID(1369) or ClOrdID(11)]
- OrderMassStatusRequest(35=AF) [MassStatusReqID(584)
- DontKnowTrade(35=Q) – may respond with OrderCancelReject(35=9) if attempting to cancel order [ExecID(17)]
- ExecutionAck(35=BN) [ExecID(17)]

1.1.1.3 Post-Trade

- AllocationInstructionAck(35=P) [AllocID(70)]
- AllocationReportAck(35=AT) [AllocID(70)]

- AllocationInstructionAlert(35=BM) [AllocID(70)]
- ConfirmationAck(35=AU) [ConfirmID(664)]
- TradeCaptureReport(35=AE) [TradeReportID(571)]
- TradeCaptureReportRequestAck(35=AQ) [TradeRequestID(568)]
- TradeCaptureReportAck(35=AR) [TradeReportID(571)]
- PositionMaintenanceReport(35=AM) [PosMaintRptID(721)]
- RequestForPositionsAck(35=AO) [PosMaintRptID(721)]
- AdjustedPositionReport(35=BL) [PosMaintRptID(721)]
- PositionReport(35=AP) [PosMaintRptID(721)]
- AssignmentReport(35=AW) [AsgnRptID(833)]
- ContraryIntentionReport(35=BO) [ContIntRptID(977)]
- SettlementInstructions(35=T) [SettInstMsgID(777)]
- SettlementObligationReport(35=BQ) [SettlObligMsgID(1160)]
- RegistrationInstructionsResponse(35=p) [RegistID(513)]
- CollateralResponse(35=AZ) [CollRespID(904)]
- CollateralInquiryAck(35=BG) [CollInquiryID(909)]
- CollateralReport(35=BA) [CollRptID(908)]

1.1.2 Scenarios for BusinessMessageReject(35=j):

BusinessRejectReason(380)
0 = Other
1 = Unkown ID
2 = Unknown Security
3 = Unsupported Message Type (receive a valid, but unsupported MsgType)
4 = Application not available
5 = Conditionally Required Field Missing
6 = Not Authorised
7 = DeliverTo firm not available at this time
18 = Invalid price increment

Whenever possible, it is strongly recommended that the cause of the failure be described in Text(58) (e.g. "UNKNOWN SYBMOL: XYZ").

1.2 Network Status Communication

It is envisaged these messages will be used in two scenarios:

Scenario A

Allow one counterparty using a "hub and spoke" FIX network to know whether another counterparty is currently connected to the hub (i.e. whether the counterparty's session to the hub is up or not).

Scenario B

Allow a counterparty connecting to a global brokerage to know which regions within that brokerage are currently available as order routing destinations.

1.2.1 Network (Counterparty System) Status Requests

The NetworkCounterpartySystemStatusRequest(35=BC) message is sent either immediately after logging on to inform a network (counterparty system) of the type of updates required or to, at any other time in the FIX conversation, to change the nature of the types of status updates required. It can also be used with NetworkRequestType(935) = 1 (Snapshot) to request a one-off report of the status of a network (or counterparty) system. Finally this message can also be used to cancel a request to receive updates into the status of the counterparties on a network by sending a NetworkCounterpartySystemRequestStatusMessage(35=BC) message with NetworkRequestType = 4 (Stop Subscribing).

1.2.2 Network (Counterparty System) Status Responses

The NetworkCounterpartySystemStatusResponse(35=BD) message is sent in response to a NetworkCounterpartySystemStatusRequest(35=BC) message with a list of counterparties and their status in the repeating group CompIDStatGrp.

If the network response payload is larger than the maximum permitted message size for that FIX conversation the response would be several NetworkCounterpartySystemStatusResponse(35=BD) messages, the first with a status of full and then as many messages, as updates to the first message, adding information as required.

1.3 User Management

These messages are provided in FIX to allow the passing of individual user information between two counterparties. The messages allow for the following functions by means of UserRequestType(924):

1 – Individual User Logon

2 – Individual User Logout

3 – Individual User password change

4 – Individual User Status Enquiries

NOTE: It is not encouraged to transmit passwords in a FIX conversation unless you can guarantee the end to end security of both the FIX conversation and any intermediate routing hubs that are involved in the routing.

1.3.1 User Requests

The UserRequest(35=BE) message is used to initiate a user action, e.g. logon, logout or password change. It can also be used to request a report on a user's status.

1.3.2 User Responses

The UserResponse(35=BF) message is used to respond to a UserRequest(35=BE) message, it reports the status of the user after the completion of any action requested in the UserRequest(35=BE) message.

1.3.3 User Notifications

The UserNotification(35=CB) message is used to notify one or more users of an event or information from the sender of the message. This message is usually sent unsolicited from a marketplace (e.g. Exchange, ECN) to a market participant.

1.4 Application Sequencing

1.4.1 Introduction

FIX has a growing need to support application-level sequencing of messages in order to segregate the transmission of data over a session. The ability to identify and retransmit a subset of data by application and application sequence number range is an important feature in support of secondary distribution of data (see definition below). The current retransmission capabilities of the FIX session require that all messages on that

session between the specified starting and ending message sequence number are resent rather than just those that have been produced by a specific upstream business process or application. This can pose capacity and performance problems for systems that need only a small set of messages related to an application. Secondary data distribution consists of a diverse set of data sourced from different applications; drop copy data, credit limit information, metrics, etc. It is **not** recommended that application sequencing is used over a conventional order routing or a transaction flow oriented connection. Standard FIX session capabilities should be used in this case.

Application sequencing greatly enhances the usefulness of FIX messages that are transmissted apart from the FIX session layer by making it possible for the receiver to detect and request missed messages on a specified feed. Market data sent over a broadcast or multicast transport is often in need of sequencing and retransmission. Application sequencing provides a means by which to sequence each message that is part of a broadcast stream such that the receiver can verify ordered delivery of the data. Application resends can then be requested when gaps are detected in the application sequence.

1.4.2 Background

The purpose of application-level sequencing is to allow messages being sent over a FIX session to be distinguished by the sending application that is upstream from the FIX engine. In the case that a session-level resend would result in an unnecessarily large number of messages being resent, application sequencing and recovery makes provision for the desired messages - and only the desired messages - to be seamlessly requested and resent while retaining the standard behaviors of the session protocol. It also provides the receiver with the flexibility to put off recovery of application level messages until a slow period or after the market has closed.

1.4.2.1 Extends control over resent data

The primary intent of application sequencing and recovery is to allow receivers to avoid a retransmission of large quantities of unusable data which may result in receivers needing to glean the retransmission for the data they actually need - such as critical drop copy information that is used in risk management applications. Application sequencing allows the channeling of different types of data across a single FIX session. For example, application sequencing can allow drop copy data to be sent over the same FIX session with order flow data. While this may not be practical from a trading standpoint the flexibility that it introduces is compelling. This allows data which has a higher importance and priority to be identified by application ID thereby allowing requests for retransmission to be issued promptly and precisely.

1.4.2.2 Support for secondary data distribution

Another goal of the proposal is to provide support for "secondary data" distribution. Application sequencing extends the capabilities of FIX such that secondary data can be distributed using a single channel. This data may be less time critical with less demanding latency requirements than order entry and market data, although this is not necessarily the case as drop copies are used for time sensitive risk management tasks. Secondary data may consist of drop copy fills, credit limit information, statistical data, trade confirmations, and best bids and offers for vendor consumption, etc. These are just a few of the possibilities. Application sequencing benefits data providers and their users by providing a common protocol which can be used to perform secondary data distribution. New applications transmitting data can be quickly introduced over an existing channel with minimal effort simply by introducing a new ApplID(1180) (application ID).

1.4.3 Transaction usage is not recommended

Application sequencing is not something that will be used in a normal order routing scenario. It has more relevance in large volume one-way connections in which the receiver would like to have some ability to control the data that is resent after a disconnect or data loss. There is no obvious advantage in using application sequencing with a regular trading connection since all data transmitted between sender and receiver is of equal importance in maintaining a viable trading session. Application sequencing should not be used to track broker connections that are in place for trading purposes. It should only be used for managing the flow of data when a FIX connection is used to deliver data in bulk and where there is a stated need to create classes of data.

1.4.4 Using Application Sequencing and Session Sequencing for Gap Detection

The use of ApplResendFlag(1352) on the ApplicationSequenceGroup component (available on all FIX messages representing reports) is used to indicate that messages are being retransmitted as a result of an ApplicationMessageRequest(35=BW) message. When using the FIX session protocol FIX4 or FIXT, it is possible for both ApplResendFlag(1352) and PossDupFlag(43) to be set on the same message if the sender's

cache size is greater than zero and the message is being resent due to a session level ResendRequest(35=2) message.

The sender and receiver may agree to use a limited cache in order to benefit from the convenience of session-level retransmission. In this case, a message that is dropped in response to an ApplicationMessageRequest(35=BW) message may have both fields present. This scenario depends on whether (1) the sender is maintaining a cache and (2) the sender and receiver have agreed to fill any gaps to the extent possible using the session level.

In this scenario, a combination of application and session-level sequencing will be used to recover missed messages. A limited cache of session-level messages may be retained by the sender in order to recover messages that have been dropped within an pre-stated window defined by time or number of messages. When a FIX session ResendRequest(35=2) message is issued within this window the sender's session will resend the messages. Once the window has been exceeded an ApplicationMessageRequest(35=BW) must be issued in order to recover dropped messages. The application level will not be aware that a gap has occurred until the session level has recovered what is available. Beyond this, the application will detect the gap according to the logic as described and issue a resend request at the application level using the ApplicationMessageRequest(35=BW) message.

Gap detection and recovery with respect to the ApplicationMessageRequest(35=BW) message and response messages (e.g. ApplicationMessageRequestAck(35=BX) and resent application messages using the ApplicationSequenceGrp component) may also need to take place at the application level since session level recovery may have been suspended.

1.4.5 Application Message Requests

The ApplicationMessageRequest(35=BW) message is used to request a retransmission of a set of one or more messages generated by the application specified in RefApplID(1355). The message can be used for various types of transmission requests:

- ApplReqType(1347) = 0 – retransmission of application messages for a specified application and sequence number range,
- ApplReqType(1347) = 1 – subscription to an application in order receive, for example, drop copy services,
- ApplReqType(1347) = 2 – request for the last application sequence number sent by an application,
- ApplReqType(1347) = 3 – request the valid set of application identifiers for which a user is authorized,
- ApplReqType(1347) = 4 – unsubscribe to one or more applications
- ApplReqType(1347) = 5 – cancel retransmission
- ApplReqType(1347) = 6 – cancel retransmission and unsubscribe

The Request message specifies the sequence number range using ApplBegSeqNum(1182) and ApplEndSeqNum(1183) for a given RefApplID(1355) to request messages for retransmission.

1.4.6 Application Message Request Acknowledgements

The ApplicationMessageRequestAck(35=BX) message is used to acknowledge an ApplicationMessageRequest(35=BW) message providing a status on the request (i.e. whether successful or not) with ApplResponseType(1348). This message does not provide the actual content of the messages to be resent.

1.4.7 Application Message Reports

The ApplicationMessageReport(35=BY) message is used for different purposesas indicated by ApplReportType(1426):

- ApplReportType(1426) = 0 – reset ApplSeqNum(1181) of a specified ApplID(1180)
- ApplReportType(1426) = 1 – indicate that the last message has been sent for a particular ApplID(1180)
- ApplReportType(1426) = 2 – keep-alive mechanism for ApplID(1180) values with infrequent message traffic

- ApplReportType(1426) = 3 – re-send of application messages completed

1.4.7.1 Using Application Message Reports to reset application-level sequence number

The ApplicationMessageReport(35=BY) message with ApplReportType(1426) = 0 (Reset) is sent by the sender of an application to alert the receiver that the application sequence number of application RefApplID(1355) is being reset to ApplNewSeqNum(1399), for one or more RefApplID(1355) values, to the specified value(s). The next application message received will then conform to this value. In other words, ApplSeqNum(1181) in this message represents the next expected application sequence number the receiver will receive from the sender for the corresponding ApplID(1180). An ApplicationMessageReport(35=BY) message with ApplReportType(1426) = 0 (Reset) has no affect on, and is independent of, the FIX session sequence number in MsgSeqNum(34).

1.4.7.2 Using Application Message Reports to indicate last message sent

The ApplicationMessageReport(35=BY) message with ApplReportType(1426) = 1 (Last message) is sent by the sender of an application to indicate that the last message has been sent for one or more RefApplID(1355) values. Reception of this message mean the recipient can safely assume that no more message will be sent for that/or those RefApplID(1355) values. RefApplLastSeqNum(1357) should be set to ApplSeqNum(1181) on the last application-level message. RefApplID(1355) is set to ApplID(1180) on this message.

1.4.7.3 Using Application Message Report as keep-alive mechanism

For recipients of applications with infrequent message traffic it is a problem to detect a gap in the message flow. The gap cannot be detected until reception of the next message for that ApplID(1180). To mitigate this problem the ApplicationMessageReport(35=BY) message can be issued by the sender of an application at regular intervals. RefApplLastSeqNum(1357) should be set to the last ApplSeqNum(1181) sent for this application, identified by ApplID(1180) and referenced on the report by RefApplID(1355).

1.4.7.4 Using Application Message Report to indicate completion of resent messages

As part of a recovery scenario, the receiver (or consumer) may request all of the messages for one or more applications. Because of the potentially lengthy re-send situation, the request can be acknowledged with an ApplicationMessageRequestAck(35=BX) prior to beginning the re-send of messages. In this case, the receiver or consumer will begin seeing re-sent messages until the re-send is complete. However, once the re-send is complete, the receiver or consumer will only know that the re-send has completed when they receive a new copied message from that application identified with ApplID(1180) that no longer has ApplResendFlag(1352) = Y. If the specified ApplID(1180) is only “heartbeating” and there are no new messages to send, the consumer will still not know the application message re-send has actually finished. It is in this case that an ApplicationMessageReport(35=BY) can be generated, which signals completion by setting ApplReportType(1426) = 3 (application message re-send completed).

2 Appendix

2.1 Application Category

2.1.1 Components

2.1.1.1 ApplIDReportGrp

Number of applications

Tag	*Name*	*Req'd*	*Description*
1351	NoApplIDs		
→1355	RefApplID	N	
→1399	ApplNewSeqNum	N	
→1357	RefApplLastSeqNum	N	

2.1.1.2 ApplIDRequestAckGrp

Number of applications

Tag	*Name*	*Req'd*	*Description*
1351	NoApplIDs		
→1355	RefApplID	N	
→1433	RefApplReqID	N	
→1182	ApplBegSeqNum	N	
→1183	ApplEndSeqNum	N	
→1357	RefApplLastSeqNum	N	
→1354	ApplResponseError	N	
→Component	**NestedParties**	N	

2.1.1.3 ApplIDRequestGrp

Specifies number of application id occurrences

Tag	*Name*	*Req'd*	*Description*
1351	NoApplIDs		
→1355	RefApplID	N	

Tag	*Name*	*Req'd*	*Description*
→1433	RefApplReqID	N	
→1182	ApplBegSeqNum	N	Message sequence number of first message in range to be resent
→1183	ApplEndSeqNum	N	Message sequence number of last message in range to be resent. If request is for a single message ApplBeginSeqNo = ApplEndSeqNo. If request is for all messages subsequent to a particular message, ApplEndSeqNo = "0" (representing infinity).
→**Component**	**NestedParties**	N	

2.1.2 Messages

2.1.2.1 ApplicationMessageRequest Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = BW
1346	ApplReqID	Y	Unique identifier for request
1347	ApplReqType	Y	Type of Application Message Request being made
Component	**ApplIDRequestGrp**	N	
Component	**Parties**	N	
58	Text	N	Allows user to provide reason for request
354	EncodedTextLen	N	
355	EncodedText	N	
Component	**StandardTrailer**	Y	

2.1.2.2 ApplicationMessageRequestAck Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = BX
1353	ApplResponseID	Y	Identifier for the Application Message Request Ack
1346	ApplReqID	N	Identifier of the request associated with this ACK message
1347	ApplReqType	N	
1348	ApplResponseType	N	
1349	ApplTotalMessageCount	N	Total number of messages included in transmission

Tag	*Name*	*Req'd*	*Description*
Component	**ApplIDRequestAckGrp**	N	
Component	**Parties**	N	
58	Text	N	
354	EncodedTextLen	N	
355	EncodedText	N	
Component	**StandardTrailer**	Y	

2.1.2.3 ApplicationMessageReport Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = BY
1356	ApplReportID	Y	Identifier for the Application Message Report
1346	ApplReqID	N	If the application message report is generated in response to an ApplicationMessageRequest(MsgType=BW), then this tag contain the ApplReqID(1346) of that request.
1426	ApplReportType	Y	Type of report
Component	**ApplIDReportGrp**	N	
58	Text	N	
354	EncodedTextLen	N	
355	EncodedText	N	
Component	**StandardTrailer**	Y	

2.2 BusinessReject Category

2.2.1 Messages

2.2.1.1 BusinessMessageReject Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = j (lowercase)
45	RefSeqNum	N	MsgSeqNum of rejected message
372	RefMsgType	Y	The MsgType of the FIX message being referenced.

Tag	*Name*	*Req'd*	*Description*
1130	RefApplVerID	N	Recommended when rejecting an application message that does not explicitly provide ApplVerID (1128) on the message being rejected. In this case the value from the DefaultApplVerID(1137) or the default value specified in the NoMsgTypes repeating group on the logon message should be provided.
1406	RefApplExtID	N	Recommended when rejecting an application message that does not explicitly provide ApplExtID(1156) on the rejected message. In this case the value from the DefaultApplExtID(1407) or the default value specified in the NoMsgTypes repeating group on the logon message should be provided.
1131	RefCstmApplVerID	N	Recommended when rejecting an application message that does not explicitly provide CstmApplVerID(1129) on the message being rejected. In this case the value from the DefaultCstmApplVerID(1408) or the default value specified in the NoMsgTypes repeating group on the logon message should be provided.
379	BusinessRejectRefID	N	The value of the business-level "ID" field on the message being referenced. Required unless the corresponding ID field (see list above) was not specified.
380	BusinessRejectReason	Y	Code to identify reason for a Business Message Reject message.
58	Text	N	Where possible, message to explain reason for rejection
354	EncodedTextLen	N	Must be set if EncodedText field is specified and must immediately precede it.
355	EncodedText	N	Encoded (non-ASCII characters) representation of the Text field in the encoded format specified via the MessageEncoding field.
Component	**StandardTrailer**	Y	

2.3 Network Category

2.3.1 Components

2.3.1.1 CompIDReqGrp

Used to restrict updates/request to a list of specific CompID/SubID/LocationID/DeskID combinations.

If not present request applies to all applicable available counterparties. EG Unless one sell side broker was a customer of another you would not expect to see information about other brokers, similarly one fund manager etc.

Tag	*Name*	*Req'd*	*Description*
936	NoCompIDs		

Tag	*Name*	*Req'd*	*Description*
→930	RefCompID	N	Used to restrict updates/request to specific CompID
→931	RefSubID	N	Used to restrict updates/request to specific SubID
→283	LocationID	N	Used to restrict updates/request to specific LocationID
→284	DeskID	N	Used to restrict updates/request to specific DeskID

2.3.1.2 CompIDStatGrp

Specifies the number of repeating CompId's

Tag	*Name*	*Req'd*	*Description*
936	NoCompIDs		
→930	RefCompID	Y	CompID that status is being report for. Required if NoCompIDs > 0,
→931	RefSubID	N	SubID that status is being report for.
→283	LocationID	N	LocationID that status is being report for.
→284	DeskID	N	DeskID that status is being report for.
→928	StatusValue	Y	
→929	StatusText	N	Additional Information, i.e. “National Holiday”

2.3.2 Messages

2.3.2.1 NetworkCounterpartySystemStatusRequest Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = “BC”
935	NetworkRequestType	Y	
933	NetworkRequestID	Y	
Component	**CompIDReqGrp**	N	Used to restrict updates/request to a list of specific CompID/SubID/LocationID/DeskID combinations. If not present request applies to all applicable available counterparties. EG Unless one sell side broker was a customer of another you would not expect to see information about other brokers, similarly one fund manager etc.
Component	**StandardTrailer**	Y	

2.3.2.2 NetworkCounterpartySystemStatusResponse Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = “BD”
937	NetworkStatusResponseType	Y	
933	NetworkRequestID	N	
932	NetworkResponseID	Y	
934	LastNetworkResponseID	N	Required when NetworkStatusResponseType=2
Component	**CompIDStatGrp**	Y	Specifies the number of repeating CompId's
Component	**StandardTrailer**	Y	

2.4 UserManagement Category

2.4.1 Components

2.4.1.1 UsernameGrp

Number of usernames

Tag	*Name*	*Req'd*	*Description*
809	NoUsernames		
→553	Username	N	Recipient of the notification

2.4.2 Messages

2.4.2.1 UserRequest Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = “BE”
923	UserRequestID	Y	
924	UserRequestType	Y	
553	Username	Y	
554	Password	N	
925	NewPassword	N	
1400	EncryptedPasswordMethod	N	
1401	EncryptedPasswordLen	N	

Tag	*Name*	*Req'd*	*Description*
1402	EncryptedPassword	N	
1403	EncryptedNewPasswordLen	N	
1404	EncryptedNewPassword	N	
95	RawDataLength	N	
96	RawData	N	Can be used to hand structures etc to other API's etc
Component	**StandardTrailer**	Y	

2.4.2.2 UserResponse Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = “BF”
923	UserRequestID	Y	
553	Username	Y	
926	UserStatus	N	
927	UserStatusText	N	Reason a request was not carried out
Component	**StandardTrailer**	Y	

2.4.2.3 UserNotification Message

Tag	*Name*	*Req'd*	*Description*
Component	**StandardHeader**	Y	MsgType = CB
Component	**UsernameGrp**	N	List of users to which the notification is directed
926	UserStatus	Y	Reason for notification - when possible provide an explanation.
58	Text	N	Explanation for user notification.
354	EncodedTextLen	N	Must be set if EncodedText field is specified and must immediately precede it.
355	EncodedText	N	Encoded (non-ASCII characters) representation of the Text field in the encoded format specified via the MessageEncoding field.
Component	**StandardTrailer**	Y	

