INTERNATIONAL STANDARD

ISO 3531-2

First edition
2022-04

Financial services — Financial information eXchange session layer —

Part 2:
FIX session layer

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Reference number
ISO 3531-2:2022(E)

© ISO 2022

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

COPYRIGHT PROTECTED DOCUMENT

© ISO 2022

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Table of Contents

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by FIX Trading Community (as FIX Session Layer Technical Specification) and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 68, *Financial services*, Subcommittee SC 9, *Information exchange for financial services* and adopted under the “fast-track procedure”.

A list of all parts in the ISO 3531 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

Introduction

FIX session protocol was written to be independent of any specific communications protocol (e.g. X.25, async, TCP/IP) or physical medium (e.g. copper, fibre, satellite) chosen for electronic data delivery. It offers a reliable stream where a message is delivered once and in order. The FIX session layer is designed to survive and resume operation in the event of the loss of transport level connections caused by any type of failure, including network outage, application failure or computer hardware failures.

The session layer is concerned with the ordered delivery of data while the application level defines business-related data content. This document focuses on the ordered delivery of data using the "FIX session protocol".

The FIX session protocol is implemented using the FIX tagvalue encoding syntax for the standard header, standard trailer and the session level messages which make up the FIX session protocol. It is possible to send messages using other encodings, such as the other FIX-defined encodings (e.g. FIXML, SBE, JSON, GPB, ASN.1) or non-FIX-defined encodings (e.g. XML, FpML, ISO 20022 XML, JSON).

The Financial Information eXchange session layer is used to provide reliable and recoverable messaging for electronic trading. The protocol is intended for use by asset managers, trading firms, brokerages, trading venues, clearing houses, custodians, depositories, asset servicers and others involved in the trading life cycle activities of a wide range of financial instruments. The FIX session layer functionality is a realization of the ISO/IEC 7498-1 Open System Interconnection basic reference model level 5 session layer.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Financial services — Financial information eXchange session layer —

Part 2: FIX session layer

1 Scope

This document provides the normative specification of the FIX session layer standard and its session profiles.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

3.1
session layer message
message carried over the FIX session that is integral to the operation of the FIX session

3.2
application message
message that is carried over a FIX session to accomplish some business purpose

Note 1 to entry: Examples of a business purpose include an order to buy or sell a financial instrument, reporting market data or reporting an execution of a trade.

3.3
message type
identifier (code) that defines the type of message being sent

Note 1 to entry: The message type for the FIX session is a case-sensitive string encoded in the MsgType(35) field.

3.4
valid FIX message
session or application message that is a tagvalue encoded string of octets that is properly formed according to the FIX tagvalue encoding specification

3.5
FIX session processor
combination of computer hardware, firmware and software that implements the FIX session layer

Note 1 to entry: Commonly referred to as a FIX Engine.

3.6
initiator
FIX session processor that establishes the transport layer connection and initiates the session via transmission of the initial Logon(35=A) message

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

3.7
acceptor
FIX session processor that is able to establish a transport layer connection and receive Logon(35=A) requests from FIX session initiators to start or resume a FIX session

3.8
rules of engagement
specification, usually provided in document form, that describes a specific use of FIX

Note 1 to entry: Often referred to as a FIX service offering. FIX Orchestra is a standard that may be used to specify machine-readable rules of engagement.

3.9
peer
FIX session processor being communicated with over a FIX session

Note 1 to entry: The peer of the initiator is the acceptor. The peer of the acceptor is the initiator. The initiator and acceptor are peers.

3.10
counterparty
firm, legal entity or individual agreeing to use the FIX session layer to conduct some form of business endeavour

3.11
NextNumIn
Each FIX session processor must keep track of the next message sequence number it is expecting to receive from its peer to guarantee ordered delivery of messages over the life of a FIX session that may span multiple FIX connections. The next expected incoming sequence number (NextNumIn) is compared to the value in the MsgSeqNum(34) field in each message received from the peer.

3.12
NextNumOut
Each FIX session processor must keep track of the next outbound sequence number (NextNumOut) it will send to its peer over a FIX connection.

3.13
retransmission
resending of a message that was previously sent in order to resynchronize the FIX session

Note 1 to entry: A retransmitted message uses the original MsgSeqNum(34) value with PossDupFlag(43)=Y.

3.14
resend
application message being resent by an application layer because it has not received an application layer acknowledgement for the message

Note 1 to entry: A resend may only be initiated by the application layer, not the session layer. The determination of whether the message was previously received is the responsibility of the application layer, not the session layer.

3.15
gap fill
process to resolve gaps in message sequence numbers within a FIX session

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

3.16
application version
The FIX session layer provides fields to communicate versions of the application layer messages. The FIX Trading Community defines standard application versions.

3.17
extension pack
approved addition to the FIX standard

Note 1 to entry: The granularity of an extension pack may vary widely from a single enumeration value addition to the definition of entire new categories of messages. Extension packs are identified by a sequential integer number and must be applied in order. An extension pack is considered available for use if it has been approved and published by the FIX Global Technical Committee. Extension packs are created on an as-needed basis and are generally driven by community requests. An extension pack is cumulative, in that the artefacts (Orchestra file, FIXimate) include all previous extension packs. When an extension pack is published, it becomes the *FIX Latest* version of FIX.

3.18
session profile
extension or restriction on the use of the standard session layer messages that can be used to represent context of usage

Note 1 to entry: FIX4, FIXT, and LFIXT are the current FIX session profiles.

3.19
CompID
alphanumeric identifier for the entity associated with a FIX session

Note 1 to entry: As this is likely a financial markets participant company, the name was viewed colloquially as a company identifier abbreviated as a *CompID*.

3.20
SubID
subidentifier optionally available to identify a subentity within a CompID

Note 1 to entry: The SubID may be used to identify a specific trader or a subunit of a business entity. The use of SubIDs is at the discretion of the counterparties. Certain regulatory regimes globally require the use of SubID to identify specific traders.

3.21
LocationID
location identifier providing additional location information either geographical or within a trading desk on a trading floor

Note 1 to entry: The use of LocationIDs is at the discretion of the counterparties.

3.22
transport layer connection
A FIX session relies on a transport layer to provide for ordered delivery of messages and message recovery during the life of the transport layer connection. The FIX session does not require a specific transport layer, although TCP/IP is widely used and is a de facto standard transport layer for FIX sessions. TCP/IP provides an ordered reliable delivery of a stream of bytes during the life of the TCP connection. The FIX session processor reads this stream identifying FIX message boundaries.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

3.23
in-band communication
transmission of control information or metadata about the application layer or session layer in application and session FIX message types over the FIX session

Note 1 to entry: Sending the HeartBtInt(108) field in the Logon(35=A) message is an example of in-band transmission of control information. Providing version information about a FIX service in the CstmApplVerID(1129) on the Logon(35=A) message is another example.

3.24
out-of-band communication
exchange of control information or metadata about a FIX session via a separate communication mechanism than the FIX session

Note 1 to entry: Providing the TestRequestThreshold in a rules of engagement document is an example of exchanging information out-of-band. Providing version information about a FIX service on a website or in a rules of engagement document is another example of out-of-band communication. Providing access to a specification online via a website or web service would be another example of out-of-band communication. Even though this is electronic communication, it is not being done over the FIX session to which that information applies.

3.25
TestRequestThreshold
amount of time expressed as a multiplier of the heart beat interval before a TestRequest(35=1) message is sent to the peer when the heartbeat interval has been exceeded without receiving a message from the peer

Note 1 to entry: This value may be specified out-of-band in a rules of engagement.

3.26
SendingTimeThreshold
amount of time expressed in seconds in which the SendingTime(52) value in an inbound message differs from the system time available to the receiving FIX session processor

Note 1 to entry: This value may be specified out-of-band in a rules of engagement.

3.27
LogoutAckThreshold
amount of time expressed in seconds that a FIX session processor that has transmitted a Logout(35=5) request will wait for the Logout(35=5) acknowledgement before terminating the transport layer connection

Note 1 to entry: This value may be specified out-of-band in a rules of engagement.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4 FIX session

4.1 General

A **FIX session** is a bidirectional stream of ordered messages between two peers within a continuous sequence number series beginning with 1. A single FIX session can exist across multiple sequential (not concurrent) FIX connections, which means that peers may intentionally or unintentionally connect and disconnect multiple times while maintaining a single FIX session. The FIX session can be thought of as a bidirectional durable session sharing characteristics of the guaranteed delivery and durable subscriber enterprise integration architecture patterns.

© ISO 2022 – All rights reserved

A **FIX connection** consists of three parts: logon process, message exchange (inclusive of resynchronization of state) and possible logout process over a transport layer connection. A FIX connection may be concluded by the unrecoverable loss of the transport layer, a system failure or an application failure.

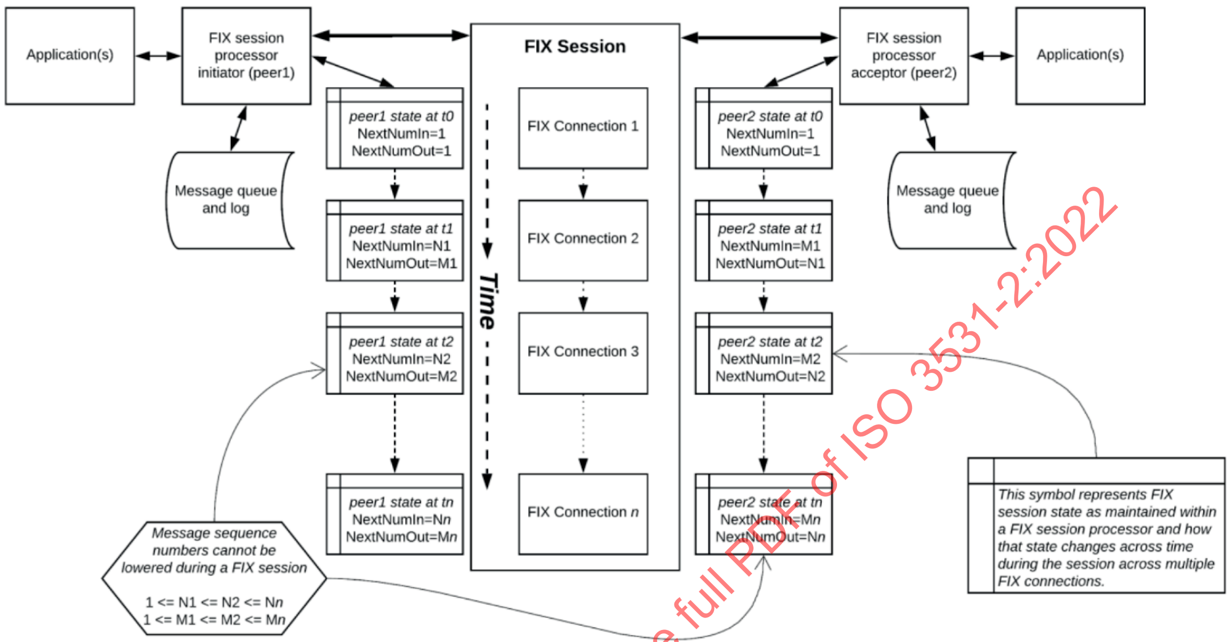

Application(s)
FIX session processor initiator (peer1)
Message queue and log
FIX Session
Time
FIX Connection 1
FIX Connection 2
FIX Connection 3
FIX Connection *n*
peer1 state at t0 NextNumIn=1 NextNumOut=1
peer1 state at t1 NextNumIn=N1 NextNumOut=M1
peer1 state at t2 NextNumIn=N2 NextNumOut=M2
peer1 state at tn NextNumIn=N*n* NextNumOut=M*n*
peer2 state at t0 NextNumIn=1 NextNumOut=1
peer2 state at t1 NextNumIn=M1 NextNumOut=N1
peer2 state at t2 NextNumIn=M2 NextNumOut=N2
peer2 state at tn NextNumIn=M*n* NextNumOut=N*n*
FIX session processor acceptor (peer2)
Application(s)
Message queue and log
Message sequence numbers cannot be lowered during a FIX session
1 <= N1 <= N2 <= N*n*
1 <= M1 <= M2 <= M*n*
This symbol represents FIX session state as maintained within a FIX session processor and how that state changes across time during the session across multiple FIX connections.

Figure 1 — Conceptual view of FIX session layer

Connecting parties shall bilaterally agree upon the time the FIX session is started and the duration of a FIX session.[1] A FIX session is often configured to correspond to a certain period of time, such as a trading day, calendar day or a trading session. A FIX session may extend beyond multiple periods by counterparty agreement.

The FIX session is based on an optimistic model. Normal delivery of data are assumed (i.e. no session layer acknowledgement of individual messages) with errors in delivery identified by message sequence number gaps.

4.2 Sequence numbers

All messages sent over a FIX session shall be identified by a unique sequence number in the MsgSeqNum(34) field.

A FIX session shall start with a next expected outgoing sequence number (NextNumOut) of 1 and a next expected incoming sequence number (NextNumIn) of 1.

A FIX session processor must maintain the NextNumOut and NextNumIn for the entire FIX session.

A FIX session processor shall persist the NextNumOut and NextNumIn in order to support FIX session recovery across multiple FIX connections.

Resetting NextNumOut to 1 and NextNumIn to 1, for whatever reason, shall constitute the beginning of a new FIX session.

FIX session layer and application layer messages shall share the same sequence number space.

[1] Firms may document the session layer configuration using the FIX Orchestra interface specification.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Each FIX session layer and application layer message sent consumes the next outbound sequence number, incrementing NextNumOut by 1.

Each FIX session layer and application layer message received consumes the next inbound sequence number, incrementing NextNumIn by 1.

A FIX session must always have peer1.NextNumIn < = peer2.NextNumOut and peer2.NextNumIn < = peer1.NextNumOut to be considered in a valid and recoverable state.

4.3 Identifying the FIX session

4.3.1 General

A FIX session is identified by the unique combination of BeginString(8) + initiator CompID + acceptor CompID.

4.3.2 The FIX session profile

BeginString(8) shall be used to identify the FIX session profile or version of FIX.

These FIX session profiles are defined within this specification.

Table 1 — The FIX session profiles

FIX session profile	BeginString(8)	Description
FIX.4.2	FIX.4.2	The FIX session profile for use with the FIX.4.2 application layer.
FIX4	FIX.4.4	The FIX session profile backward compatible with FIX.4.4 recommended when counterparties will only be using a single application version during the FIX session, such as *FIX Latest*.
FIXT	FIXT.1.1	The FIX session profile that must be used when mixing multiple application versions over the same FIX session. May be used with a single application version of FIX such as *FIX Latest*.
LFIXT	FIXT.1.1	Lightweight FIXT restricted session layer message recovery[2] to simplify the protocol while maintaining compatibility with FIXT when using LFIXT compatible model of operation.

FIX session acceptor may be configured to support multiple FIX session profiles (or *FIX versions* prior to FIX.4.4) over the same transport layer with only one FIX session profile (or *FIX version* prior to FIX.4.4) being used per FIX session.

The FIX session acceptor should use the incoming Logon(35 = A) request BeginString(8) to identify the FIX session profile or FIX version being requested by the initiator.

4.3.3 Identification of FIX session peers

FIX relies on alphanumeric strings known as CompIDs to identify the initiator and the acceptor of FIX messages.

Counterparties must agree to their respective CompID values, which act as an identifier for the peer.

Each message sent over a FIX session layer must contain the sending entity in SenderCompID(49) and the receiving (destination) entity in TargetCompID(56). This requirement applies to both session layer and application layer messages.

[2] The LFIXT session profile uses the same BeginString(8) value as the FIXT session profile. The use of LFIXT and its mode of operation must be agreed upon out-of-band by counterparty agreement.

STANDARDSISO.COM: Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

FIX service processors must validate the SenderCompID(49) and TargetCompID(56) of each message and ensure that it is identical to the values present in the FIX connection Logon(35 = A) message. A discrepancy in the SenderCompID(49)+TargetCompID(56) pair should result in the termination of the FIX connection by sending a Logout(35 = A) message using the Text(58) field to indicate the error, followed by termination of the transport layer connection.

4.3.4 Validation of SendingTime(52)

SendingTime(52) must be set to the time the message is transmitted by the FIX session processor, not the time the message was queued for transmission.

SendingTime(52) must be in UTC (Universal Time Coordinated).

SendingTime(52) should be within the reasonable time of a synchronized time source of the receiving FIX session processor. This *SendingTimeThreshold* is highly dependent upon the type of application using the FIX session layer. The *SendingTimeThreshold* may be as low as seconds for high volume, low latency applications up to several minutes for certain order routing applications.

The *SendingTimeThreshold* may be defined and communicated to counterparties out-of-band within the rules of engagement.

The receiving peer shall transmit a Reject(35 = 3) message with SessionRejectReason(373) set to 10 (SendingTime Accuracy Problem) followed by a Logout(35 = 5) request when the SendingTime(52) is not within a specified tolerance of a synchronized clock.

4.3.5 Additional fields available for peer identification

The FIX session layer provides additional fields which may be used to further identify recipients within the respective counterparties.

Each counterparty may provide a subidentifier (SenderSubID(50) and TargetSubID(57)) known as SubID.

Each counterparty may provide a location identifier (SenderLocationID(142) and TargetLocationID(143)) known as LocationID.

The SubID and LocationID may vary across messages across the same FIX session.

The counterparties must agree upon the context and semantics of the SubID and LocationID if used.

4.4 Establishing a FIX connection

Establishing a FIX connection involves three distinct operations: creation of a transport layer connection, acceptance with optional authentication of the initiator by the acceptor and message synchronization (initialization).

The initiator shall establish a transport layer connection with the acceptor to establish a FIX connection.

4.4.1 Transport layer requirements

FIX session layer requires that the transport layer provides ordered and lossless message delivery and full duplex operation for the life of the transport layer connection.

FIX implementations that use the TCP/IP protocol as the transport layer must use the FIX-over-TLS (FIXS) specification which specifies the use of Transport Layer Security (TLS) with the FIX session layer to provide transport layer encryption.

The initiator shall send a Logon(35 = A) request message to initiate a FIX connection.

The Logon(35 = A) request message must always be the first message transmitted over a FIX connection. If the acceptor receives anything other than a valid Logon(35 = A) request message, an error should be logged and the transport layer connection terminated without Logout(35 = 5) processing.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.4.2 Using the TestMessageIndicator(464) to explicitly identify testing

Counterparties should populate the TestMessageIndicator(464) field with "Y" in the Logon(35=A) request and acknowledgement when testing.

FIX counterparties should validate that the TestMessageIndicator(464) field corresponds to the environment.

If the TestMessageIndicator(464) with value "Y" is present in a Logon(35 = A) message indicating a test session and the environment is not a test environment, the peer should transmit a Logout(35 = 5) message with the Text(58) indicating that the TestMessageIndicator(464) was set to "Y" and the environment is a production environment. Certain venues support sending test messages within the production environment. Counterparties should communicate how testing is conducted within the production environment to prevent test messages from being accepted as production messages.

If the TestMessageIndicator(464) with value "N" is present in a Logon(35 = A) message indicating a production session and the environment is a test environment, the peer should transmit a Logout(35 = 5) message with the Text(58) indicating that the TestMessageIndicator(464) was set to "N" and the environment is a test environment.

Failure to validate the TestMessageIndicator(464) against the environment may potentially lead to financial losses.

4.4.3 Application layer encryption

The use of application layer message encryption has been discouraged since the FIX.4.3 version of the FIX protocol.

EncyptionMethod(98) must be present in the Logon(35 = A) message and set to "0" when application layer encryption is not in use.

Firms may implement application layer encryption by counterparty agreement.

4.4.4 Heartbeat interval

The HeartBtInt(108) field must be present and specify the interval in seconds for generating heartbeat messages during periods of inactivity.

The HeartBtInt(108) field must be present in the Logon(35 = A) message.

The HeartBtInt(108) value must be agreed upon by the initiator and the acceptor.

The initiator and acceptor must use the same HeartBtInt(108) value within a FIX connection.

The HeartBtInt(108) value is the maximum time allowed between receipt of messages from the peer.

Each FIX session processor should implement a heartbeat interval timer. The timer should be reset upon receipt of each message from the peer.

4.4.5 Heartbeat interval determination

The acceptor should convey the rules placed on the expected heartbeat interval via out-of-band rules of engagement[3] when such rules are required by the acceptor.

There are three methods for determining the heartbeat interval:

1. Acceptor requires a specific heartbeat interval

2. Acceptor requires initiator to specify a value within a heartbeat interval range

[3] FIX Orchestra may be used to communicate which FIX session profile and which operating mode is being used for a FIX enabled service.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

3. Acceptor accepts the initiator specified heartbeat interval

4.4.5.1 Acceptor requires a specific heartbeat interval

The acceptor may refuse to establish a FIX connection if the HeartBtInt(108) value received on the Logon(35=A) message sent by the initiator does not match the heartbeat interval required by the acceptor.

The acceptor must terminate the FIX session using a Logout(35 = 5) message when the acceptor refuses to establish a FIX connection due to the inbound HeartBtInt(108) value not matching the expected value.

The acceptor should provide the reason for terminating the FIX connection in the Text(58) field of the Logout(35 = 5) message by setting it to “Invalid HeartBtInt(108), expected value N seconds”, where N is the number of seconds expected in the HeartBtInt(108) field on the initiator Logon(35 = A) message. N shall be larger than zero.

4.4.5.2 Acceptor requires initiator specify a value within a heartbeat interval range

The acceptor may refuse to establish a FIX connection if the HeartBtInt(108) value received on the Logon(35=A) message sent by the initiator does not fall within a heartbeat interval range required by the acceptor.

The acceptor must terminate the FIX connection using a Logout(35 = 5) message when the acceptor refuses to establish a FIX session due to the inbound HeartBtInt(108) value not falling within the required heartbeat interval range.

The acceptor should specify the reason for terminating the connection in the Text(58) field of the Logout(35 = 5) message by setting it to “Invalid HeartBtInt(108), expected value between N and M seconds”, where N and M are the start and end values of the range of the heartbeat interval in seconds permitted by the acceptor. N shall be larger than zero and M shall be larger than N.

4.4.5.3 Acceptor accepts the initiator specified heartbeat interval

The acceptor may accept the heartbeat interval specified by the initiator in the HeartBtInt(108) field of the initiator Logon(35=A) message by echoing back the heartbeat interval of the initiator in the Logon(35=0) acknowledgement.

4.4.6 Maximum message size

The initiator and acceptor may specify the maximum message size supported in the MaxMessageSize(383) field in the Logon(35=A) message.

Either peer may terminate a FIX connection due to inability to process the maximum message size specified by the other peer by sending a Logout(35 = 5) message.

The peer terminating the FIX connection should specify the reason in the Text(58) field of the Logout(35 = 5) message by setting it to “MaxMessageSize(383)=*InboundValue* exceeds maximum message size of *maximum message size*”.

4.4.7 Specifying application version

Three fields are provided to identify the application version to counterparties in the Logon(35=A) message. The use of these fields varies across FIX session profiles. For instance, these fields are not applicable to the FIX.4.2 session profile as the FIX.4.2 session profile may only be used with the FIX 4.2 application version.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Field	Definition
DefaultApplVerID(1137)	The FIX application version upon which the application layer is derived.
DegaultApplExtID(1407)	An extension pack that is used with the application layer.
DefaultCstmApplVerID(1408)	User defined custom application version information. DefaultCstmAppVerID(1400) may be used to identify the FIX Orchestra machine readable rules of engagement.

4.4.8 Specifying supported message types

The initiator and acceptor may specify the types of FIX messages by using the repeating group MsgTypeGrp in the Logon(35=A) message.

4.4.9 Identification of application system and FIX session processor[4]

By counterparty agreement, peers may identify the application system as well as the FIX session processor ("FIX Engine") in use by their respective name, version, and vendor as part of the Logon(35 = A) message in the ApplicationSystemName(1603), ApplicationSystemVersion(1604), ApplicationSystemVendor(1605), FIXEngineName(1600), FIXEngineVersion(1601), and the FIXEngineVendor(1602) fields.

4.4.10 Responding to a request to establish a FIX session

The acceptor should authenticate the identity of the initiator by examining the incoming Logon(35=A) message, referred to as Logon(35=A) request.

The Logon(35 = A) message should contain the data necessary to support the authentication method chosen by counterparty agreement.

The following fields are available for exchanging credentials for authentication:

Table 2 — Fields in the Logon(35=A) message used for authentication

Field	Use
Username(553)	Clear text userid
Password(554)	Clear text password - use is not recommended
NewPassword(925)	Clear text new password - use is not recommended
EncryptedPasswordMethod(1400)	Method of encryption used for password field
EncryptedPasswordLen(1401)	Length in octets of EncryptedPassword(1402)
EncryptedPassword(1402)	Encrypted password
EncryptedNewPasswordLen(1403)	Length in octets of EncryptedNewPassword(1403)
EncryptedNewPassword(1404)	Encrypted new password permits changing password during logon
RawDataLength(95)	Length in octets of RawData(96)
RawData(96)	Binary octet stream that can be used for authentication

SessionStatus(1409) may be used by counterparty agreement to communicate session status regarding authentication on both Logon(35 = A) and Logout(35 = 5) messages.[5]

The acceptor shall respond with a Logon(35 = A) message to accept the FIX connection initiation when the request is accepted, referred to as the Logon(35 = A) acknowledgement.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

[4] Functionality added by EP113.
[5] Field added in EP56

The session acceptor should send a Logout(35 = 5) message to reject FIX connection initiation and provide a reason for the rejection in the Text(58) field and then immediately terminate the transport layer connection.

The initiator should not transmit any application message until the Logon(35 = A) acknowledgement has been received from the acceptor (see the next section for further recommendations).

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

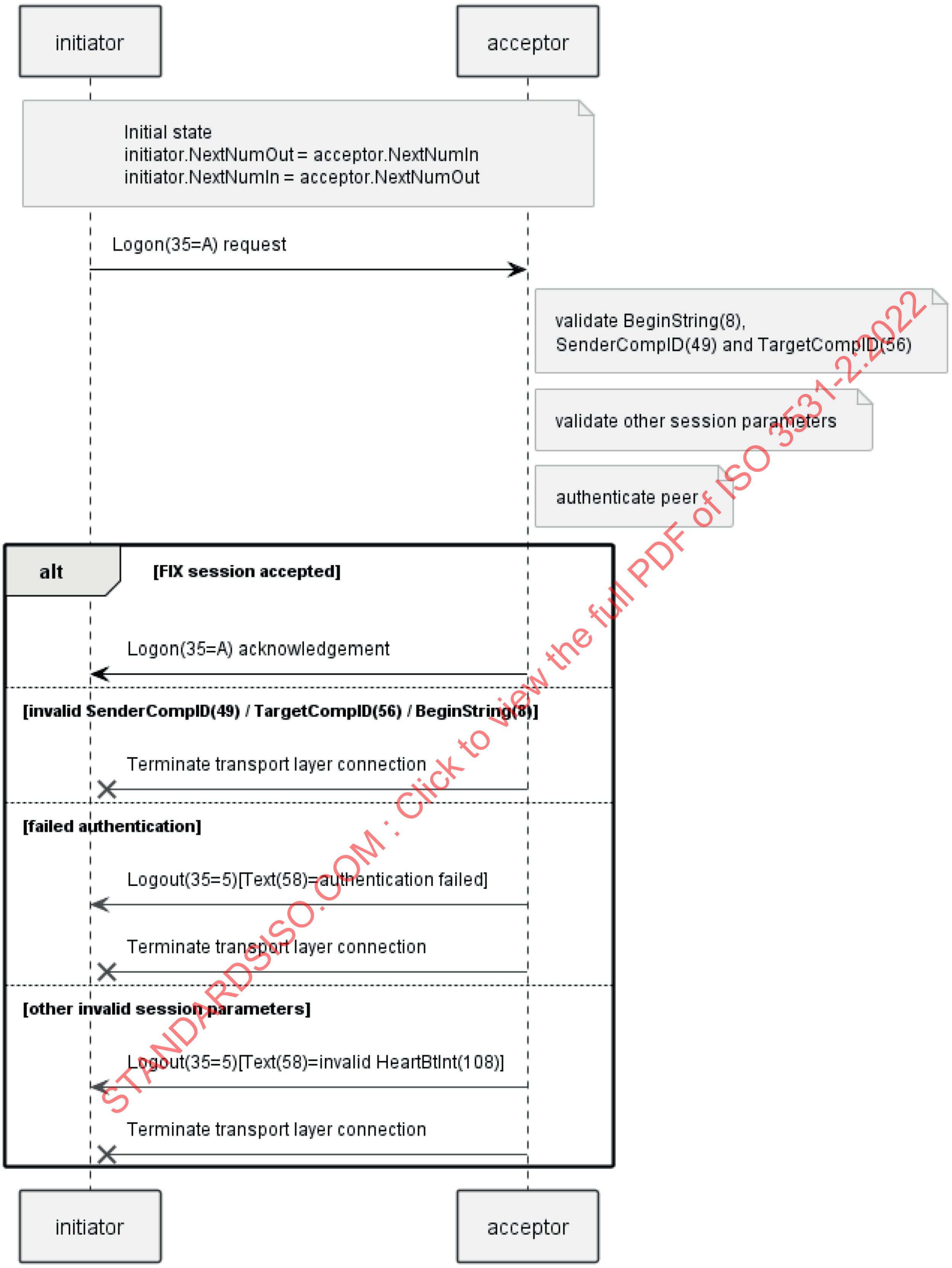

initiator
acceptor
Initial state
initiator.NextNumOut = acceptor.NextNumIn
initiator.NextNumIn = acceptor.NextNumOut
Logon(35=A) request
validate BeginString(8), SenderCompID(49) and TargetCompID(56)
validate other session parameters
authenticate peer
alt
[FIX session accepted]
Logon(35=A) acknowledgement
[invalid SenderCompID(49) / TargetCompID(56) / BeginString(8)]
Terminate transport layer connection
[failed authentication]
Logout(35=5)[Text(58)=authentication failed]
Terminate transport layer connection
[other invalid session parameters]
Logout(35=5)[Text(58)=invalid HeartBtInt(108)]
Terminate transport layer connection
initiator
acceptor

Figure 2 — FIX connection establishment

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.4.11 Initial synchronization of messages in a FIX connection

The acceptor shall compare MsgSeqNum(34) in the Logon(35=A) request received from the initiator with NextNumIn maintained by the acceptor.

The initiator shall compare MsgSeqNum(34) in the Logon(35 = A) acknowledgement received from the acceptor with NextNumIn maintained by the initiator.

If MsgSeqNum(34) is present in the incoming Logon(35 = A) message and equal to NextNumIn, then normal processing may commence over the FIX connection.

If MsgSeqNum(34) is present in the incoming Logon(35 = A) message and greater than NextNumIn, then message recovery must be performed.

If MsgSeqNum(34) is present in the incoming Logon(35 = A) message and is less than NextNumIn, then a Logout(35 = 5) message should be sent assuming that an error exists in the state of either the initiator or acceptor FIX session processor, followed by a termination of the transport layer connection.

The initiator and acceptor should wait a short period of time following receipt of the Logon(35 = A) message from the counterparty before transmitting queued or new application messages to permit both sides to synchronize the FIX session.

Alternatively, the initiator and acceptor may send a TestRequest(35 = 1) message and wait for the HeartBeat(35 = 0) message sent in response to the TestRequest(35 = 1) message, before sending queued or new messages, as this practice forces both sides to perform message synchronization.

4.4.12 Synchronization after successful logon

The following sequence diagrams show three scenarios.

1. acceptor requires retransmission of messages from initiator
2. initiator requires retransmission of messages from acceptor
3. both acceptor and initiator require retransmission of messages

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

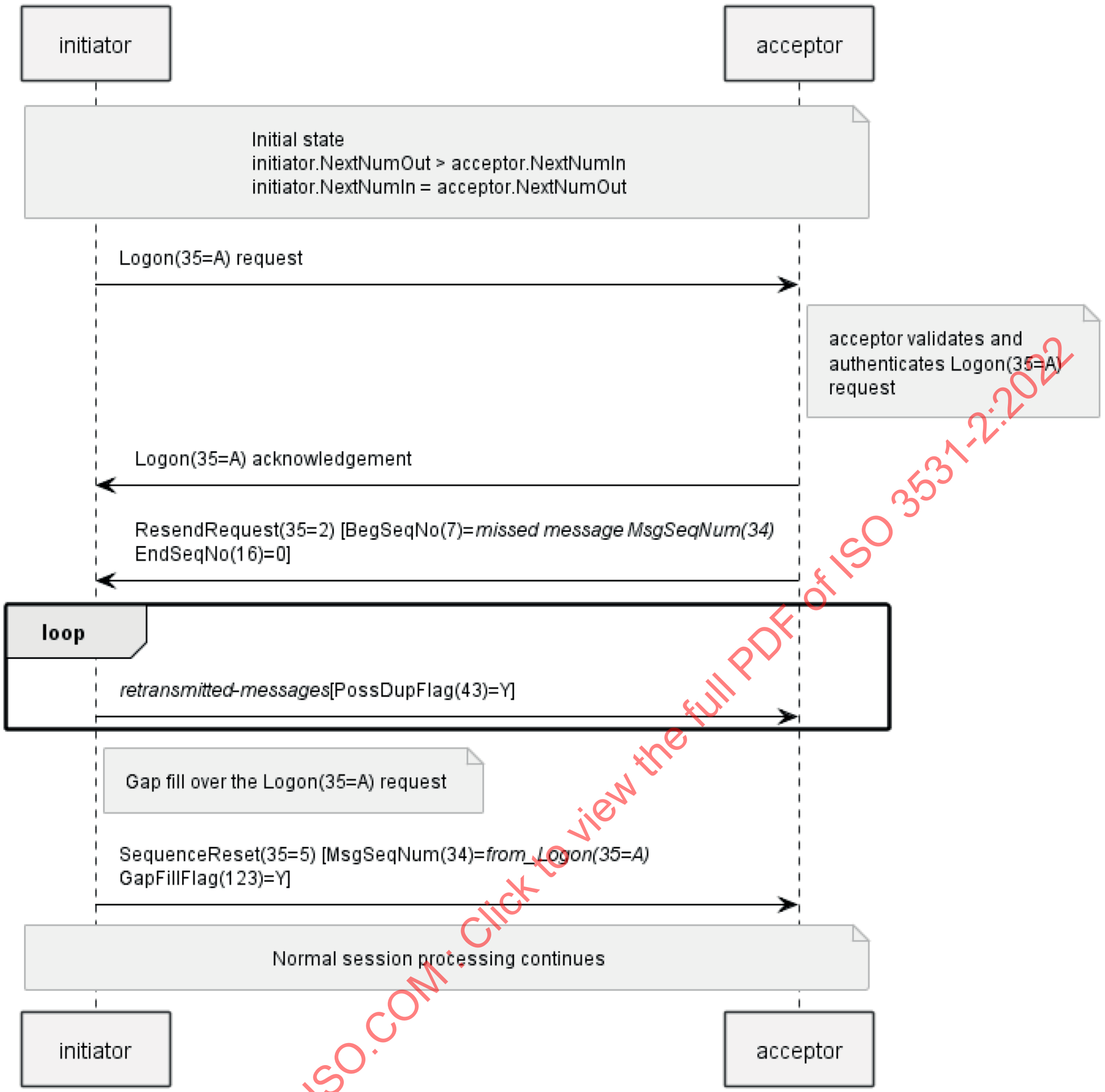

initiator
acceptor
Initial state
initiator.NextNumOut > acceptor.NextNumIn
initiator.NextNumIn = acceptor.NextNumOut
Logon(35=A) request
acceptor validates and authenticates Logon(35=A) request
Logon(35=A) acknowledgement
ResendRequest(35=2) [BegSeqNo(7)=missed message MsgSeqNum(34) EndSeqNo(16)=0]
loop
retransmitted-messages[PossDupFlag(43)=Y]
Gap fill over the Logon(35=A) request
SequenceReset(35=5) [MsgSeqNum(34)=from_Logon(35=A) GapFillFlag(123)=Y]
Normal session processing continues
initiator
acceptor

Figure 3 — ResendRequest(35=2) sent by acceptor after Logon(35=A) acknowledgement

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

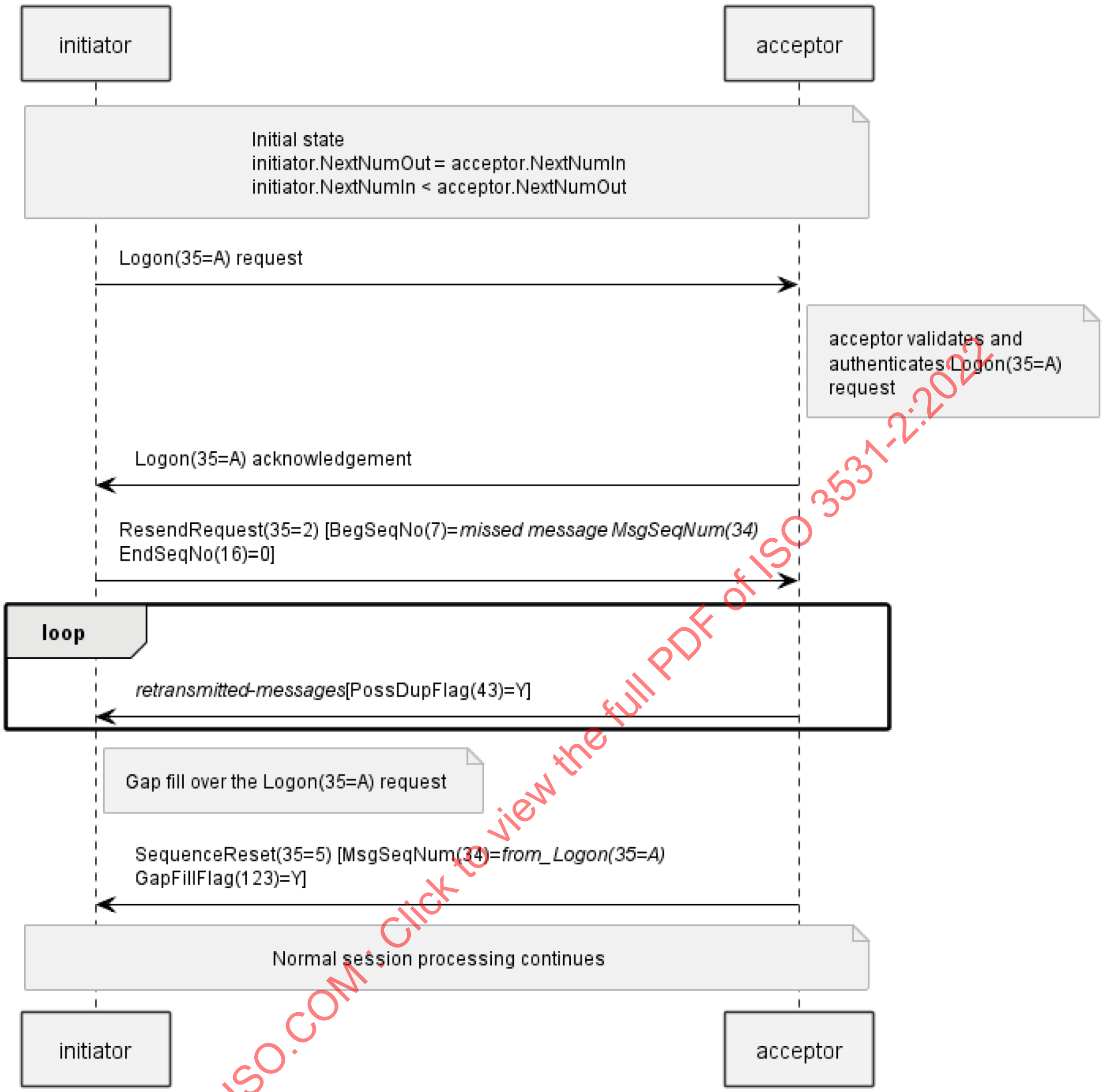
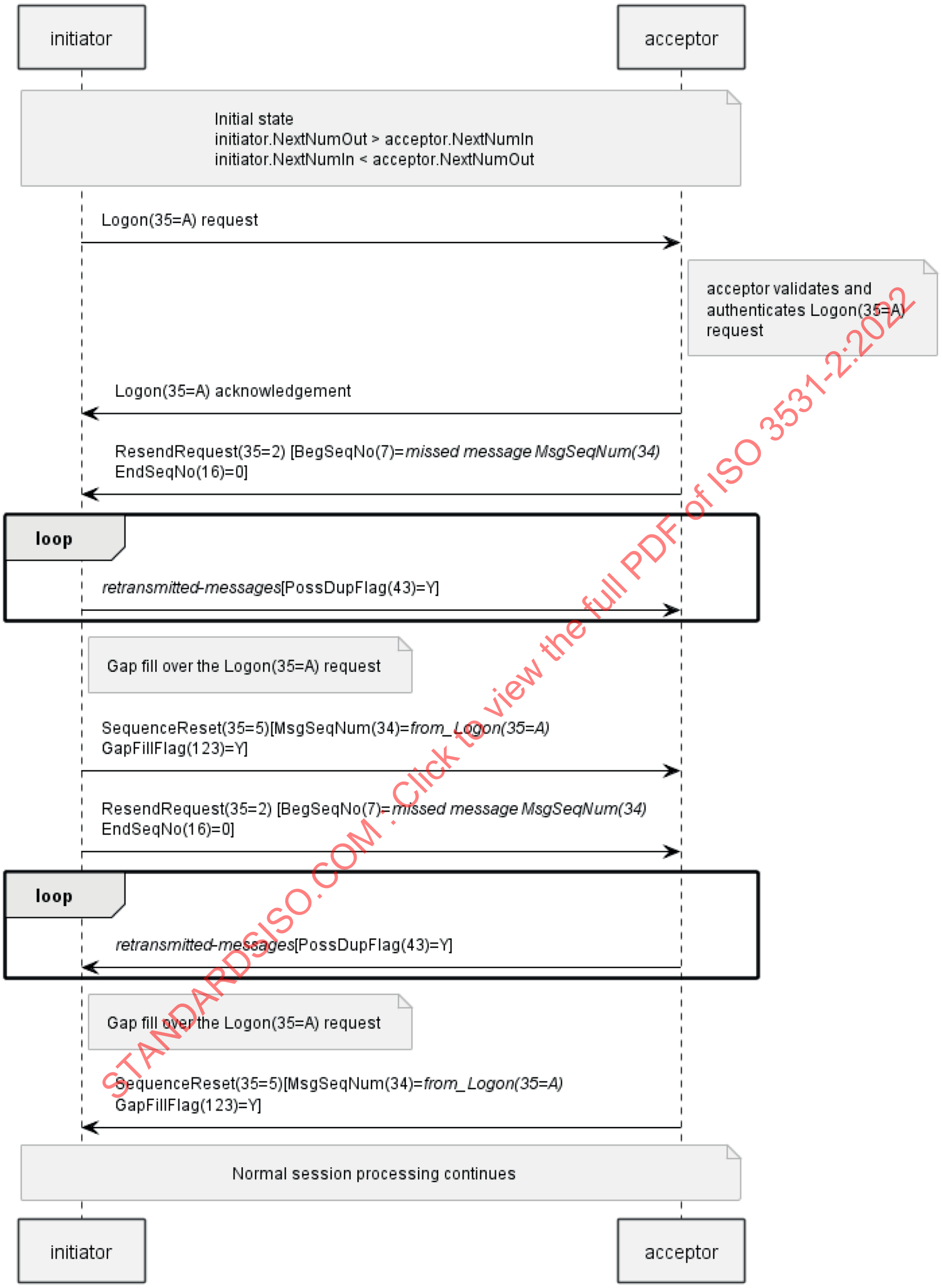

initiator
acceptor
Initial state
initiator.NextNumOut = acceptor.NextNumIn
initiator.NextNumIn < acceptor.NextNumOut
Logon(35=A) request
acceptor validates and authenticates Logon(35=A) request
Logon(35=A) acknowledgement
ResendRequest(35=2) [BegSeqNo(7)=*missed message MsgSeqNum(34)* EndSeqNo(16)=0]
loop
retransmitted-messages[PossDupFlag(43)=Y]
Gap fill over the Logon(35=A) request
SequenceReset(35=5) [MsgSeqNum(34)=*from_Logon(35=A)* GapFillFlag(123)=Y]
Normal session processing continues
initiator
acceptor

Figure 4 — ResendRequest(35=2) sent by initiator after Logon(35=A) acknowledgement

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022


initiator
acceptor
Initial state
initiator.NextNumOut > acceptor.NextNumIn
initiator.NextNumIn < acceptor.NextNumOut
Logon(35=A) request
acceptor validates and authenticates Logon(35=A) request
Logon(35=A) acknowledgement
ResendRequest(35=2) [BegSeqNo(7)=*missed message MsgSeqNum(34)* EndSeqNo(16)=0]
loop
retransmitted-messages[PossDupFlag(43)=Y]
Gap fill over the Logon(35=A) request
SequenceReset(35=5)[MsgSeqNum(34)=*from_Logon(35=A)* GapFillFlag(123)=Y]
ResendRequest(35=2) [BegSeqNo(7)=*missed message MsgSeqNum(34)* EndSeqNo(16)=0]
loop
retransmitted-messages[PossDupFlag(43)=Y]
Gap fill over the Logon(35=A) request
SequenceReset(35=5)[MsgSeqNum(34)=*from_Logon(35=A)* GapFillFlag(123)=Y]
Normal session processing continues
initiator
acceptor

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 5 — ResendRequest(35=2) sent by initiator and acceptor after Logon(35=A) acknowledgement

4.5 Extended features for FIX session and FIX connection initiation

4.5.1 Using NextExpectedMsgSeqNum(789) to synchronize a FIX session

The FIX session processors may use the NextExpectedMsgSeqNum(789) field in the Logon(35=A) message to synchronize a FIX session by counterparty agreement defined in the FIX session rules of engagement[6].

The time required to resume a FIX session over a new FIX connection can be reduced by alerting the peer to the next message that is expected using NextExpectedMsgSeqNum(789) in the Logon(35 = A) message. Any missing messages can be retransmitted by the peer without requiring to use ResendRequest(35 = 2) messages.

A FIX session may implement next expected sequence number processing to reduce the amount of message retransmission needed to re-establish a FIX session by counterparty agreement specified in the rules of engagement.

Each FIX session processor may populate the NextExpectedMsgSeqNum(789) with the next message sequence number expected from its peer on the Logon(35 = A) when establishing a FIX connection.

Peers should not generate a ResendRequest(35 = 2) message based on MsgSeqNum(34) of the incoming Logon(35 = A) message but should expect any gaps to be filled automatically via the following process.

The receiving FIX session processor receiving a Logon(35 = A) message with NextExpectedMsgSeqNum(789) present shall compare the value of NextExpectedMsgSeqNum(789) with NextNumOut.

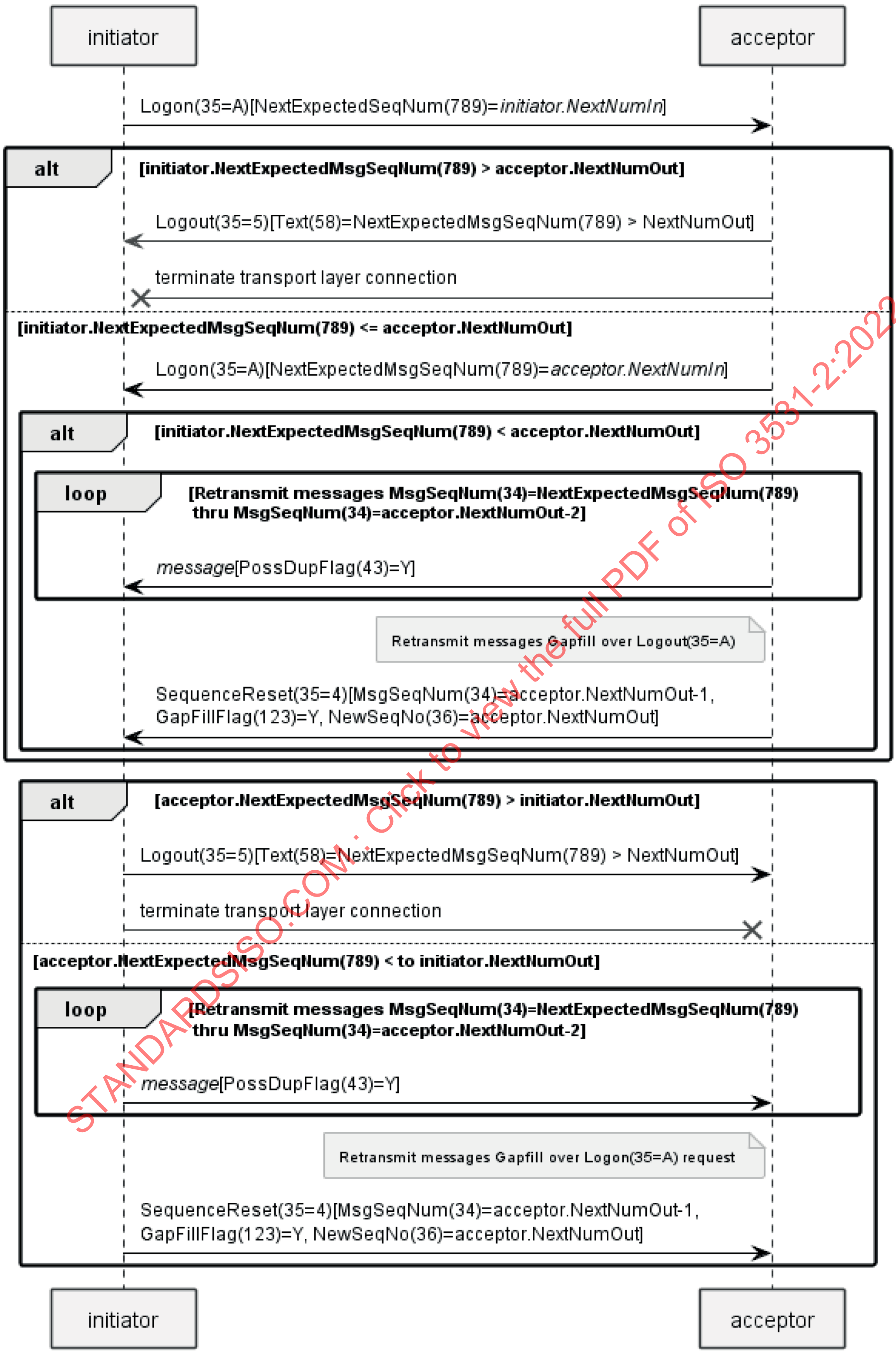
- If the received NextExpectedMsgSeqNum(789) is equal to NextNumOut, proceed sending new messages beginning with that number.

- If the received NextExpectedMsgSeqNum(789) is less than NextNumOut, perform message recovery for messages starting with the message with MsgSeqNum(34) equal to the NextExpectedMsgSeqNum(789) through to the message with MsgSeqNum(34) equal to NextNumOut - 1.

- If the received NextExpectedMsgSeqNum(789) is greater than NextNumOut, send a Logout(35=5) message to end the session with the Text(58) field populated explaining that “NextExpectedMsgSeqNum(789) > than last message sent”.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

[6] FIX Orchestra may be used to communicate which FIX session profile and which operating mode is being used for a FIX enabled service.


initiator
acceptor
Logon(35=A)[NextExpectedSeqNum(789)=initiator.NextNumIn]
alt
[initiator.NextExpectedMsgSeqNum(789) > acceptor.NextNumOut]
Logout(35=5)[Text(58)=NextExpectedMsgSeqNum(789) > NextNumOut]
terminate transport layer connection
[initiator.NextExpectedMsgSeqNum(789) <= acceptor.NextNumOut]
Logon(35=A)[NextExpectedMsgSeqNum(789)=acceptor.NextNumIn]
alt
[initiator.NextExpectedMsgSeqNum(789) < acceptor.NextNumOut]
loop
[Retransmit messages MsgSeqNum(34)=NextExpectedMsgSeqNum(789) thru MsgSeqNum(34)=acceptor.NextNumOut-2]
message[PossDupFlag(43)=Y]
Retransmit messages Gapfill over Logout(35=A)
SequenceReset(35=4)[MsgSeqNum(34)=acceptor.NextNumOut-1, GapFillFlag(123)=Y, NewSeqNo(36)=acceptor.NextNumOut]
alt
[acceptor.NextExpectedMsgSeqNum(789) > initiator.NextNumOut]
Logout(35=5)[Text(58)=NextExpectedMsgSeqNum(789) > NextNumOut]
terminate transport layer connection
[acceptor.NextExpectedMsgSeqNum(789) < to initiator.NextNumOut]
loop
[Retransmit messages MsgSeqNum(34)=NextExpectedMsgSeqNum(789) thru MsgSeqNum(34)=acceptor.NextNumOut-2]
message[PossDupFlag(43)=Y]
Retransmit messages Gapfill over Logon(35=A) request
SequenceReset(35=4)[MsgSeqNum(34)=acceptor.NextNumOut-1, GapFillFlag(123)=Y, NewSeqNo(36)=acceptor.NextNumOut]
initiator
acceptor

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 6 — Using NextExpectedMsgSeqNum(789) to synchronize the session during the logon process

4.5.2 Using ResetSeqNumFlag(141) to reset FIX session for 24 h connectivity

The FIX session may be reset to NextNumIn=1 and NextNumOut=1 over an active FIX session. This capability is provided for continuously operating markets that need to periodically reset the FIX session. When using the ResetSeqNumFlag(141) to maintain 24 hour connectivity and establish a new set of sequence numbers, the process should be as follows.

Counterparties must agree to the time of day when the FIX session reset will be initiated.

Counterparties must agree which peer will initiate the FIX session reset.

The peer that initiates the FIX session reset may send a TestRequest(35 = 1) message and wait for the HeartBeat(35 = 0) response to ensure that there are no message gaps prior to sending a Logon(35 = A) message to reset the FIX session.

The peer initiating the FIX session reset should set its NextNumIn to 1 and NextNumOut to 1, then send a Logon(35 = A) message with ResetSeqNumFlag(141) set to “Y” and MsgSeqNum(34) set to 1.

The peer receiving the Logon(35 = A) message should set its NextNumIn to 2 and NextNumOut to 1, then send a Logon(35 = A) acknowledgement with ResetSeqNumFlag(141) set to “Y” and with MsgSeqNum(24) set of 1.

Upon completion of the session reset, both peers must have NextNumIn = 2 and NextNumOut = 2.

If the peer is not configured to accept resetting of an inbound session the peer should send a Logout(35 = 5) with Text(58) indicating that resetting the sequence number is not supported and then terminate the transport layer connection.

4.5.3 Using ResetSeqNumFlag(141) to reset FIX session during FIX connection establishment[7]

The initiator of a FIX session may reset the FIX session upon initial logon via sending the Logon(35 = A) with ResetSeqNumFlag(141) set to “Y”.

Counterparties must agree to support the reset of a FIX session during FIX connection establishment.

Operational risks, such as order overfills, can occur if both parties are not configured to support resetting the FIX session upon initial logon. Users of the FIX session layer are warned to take special care when permitting the FIX sessions to be reset upon FIX connection establishment.

If the acceptor is not configured to accept resetting of a FIX session during FIX connection establishment, the acceptor should send a Logout(35 = 5) with Text(58) indicating that resetting the sequence number upon FIX connection establishment is not supported then terminate the transport layer connection.

4.5.4 Using initiator state to restore acceptor session state[8]

An acceptor may re-establish its state via the exchange of NextExpectedMsgSeqNum(789).

This should only be used for FIX service offerings where application message recovery is performed by the application layer instead of the FIX session layer.

The state of the FIX session is assumed to be:

- the acceptor FIX session processor NextNumOut is unknown and NextNumIn is unknown.
- the initiator FIX session processor has maintained and knows its state (the values of NextNumIn and NextNumOut) between FIX connections.

[7] Functionality added by EP124.
[8] Functionality added by EP124.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Acceptor state may be recovered when NextExpectedMsgSqNum(789) is present in the Logon(35=A) request by:

- The acceptor may reconstruct its session state solely based upon the Logon(35=A) request from an initiator if NextExpectedMsgSeqNum(789) is present in the Logon(35=A) request.
- The acceptor should set its NextNumIn to the MsgSeqNum(34) + 1 from the Logon(35=A) request and set its NextNumOut to the NextExpectedMsgSeqNum(789) from the Logon(35=A) request.
- The acceptor should then send a SequenceReset(35=4) message with GapFillFlag(123) set to "Y" and NewSeqNo(36) set to NextNumOut+1 and AppLevelIndicator(1744) set to "1" (application layer recovery is needed) to inform the initiator that application messages will need to be recovered.

Acceptor state may be recovered when NextExpectedMsgSeqNum(789) is not present in the Logon(35=A) request by:

- The acceptor may reconstruct its session state based upon the Logon(35=A) request from an initiator by setting its NextNumIn to the MsgSeqNum(34) + 1 from the Logon(35=A) request and setting its NextNumOut to 1 when the NextExpectedMsgSeqNum(789) is not present in the Logon(35=A) request.
- The initiator should respond to the Logon(35=A) acknowledgement with a Logout(35=5) with SessionStatus(1409) set to 9 (received MsgSeqNum(34) is too low) and NextExpectedMsgSeqNum(789) set to the initiator's NextNumIn, followed by terminating the transport layer connection.
- The acceptor should set its NextNumOut to the NextExpectedSeqNum(789) value from the Logout(35=5) received from the initiator.
- The initiator should then transmit a new Logon(35=A) request with MsgSeqNum(34) set to its NextNumOut value.
- The acceptor should respond with Logon(35=A) acknowledgement with the MsgSeqNum(34) set to the NextNumOut value obtained from the previous initiator Logout(35=5).
- The acceptor should then send a SequenceReset(35=4) message with GapFillFlag(123) set to "Y" and NewSeqNo(36) set to NextNumOut+1 and AppLevelIndicator(1744) set to "1" (application layer recovery is needed) to inform the initiator that application messages may need to be recovered.

4.6 Message exchange during a FIX connection

After establishing a FIX connection within a FIX session and synchronizing messages, normal message exchange begins.

The initiator and acceptor may send a TestRequest(35 = 1) message and wait for the HeartBeat(35 = 0) message sent in response to the TestRequest(35 = 1) message, before sending queued or new messages, to further confirm that the FIX connection is established.

The initiator and acceptor shall validate incoming FIX messages to ensure proper tagvalue encoding.

The integrity of message data content shall be verified for message length and the tagvalue encoding checksum according to the FIX tagvalue encoding specification.

The initiator and acceptor shall synchronize their messages and detect gaps by comparing the MsgSeqNum(34) field in each message received from its peer with the NextNumIn it maintains and increments after successful receipt of each message.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

A FIX session processor that encounters a sequence number gap shall perform message recovery.

4.6.1 FIX connection keep alive (heartbeat)

The FIX session processor must send a HeartBeat(35=0) message to the peer during periods of inactivity, ensuring that the HeartBeat(35=0) message will be received by the peer within the heartbeat interval.

If a FIX session processor does not receive a message within the heartbeat interval, test request processing should be initiated.

4.6.2 Garbled message processing

A message shall be considered garbled if any of the following occur as a result of encoding, decoding, or transmission errors:

- BeginString(8) is not the first tag in a message or is not one of the defined FIX session profile identifiers (“8=FIX.4.4”, “8=FIXT.1.1”).
- BodyLength(9) is not the second tag in a message or does not contain the correct byte count.
- MsgType(35) is not the third tag in a message.
- Checksum(10) is not the last tag or contains an incorrect value.

The FIX session layer presumes that a garbled message is received due to a transmission error rather than a FIX session processor defect and should be recoverable from the peer.

The receiving FIX session processor must disregard the garbled message and not increment NextNumIn.

The FIX session processor should log each encountered garbled message to assist in problem detection and diagnosis.

Receipt of the next valid FIX message after a garbled message was ignored will result in the detection of a sequence gap as the MsgSeqNum(34) will be greater than NextNumIn which should then cause the receiving FIX session processor to initiate message recovery.

It is recommended that FIX session processors implement logic to recognize the possible infinite retransmission loop, which may be encountered when a garbled message is repeatedly retransmitted due to a fault in the peer FIX session processor.

4.6.3 Missing sequence number

If MsgSeqNum(34) is missing, a Logout(35=5) message should be sent, terminating the FIX connection with the Text(58) field describing the missing field, as this likely indicates a serious application error that is likely only circumvented by software modification.

4.6.4 Rejecting invalid messages

Incoming messages should be rejected using the Reject(35=3) message when:

- an incoming message is not garbled but does not contain the appropriate required fields may be rejected by sending a Reject(35=3) message to the peer.
- MsgType(35) of the incoming message is unknown or not supported.

As a rule, messages should be forwarded to the trading application for business level rejections whenever possible instead of being rejected at the session layer.

Rejected messages must be logged and NextNumIn incremented by 1.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Generation and receipt of a Reject(35 = 3) message indicates a serious error that is possibly the result of faulty logic or noncompliance with the rules of engagement in either peer.

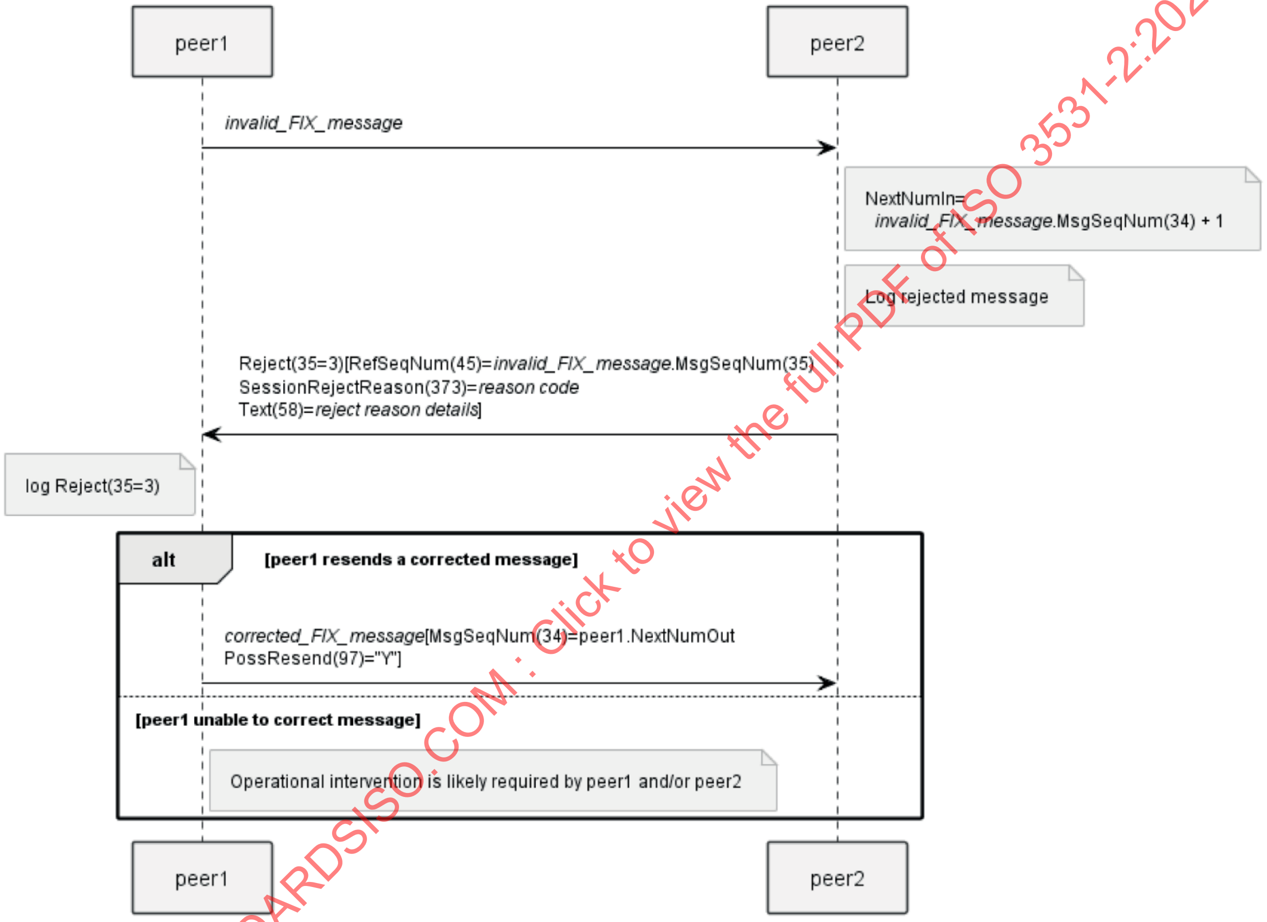
A diagnostic message describing the error should be provided in the Text(58) field of the Reject(35 = 3) message and the SessionRejectReason(373) field set to the appropriate reject reason.

The MsgSeqNum(34) from the message being rejected should be returned in RefSeqNum(45).

The RefTagID(371) should be set to the tag of the field associated with the reject reason.

The RefMsgType(372) field should be the MsgType(35) of the rejected message.

If the sending application chooses to retransmit the rejected message, the message being resent must be assigned a new MsgSeqNum(34) and sent with PossResend(97) set to "Y".


peer1
peer2
invalid_FIX_message
NextNumIn=
invalid_FIX_message.MsgSeqNum(34) + 1
Log rejected message
Reject(35=3)[RefSeqNum(45)=invalid_FIX_message.MsgSeqNum(35)
SessionRejectReason(373)=reason code
Text(58)=reject reason details]
log Reject(35=3)
alt
[peer1 resends a corrected message]
corrected_FIX_message[MsgSeqNum(34)=peer1.NextNumOut
PossResend(97)="Y"]
[peer1 unable to correct message]
Operational intervention is likely required by peer1 and/or peer2
peer1
peer2

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 7 — Session layer reject using Reject(35=3) due to invalid message received

4.6.5 Test Request Processing

A TestRequest(35=1) is sent to determine the status of the peer FIX session processor.

The TestReqID(112) field must be present in the TestRequest(35 = 1) message.

The recipient of a TestRequest(35 = 1) message must respond by sending a HeartBeat(35 = 0) message with TestReqID(112) present and set to the TestReqID(112) value from the TestRequest(35 = 1) message.

© ISO 2022 – All rights reserved

Depending upon the implementation of the FIX session processor it may be possible to receive HeartBeat(35 = 0) messages from the peer even though the peer FIX session processor is not functioning properly. A TestRequest(35 = 1) may be sent to determine if the peer FIX session processor is still responsive. Not receiving a valid response to the TestRequest(35 = 1), which is a HeartBeat(35 = 0) message containing the value of the TestReqID(112) from the TestRequest(35 = 1) message, provides some evidence that the peer is not functioning properly.

Failure to receive a HeartBeat(35 = 0) response with TestReqID(112) present and set to the value from the TestRequest(35 = 1) message is considered an error. The FIX connection should be terminated by sending a Logout(35 = 5) message with the Text(58) field stating that the session was terminated due to lack of a response to a TestRequest(35 = 1) message, followed by terminating the transport layer connection.

The period of time that the FIX session processor should wait for the HeartBeat(35 = 0) response should be defined as a *TestRequestThreshold* multiplied by the *heartbeat interval*.

A reasonable *TestRequestThreshold* is anywhere from 1.2 to 2.0 depending on latency sensitivity of the application using the FIX session layer.

The *TestRequestThreshold* may be specified by counterparty agreement or specified in the rules of engagement[9].

The operator of the FIX session processor may send a TestRequest(35 = 1) message at any time during a FIX connection. The following figure shows an example of a TestRequest(35 = 1) message issued to determine if the peer is still responsive.

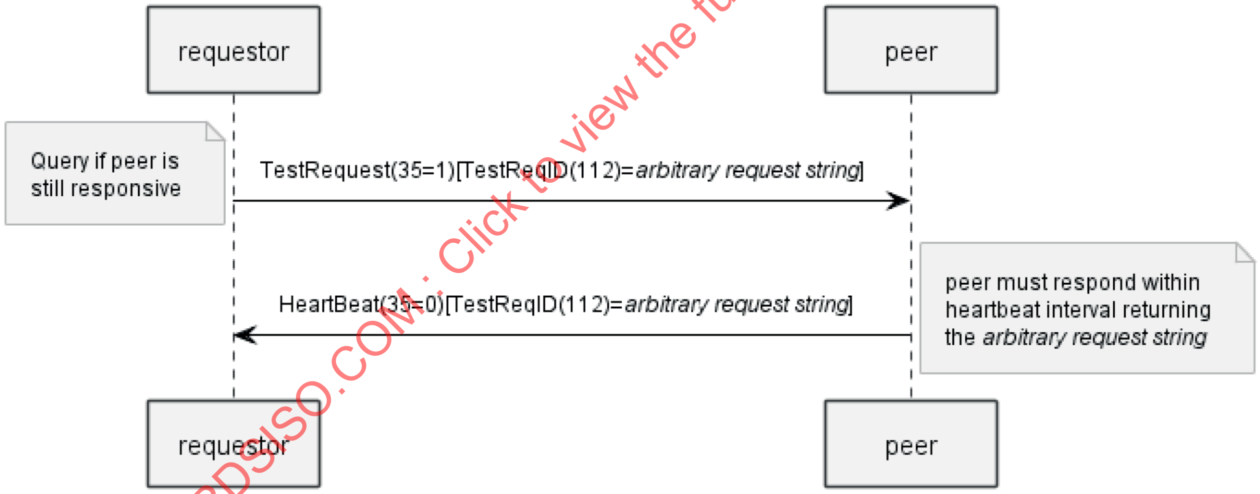

requestor
peer
Query if peer is still responsive
TestRequest(35=1)[TestReqID(112)=*arbitrary request string*]
HeartBeat(35=0)[TestReqID(112)=*arbitrary request string*]
peer must respond within heartbeat interval returning the *arbitrary request string*
requestor
peer

Figure 8 — Issuing a TestRequest(35=1) to determine if peer is still responsive

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.7 FIX connection termination

Either the initiator or the acceptor may send a Logout(35=5) message to request termination of a FIX connection. Normal termination of the message exchange session shall be completed via the exchange of Logout(35=5) messages. Termination by other means should be considered an abnormal condition and dealt with as an error. FIX connection termination without receiving a Logout(35=5) message should treat the peer as logged out.

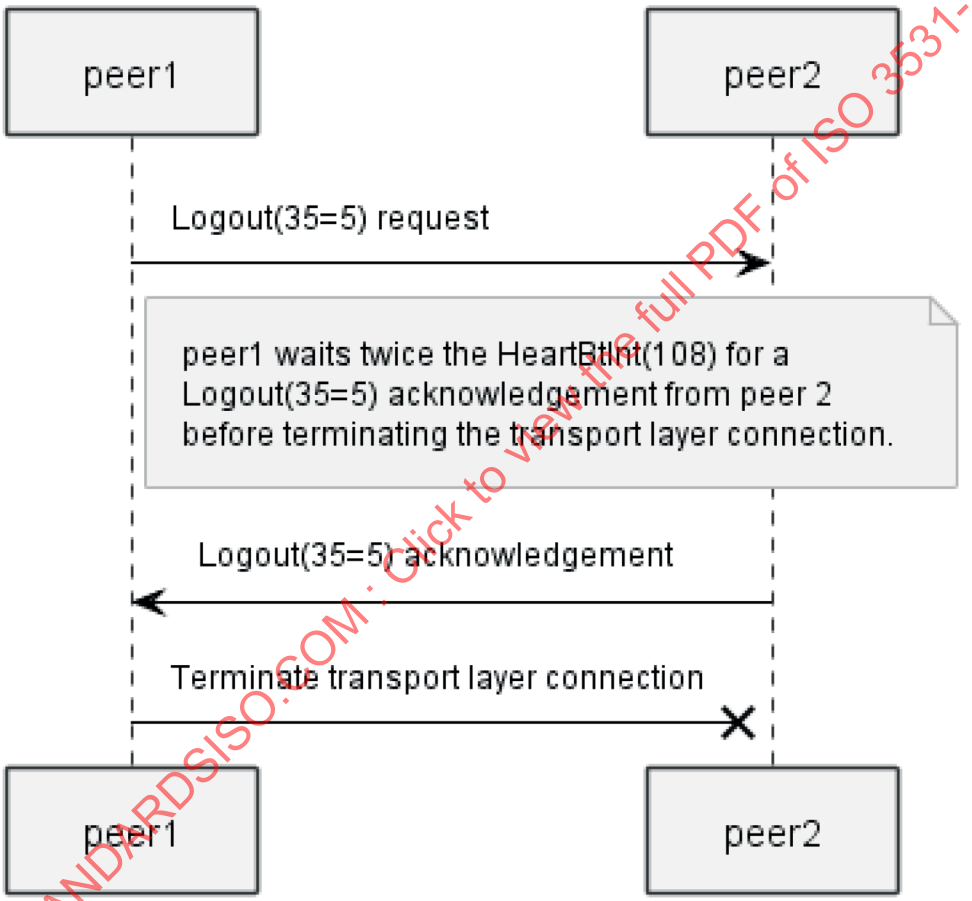
[9] The FIX Orchestra Interface specification may be used to specify the *TestRequestThreshold*.

A FIX session processor may send a TestRequest(35 = 1) message before sending the Logout(35 = 5) message to force a heartbeat from the other side. This helps to ensure that there are no sequence number gaps, which indicate that a ResendRequest(35 = 2) message is necessary before completing the session.

The Logout(35 = 5) initiator must wait for the peer to respond with a Logout(35 = 5) acknowledgement message before terminating the FIX connection. This gives the peer a chance to perform any gap fill operations that may be necessary. Once the messages from the ResendRequest(35 = 2) message have been received, the peer should issue the Logout(35 = 5) message.

The Logout(35 = 5) initiator should terminate the FIX connection if the peer does not respond within a reasonable time frame. Twice the value specified in the HeartBtInt(108) field of the Logon(35 = A) message is recommended.

4.7.1 Normal logout processing


peer1
peer2
Logout(35=5) request
peer1 waits twice the HeartBtInt(108) for a Logout(35=5) acknowledgement from peer 2 before terminating the transport layer connection.
Logout(35=5) acknowledgement
Terminate transport layer connection
peer1
peer2

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 9 — Successful Logout scenario

4.7.2 Logout without acknowledgement from peer

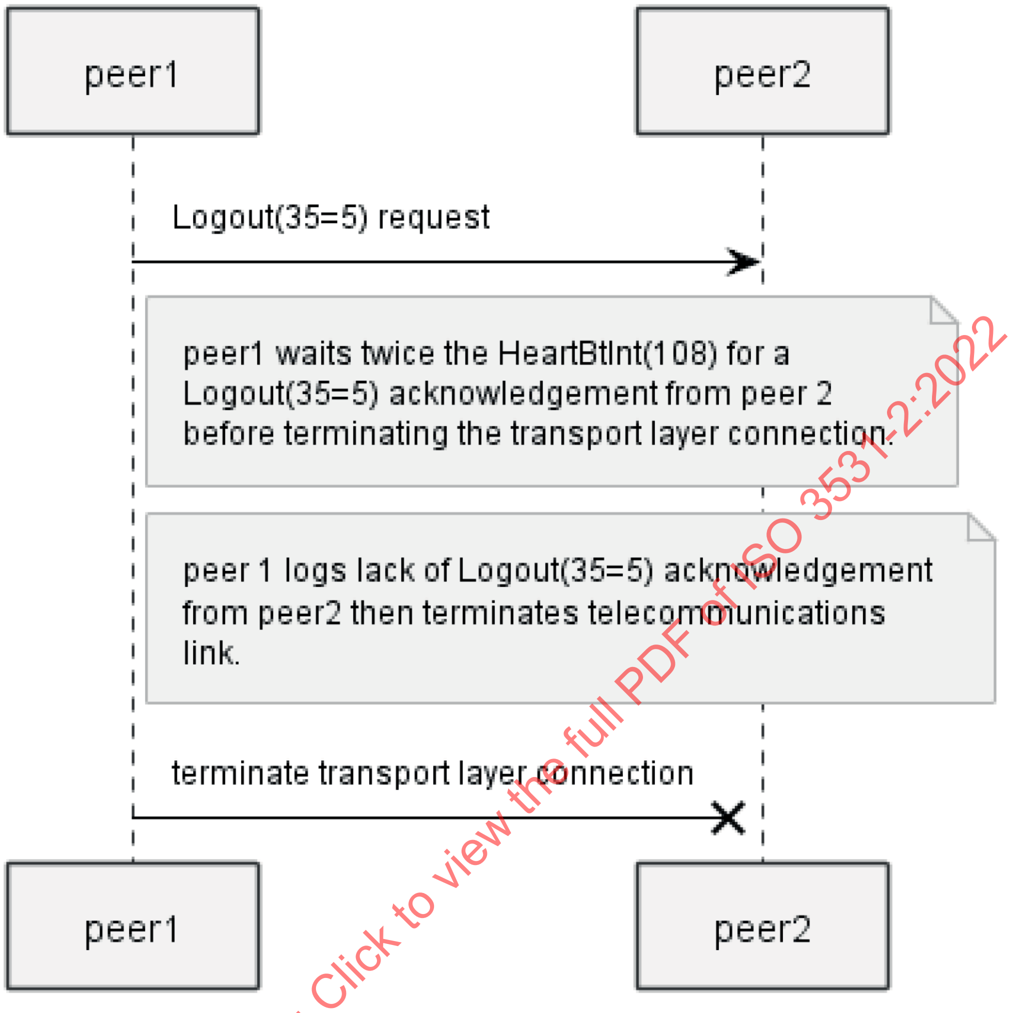
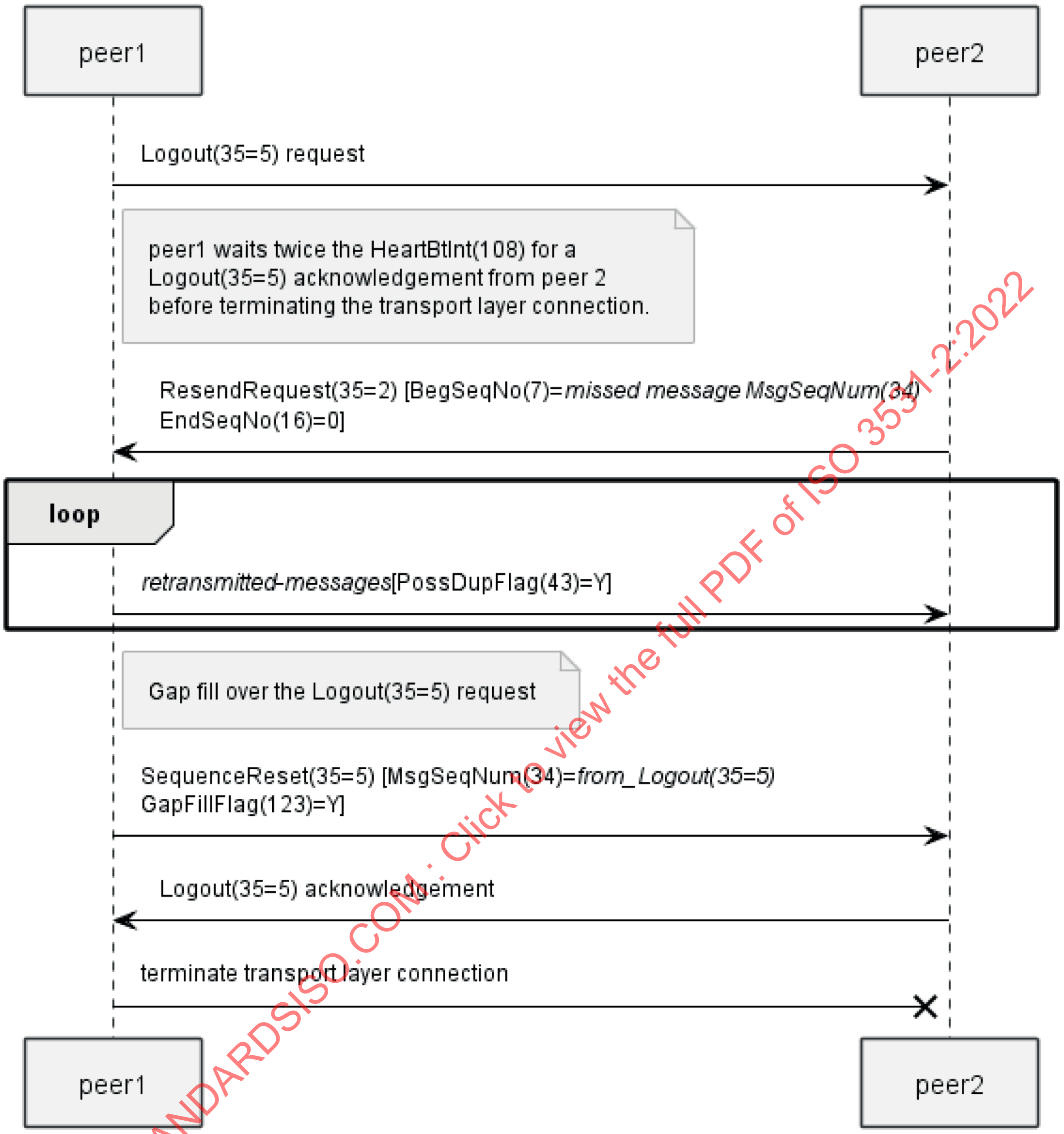

peer1
peer2
Logout(35=5) request
peer1 waits twice the HeartBtInt(108) for a Logout(35=5) acknowledgement from peer 2 before terminating the transport layer connection.
peer 1 logs lack of Logout(35=5) acknowledgement from peer2 then terminates telecommunications link.
terminate transport layer connection
peer1
peer2

Figure 10 — Logout(35=5) acknowledgement not received

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.7.3 Logout with retransmission of missed messages


peer1
peer2
Logout(35=5) request
peer1 waits twice the HeartBtInt(108) for a Logout(35=5) acknowledgement from peer 2 before terminating the transport layer connection.
ResendRequest(35=2) [BegSeqNo(7)=*missed message MsgSeqNum(34)* EndSeqNo(16)=0]
loop
retransmitted-messages[PossDupFlag(43)=Y]
Gap fill over the Logout(35=5) request
SequenceReset(35=5) [MsgSeqNum(34)=*from_Logout(35=5)* GapFillFlag(123)=Y]
Logout(35=5) acknowledgement
terminate transport layer connection
peer1
peer2

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 11 — Processing of ResendRequest(35=2) messages before Logout(35=5) acknowledgement

The impact of a FIX connection termination on the application layer is left to the application layer; it is undefined in the FIX session layer.

© ISO 2022 – All rights reserved

4.7.4 When to terminate a FIX connection by terminating the transport layer connection instead of sending a Logout(35 = 5)

In general, a Logout(35=5) message should always be sent prior to shutting down a connection. If it is being sent due to an error condition, the Text(58) field of the Logout(35=5) should provide a descriptive reason, so that operational support of the remote FIX system can diagnose the problem. There are exceptions, when it is recommended that a Logout(35=5) message not be sent, these include:

- The FIX session should be immediately terminated by terminating the transport layer connection without sending a Logout(35=5) message when the BeginString(8), SenderCompID(49), TargetCompID(56), or IP address in the Logon(35=A) request of the session initiator is invalid. This logon attempt is possibly an unauthorized access attempt. Terminating the connection without transmitting a Logout(35=5) message avoids divulging information about the SenderCompID(49)/TargetCompID(56) and the version of FIX being used over the transport layer.

- If, during a logon, one receives a second connection attempt while a valid FIX session is already underway for that same SenderCompID(49), the session acceptor should immediately terminate the second connection attempt and not send a Logout(35=5) message. Sending a Logout(35=5) message runs the risk of interfering with and possibly adversely affecting the current active FIX connection. For example, sending a Logout(35=5) message should consume a sequence number that would cause an out of sequence condition for the established FIX session.

In all other cases, if sending a Logout(35=5) message does not create risk or violate security, it should be sent with a descriptive text message in the Text(58) field.

4.8 Extended features for FIX connection Termination

4.8.1 Using NextExpectedMsgSeqNum(789) when terminating FIX connection due to invalid MsgSeqNum(34)[10]

When a FIX session processor receives a Logon(35 = A) message with a MsgSeqNum(34) that is less than the NextNumIn from a peer, a Logout(35 = 5) message may be sent that includes the NextExpectedMsgSeqNum(789) set to NextNumIn and the SessionStatus(1409) set to 9 (received MsgSeqNum(34) is too low). This approach permits the peer to reset their NextNumOut to reestablish the FIX session.

When a FIX session processor receives a Logon(35 = A) message with a NextExpectedMsgSeqNum(789) value that is greater than NextNumOut from a peer, a Logout(35 = 5) message may be sent that includes the SessionStatus(1409) set to 10 (received NextExpectedMsgSeqNum(789) is too high), then terminate the transport layer connection. This approach allows the peer to reset their NextNumIn to the MsgSeqNum(34) present in the Logout(35 = A) message + 1.

When a FIX session processor receives a Logon(35 = A) message with a MsgSeqNum(34) that is less than the NextNumIn and a NextExpectedMsgSeqNum(789) value that is greater than NextNumOut from a peer, a Logout(35 = 5) message may be sent that includes SessionStatus(1409) equal 9 (received MsgSeqNum(34) is too low) and the NextExpectedMsgSeqNum(789) set to NextNumIn, then terminate the transport layer connection. This approach allows the peer to reset their NextNumIn to the MsgSeqNum(34) present in the Logout(35 = 5) message + 1 and the NextNumOut reset to the NextExpectedMsgSeqNum(789) value.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

[10] Functionality added by EP124.

4.9 Message recovery

A message gap occurs when the MsgSeqNum(34) of an incoming message is greater than the NextNumIn maintained by the FIX session processor.

During initialization of a FIX connection or at any time during the FIX connection, a message gap may occur.

When the MsgSeqNum(34) on an inbound message is greater than NextNumIn, the FIX session processor must perform message recovery processing as this is an indication that there were messages sent by the peer that were not received or may have been garbled during transmission.

4.9.1 Ordered message processing

The FIX protocol assumes complete ordered delivery of messages processed in MsgSeqNum(34) order.

FIX session processors must detect gaps in the sequence of MsgSeqNum(34) values by maintaining the next expected sequence number NextNumIn.

If the MsgSeqNum(34) of an incoming message is greater than NextNumIn, then the receiving FIX session processor should request the peer to retransmit messages following the Request retransmission of messages specifications.

The FIX session should be terminated if the incoming message MsgSeqNum(34) is less than NextNumIn and PossDupFlag(43) is not set to "Y", except if the message is the SequenceReset(35 = 4) with the GapFillFlag(123) set to "N" (Sequence Reset). A Logout(35 = 5) message should be sent with the Text(58) field set to indicate that the value in MsgSeqNum(34) is less than NextNumIn when PossDupFlag(43) is not set to "Y" and SessionStatus(1409) set to 9 (received MsgSeqNum(34) is too low.), followed by terminating the transport layer connection.

4.9.2 Request retransmission of messages

There are two alternative approaches to resolving message sequence number gaps.

The two approaches are:

- Request all messages subsequent to the last message received.

  For example, if the receiver misses the second of five messages, the application can ignore messages 3 to 5 and generate a resend request for messages 2 to 5, or, preferably 2 to 0 (where 0 represents all messages after message 2).

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

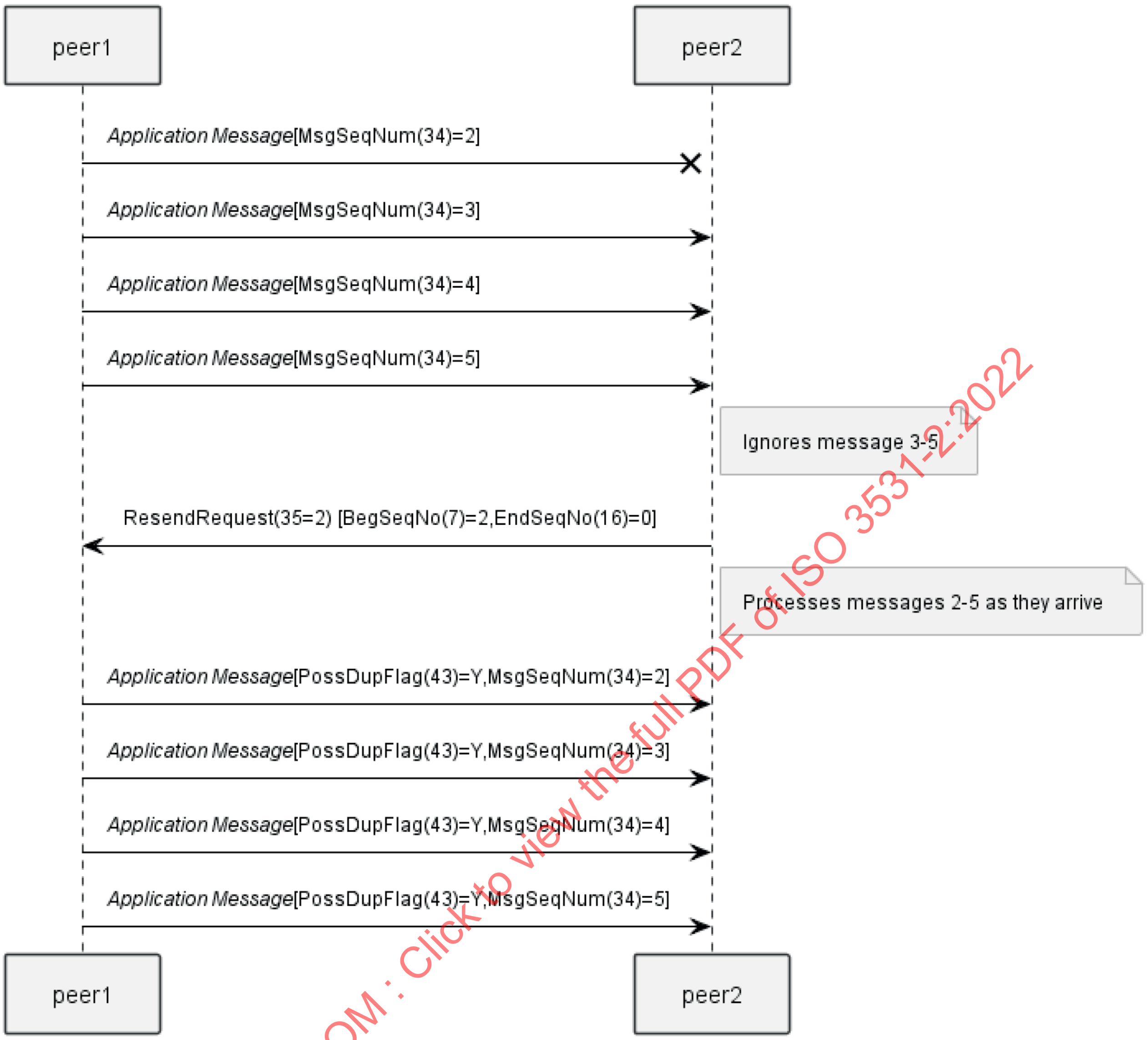

peer1
peer2
Application Message[MsgSeqNum(34)=2]
Application Message[MsgSeqNum(34)=3]
Application Message[MsgSeqNum(34)=4]
Application Message[MsgSeqNum(34)=5]
Ignores message 3-5
ResendRequest(35=2) [BegSeqNo(7)=2,EndSeqNo(16)=0]
Processes messages 2-5 as they arrive
Application Message[PossDupFlag(43)=Y,MsgSeqNum(34)=2]
Application Message[PossDupFlag(43)=Y,MsgSeqNum(34)=3]
Application Message[PossDupFlag(43)=Y,MsgSeqNum(34)=4]
Application Message[PossDupFlag(43)=Y,MsgSeqNum(34)=5]
peer1
peer2

Figure 12 — Using ResendRequest(35=2) message to request missed message and subsequent messages

- Request only the specific messages missed while maintaining an ordered list of all newer messages.

  Using the above example, messages 3 to 5 would be saved for later processing and a ResendRequest(35=2) message would be sent only for message 2.

In both cases, messages 3 to 5 should not be processed before message 2.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

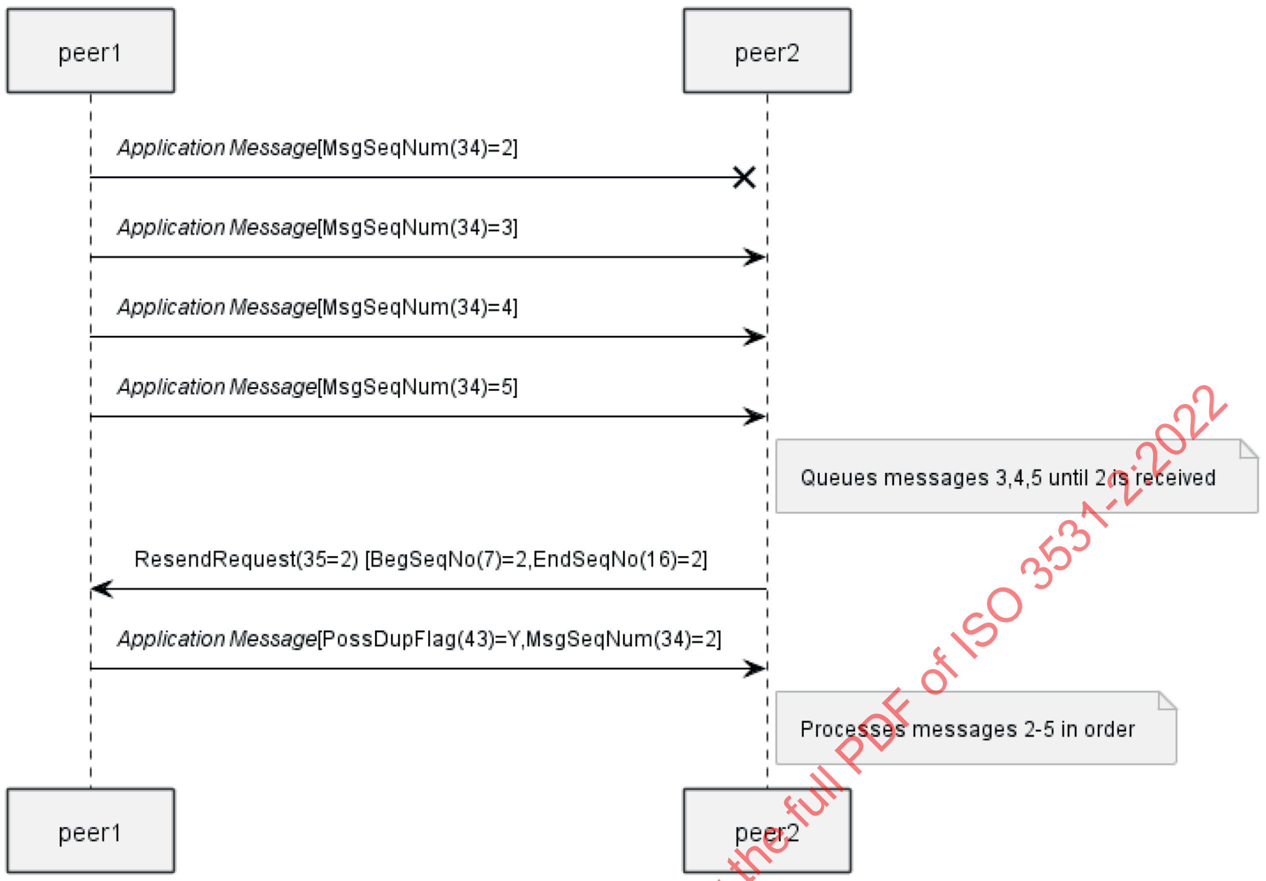

peer1
peer2
Application Message[MsgSeqNum(34)=2]
Application Message[MsgSeqNum(34)=3]
Application Message[MsgSeqNum(34)=4]
Application Message[MsgSeqNum(34)=5]
Queues messages 3,4,5 until 2 is received
ResendRequest(35=2) [BegSeqNo(7)=2,EndSeqNo(16)=2]
Application Message[PossDupFlag(43)=Y,MsgSeqNum(34)=2]
Processes messages 2-5 in order
peer1
peer2

Figure 13 — Using ResendRequest(35=2) message to request only the missed message

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

The ResendRequest(35 = 2) message with BeginSeqNo(7) and EndSeqNo(16) present should be sent to the peer FIX session processor to request missing messages when a gap is detected.

The ResendRequest(35 = 2) may be used to request retransmission of a single message, a range of messages or all messages subsequent to a particular message.

The receiving FIX session processor must process incoming messages in MsgSeqNum(34) order.

The ResendRequest(35 = 2) message may be used to specify which messages require retransmission in the following ways:

- To request a single message: Both BeginSeqNo(7) and EndSeqNo(16) are set the MsgSeqNum(34) of the missing message.
- To request a range of messages: BeginSeqNo(7) is set to the MsgSeqNum(34) of the first message of the range of missing messages and EndSeqNo(16) is set to the MsgSeqNum(34) of the last message of the range of missing messages.
- To request all messages subsequent to a particular missing message: BeginSeqNo(7) is set to the MsgSeqNum(34) of the first message of range and EndSeqNo(16) is set to 0. Zero is used to specify all messages after the message with MsgSeqNum(34)=BeginSeqNo(7)[11].

[11] Setting EndSeqNo(16) to 0 has been the recommended approach since FIX.4.2.

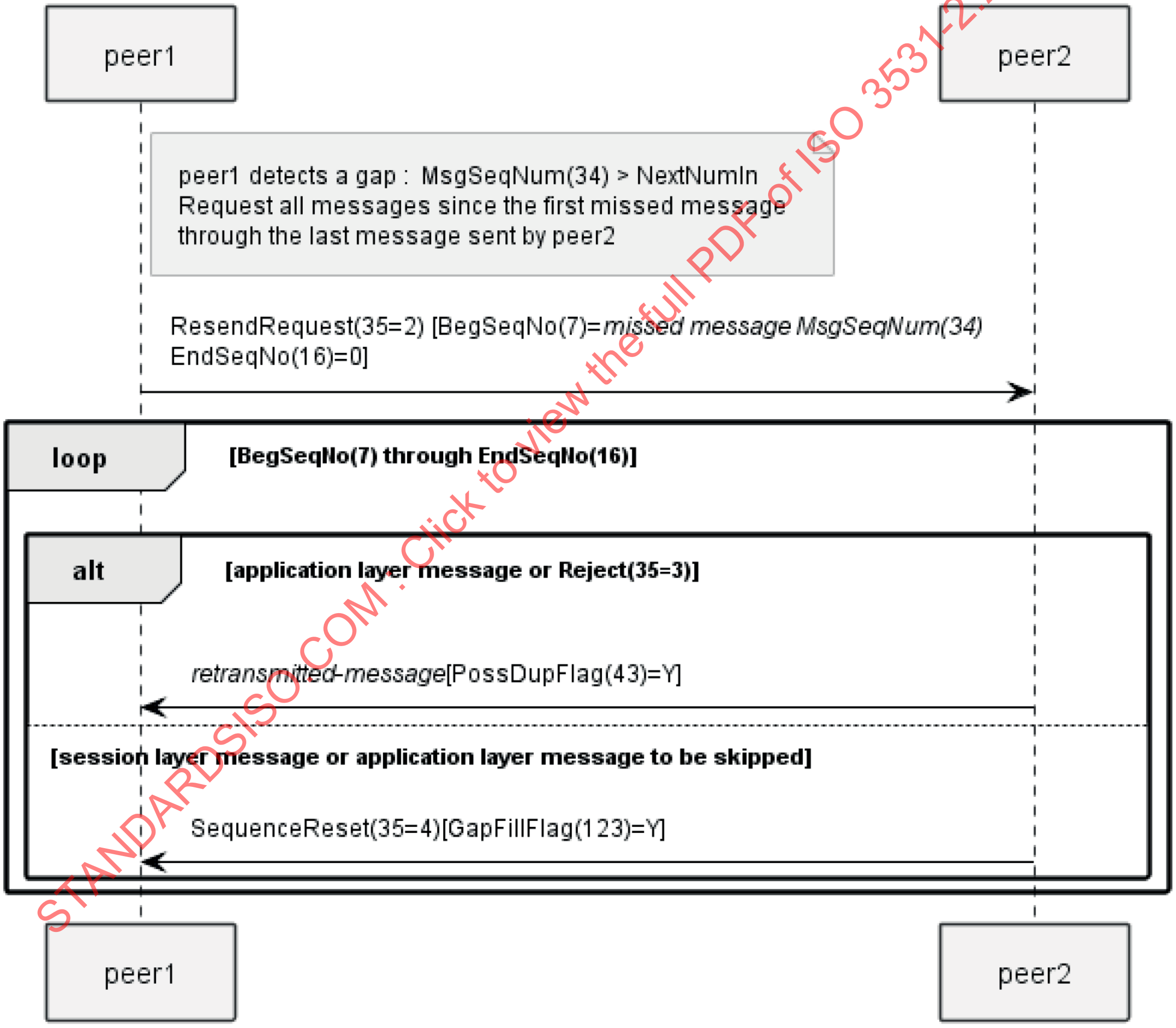
© ISO 2022 – All rights reserved

4.9.3 Responding to a ResendRequest(35 = 2) message

Upon receipt of a ResendRequest(35=2) message, the resender shall respond by either initiating a normal gap fill process or, in exceptional circumstances, by forcing a reset of the sequence number. In either case, the resender assumes responsibility for the results of the decision to retransmit or not retransmit messages.

The peer receiving the ResendRequest(35 = 2) may choose not to retransmit certain application messages. For instance, if orders or quotes require retransmission, the sender may choose not to retransmit them because too much time has elapsed, causing the orders to become stale or no longer acceptable due to a change in market state.

The SequenceReset(35 = 4) message with GapFillFlag(123) set to "Y" (Gap Fill) must be used to skip messages that a sender does not wish to retransmit in order to eliminate all gaps.



peer1
peer2
peer1 detects a gap : MsgSeqNum(34) > NextNumIn
Request all messages since the first missed message
through the last message sent by peer2
ResendRequest(35=2) [BegSeqNo(7)=missed message MsgSeqNum(34)
EndSeqNo(16)=0]
loop
[BegSeqNo(7) through EndSeqNo(16)]
alt
[application layer message or Reject(35=3)]
retransmitted-message[PossDupFlag(43)=Y]
[session layer message or application layer message to be skipped]
SequenceReset(35=4)[GapFillFlag(123)=Y]
peer1
peer2

Figure 14 — ResendRequest(35=2) with Gap Fill processing

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.9.4 Possible duplicates

When a FIX session processor is unable to determine if a message was successfully received at its intended destination or when responding to a ResendRequest(35=2) message, a possible duplicate message is generated. The message will be a retransmission (with the same sequence number) of the application data in question with the PossDupFlag(43) included and set to “Y”. It is the responsibility of the receiving firm to properly respond to the message with PossDupFlag(43) set to “Y” by determining if the message was previously processed.

Any message resent in response ResendRequest(35 = 2) message must contain the PossDupFlag(43) field set to “Y”.

Messages lacking the PossDupFlag(43) field or with the PossDupFlag(43) field set to “N” shall be treated as an original transmission.

The FIX session processor retransmitting a message with the PossDupFlag(43) set to “Y” must modify the following fields:

- SendingTime(52) set to the current sending time
- OrigSendingTime(122) set to the SendingTime(52) from the original message
- Recalculate the BodyLength(9)
- Recalculate the CheckSum(10)

If the message is encrypted, SecureDataLen(90) and SecureData(91) may also require re-encryption and re-encoding[12].

4.9.5 Gap fill process

The peer responding to a ResendRequest(35=2) message shall retransmit messages requested by the peer in message sequence number order, with the original sequence numbers and PossDupFlag(43) set to “Y”.

The resender shall not retransmit the following session layer message types:

- Logon(35=A)
- Logout(35=5)
- ResendRequest(35=2)
- HeartBeat(35=0)
- TestRequest(35=1)
- SequenceReset(35=4)

The responding peer shall send a SequenceReset(35=4) message with GapFillFlag(123) set to “Y” (Gap Fill) for each session layer or application layer message not retransmitted.

Reject(35 = 3) and XMLnonFIX(35 = n) are the only session messages which may be retransmitted.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

[12] The use of SecureData(91) and its associated length field SecureDataLen(90) were deprecated with the introduction of the FIXT.1.1 Session Layer specification. The use of application layer encryption is still supported by counterparty agreement.

© ISO 2022 – All rights reserved

The responding peer may choose, by counterparty agreement, not to retransmit all application messages. The definition of which application messages are retransmitted is defined in the application layer.

A continuous sequence of messages not being retransmitted should be skipped over using a single SequenceReset(35 = 4) message with GapFillFlag(123) set to “Y” and MsgSeqNum(34) set to the sequence number of the first skipped message and NewSeqNo(36) set to the next sequence number after the sequence number of the last message to be skipped (or “gap filled over”).

NewSeqNo(36) must always be set to the value of the next sequence number to be expected by the peer immediately following the messages being skipped.

The SequenceReset(35 = 4) message with GapFillFlag(123) set to “Y” (Gap Fill) may also be used to skip application messages that the sender chooses not to retransmit (for example, aged orders).

4.9.5.1 Example using SequenceReset(35 = 4) to gap fill over multiple messages

The following example is provided to show how SequenceReset(35=4) should be used to gap fill over messages in response to a ResendRequest(35=2).

In this example a ResendRequest(35 = 2) is received with BeginSeqNo(7) set to 5 and EndSeqNo(16) set to 0 (meaning all subsequent messages following 5). Messages with MsgSeqNum(34) equal to 8, 10, 11 are application messages. Messages 5-7 and 9 are session layer messages. Only messages 8, 10, and 11 should be retransmitted.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

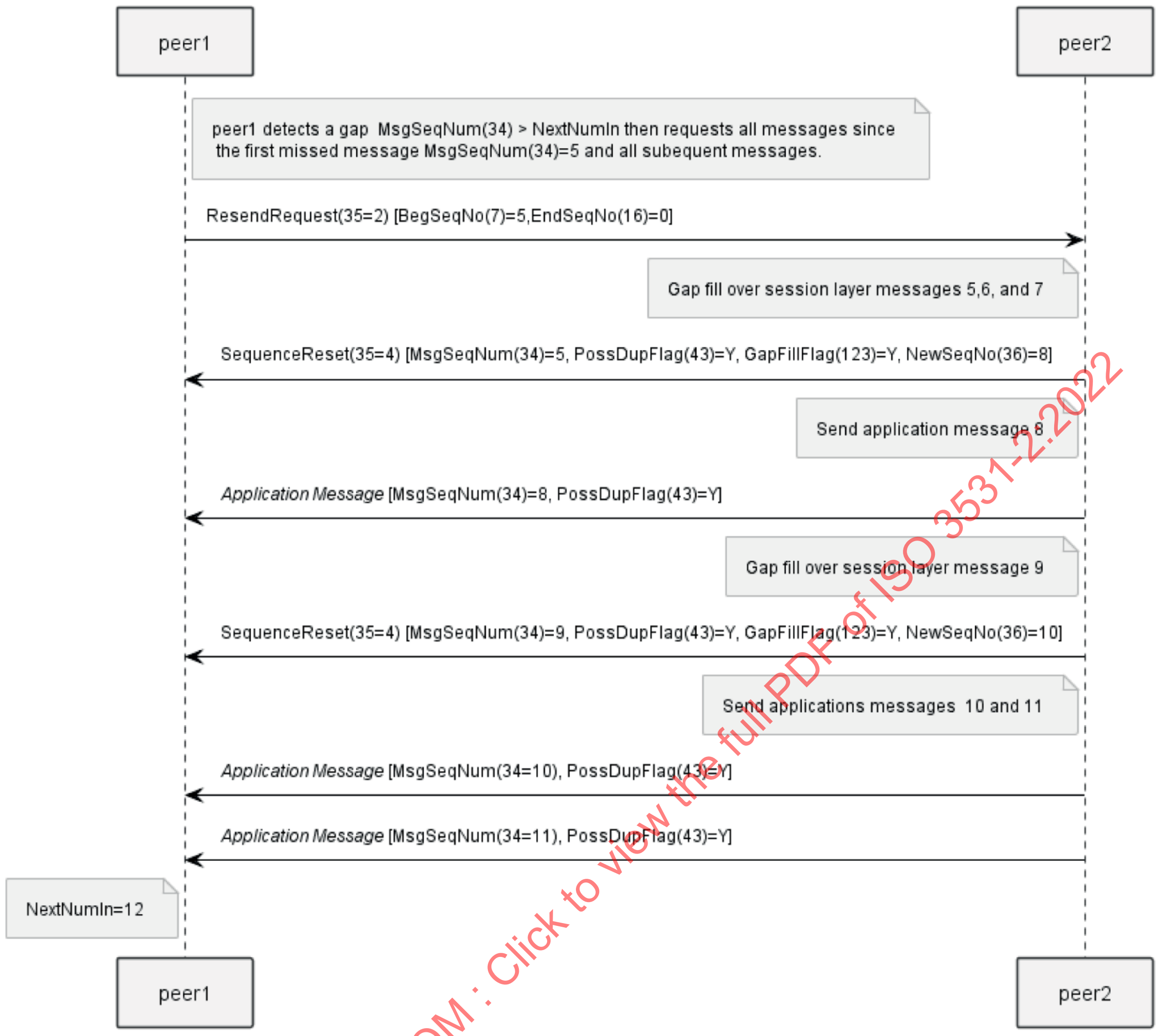

peer1
peer2
peer1 detects a gap MsgSeqNum(34) > NextNumIn then requests all messages since the first missed message MsgSeqNum(34)=5 and all subequent messages.
ResendRequest(35=2) [BegSeqNo(7)=5,EndSeqNo(16)=0]
Gap fill over session layer messages 5,6, and 7
SequenceReset(35=4) [MsgSeqNum(34)=5, PossDupFlag(43)=Y, GapFillFlag(123)=Y, NewSeqNo(36)=8]
Send application message 8
Application Message [MsgSeqNum(34)=8, PossDupFlag(43)=Y]
Gap fill over session layer message 9
SequenceReset(35=4) [MsgSeqNum(34)=9, PossDupFlag(43)=Y, GapFillFlag(123)=Y, NewSeqNo(36)=10]
Send applications messages 10 and 11
Application Message [MsgSeqNum(34=10), PossDupFlag(43)=Y]
Application Message [MsgSeqNum(34=11), PossDupFlag(43)=Y]
NextNumIn=12
peer1
peer2

Figure 15 — Example using the SequenceReset(35=4) to gap fill over multiple messages.

4.9.6 Sequence reset

A FIX session processor may send a SequenceReset(35=4) message with GapFillFlag(123) set to “N” (Reset) and PossDupFlag(43) set to “Y” to force sequence number synchronization in the peer.

The firm sending the SequenceReset(35 = 4) with GapFillFlag(123) set to “N” is responsible for the results that occur at the application layer caused by resetting the sequence number and not retransmitting messages requested to be resent by the peer.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

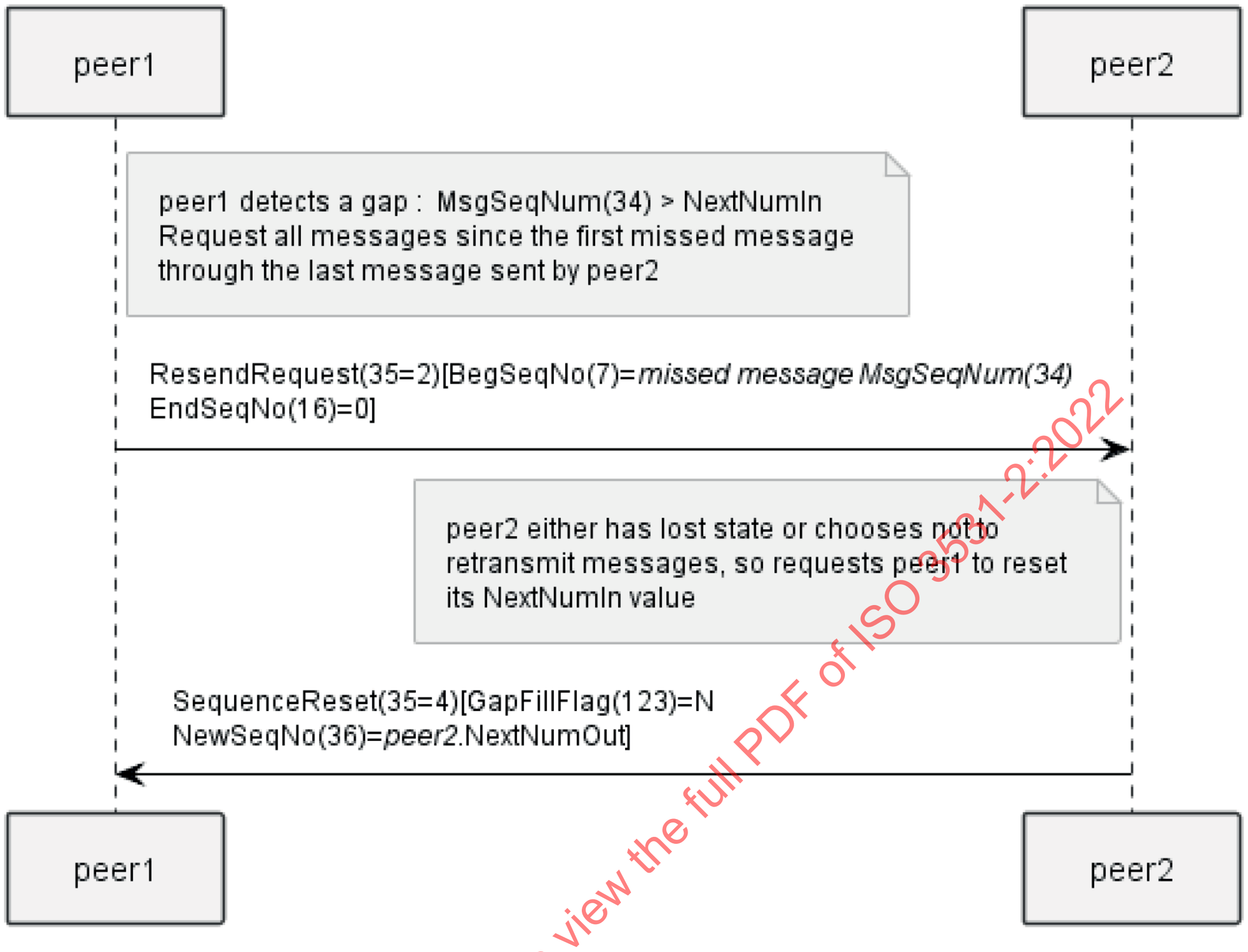

peer1
peer2
peer1 detects a gap : MsgSeqNum(34) > NextNumIn
Request all messages since the first missed message
through the last message sent by peer2
ResendRequest(35=2)[BegSeqNo(7)=missed message MsgSeqNum(34)
EndSeqNo(16)=0]
peer2 either has lost state or chooses not to
retransmit messages, so requests peer1 to reset
its NextNumIn value
SequenceReset(35=4)[GapFillFlag(123)=N
NewSeqNo(36)=peer2.NextNumOut]
peer1
peer2

Figure 16 — Performing a sequence reset instead of gap fill in response to a ResendRequest(35=2)

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.9.7 Processing inbound possible duplicate messages (PossDup(43) set to "Y")

All FIX implementations must monitor incoming messages to detect inadvertently retransmitted session layer messages (PossDupFlag(43) set to "Y", indicating a retransmission).

A FIX session processor, upon receipt of an inadvertently (or improperly) retransmitted session layer message as identified by the PossDupFlag(43) set to "Y", should perform sequence number processing (increment NextNumIn) only and avoid processing the session layer message.

A message received with the PossDupFlag(43) set to "Y" should be ignored when a message with the same MsgSeqNum(34) was previously processed.

4.9.8 Processing gaps when receiving FIX session layer messages

Gaps detected in message sequence numbers must be processed differently for certain FIX session layer messages. The table below lists the actions to be taken when the incoming FIX session layer message has a MsgSeqNum(34) that is greater than NextNumIn.

Table 3 — Response by message type

Message type received	Action to be taken on sequence number mismatch MsgSeqNum(34) > NextNumIn
Logon(35=A)	Must always be the first message transmitted. Authenticate and accept the connection. After sending a Logon(35=A) acknowledgement, send a ResendRequest(35=2) message.
Logout(35=5) request	Send ResendRequest(35=2) message requesting missing messages, receive and process retransmission of missing messages and gap fills, then send the Logout(35=5) acknowledgement message which serves as a confirmation of the Logout request. DO NOT terminate the session. The Logout(35=5) requestor should the terminate the transport layer connection. The only exception to the "do not terminate the session" rule is for an invalid Logon(35=5) request. The session acceptor has the right to send a Logout(35=5) message and terminate the transport layer connection immediately. This minimizes the threat of unauthorized connection attempts.
ResendRequest(35=2)	Perform the requested retransmission then send a ResendRequest(35=2) message requesting missing messages, receive and process the retransmission of missing messages and gap fills.
SequenceReset(35=4) (GapFillFlag(123)=N)	Reset: Ignore the incoming sequence number. The NewSeqNo(36) field of this message will contain the sequence number of the next message to be transmitted.
SequenceReset(35=4) (GapFillFlag(123)=Y)	Gap Fill: Send a ResendRequest(35=2). These messages behave similarly to SequenceReset(35=4) messages with GapFillFlag(123) set to "N". However, it is important to ensure that no messages have been inadvertently skipped over. This means that these messages must be received in sequence. An out of sequence SequenceReset(35=4) message with GapFillFlag(123) set to "Y" is an abnormal condition.
All other messages	Perform gap fill operations.

4.10 Resending an unacknowledged application message

The application layer is responsible for the detection and proper handling of duplicate application messages.

The FIX session layer does not initiate a resend of an application message nor does the FIX session layer detect duplication application messages.

The application layer may resend a message if it has not received an application layer acknowledgement for that message.

The amount of time allowed to elapse before resending the message may be specified by counterparty agreement in a rules of engagement.

The application layer may use the PossResend(97) set to "Y" field in the StandardHeader component along with other fields in an application message to identify duplication application messages.

The use or non-use of the PossResend(97) field when resending application messages should be documented in the rules of engagement.

The resent application layer message shall consume the next value of NextNumOut resulting in a new MsgSeqNum(34) for the application message.

The FIX session processor shall treat the resent message as a new message.

The FIX session processor must pass the PossResend(97) field to the application layer when it is present in an application message.

The use of a globally unique persistent identifier that enforces idempotent behaviour at the application layer may be used as an alternative to sending a PossResend(97) field by counterparty agreement.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

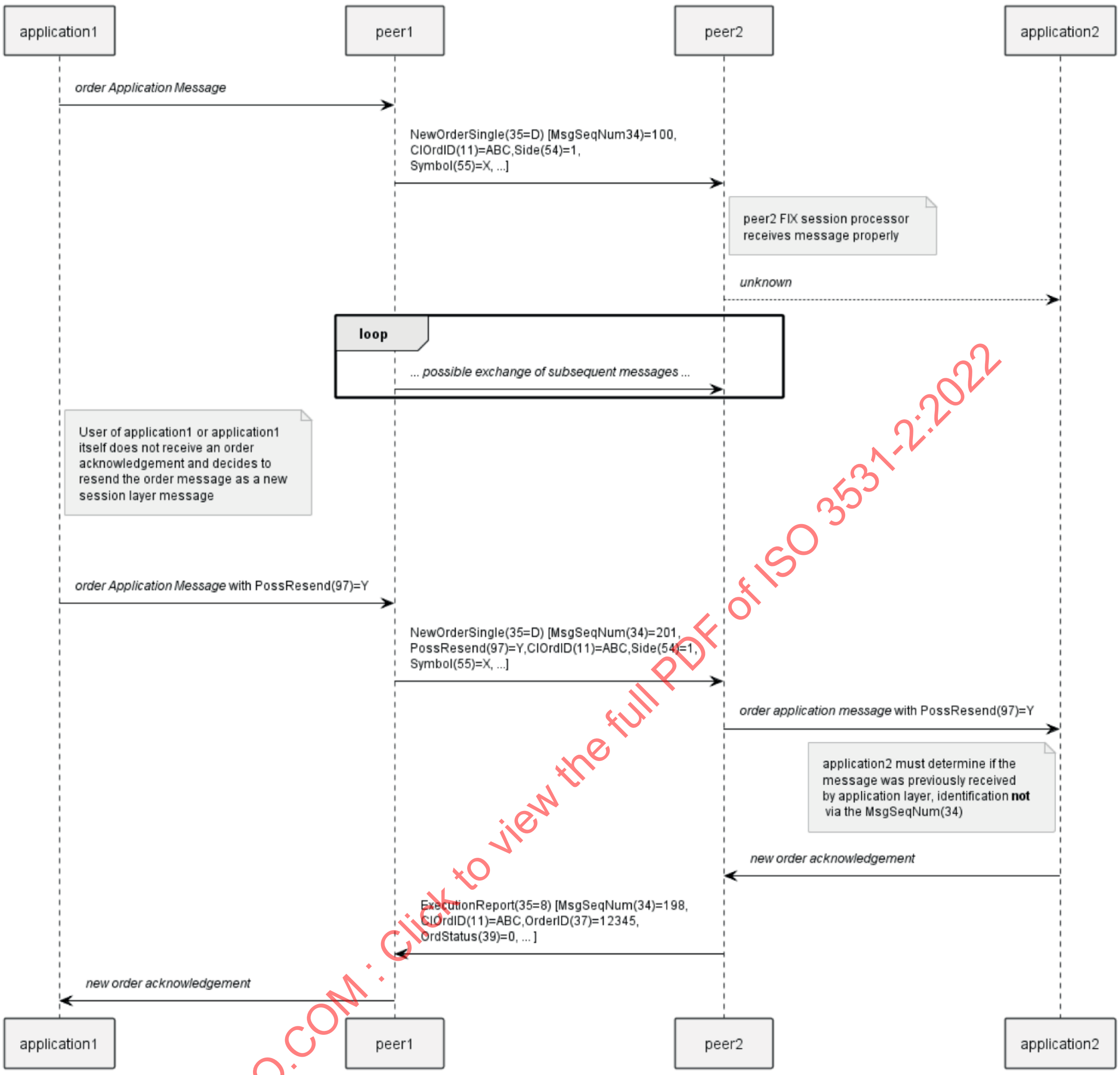

application1
peer1
peer2
application2
order Application Message
NewOrderSingle(35=D) [MsgSeqNum34)=100, ClOrdID(11)=ABC,Side(54)=1, Symbol(55)=X, ...]
peer2 FIX session processor receives message properly
unknown
loop
... possible exchange of subsequent messages ...
User of application1 or application1 itself does not receive an order acknowledgement and decides to resend the order message as a new session layer message
order Application Message with PossResend(97)=Y
NewOrderSingle(35=D) [MsgSeqNum(34)=201, PossResend(97)=Y,ClOrdID(11)=ABC,Side(54)=1, Symbol(55)=X, ...]
order application message with PossResend(97)=Y
application2 must determine if the message was previously received by application layer, identification not via the MsgSeqNum(34)
new order acknowledgement
ExecutionReport(35=8) [MsgSeqNum(34)=198, ClOrdID(11)=ABC,OrderID(37)=12345, OrdStatus(39)=0, ...]
new order acknowledgement
application1
peer1
peer2
application2

Figure 17 — Application layer resend using PossResend(97)

4.10.1 The difference between application layer resend and session layer retransmission

The purpose of the following diagram is to visually show the difference between an application using the PossResend(97) value of a previously unacknowledged application message and a FIX session processor retransmitting messages that may not have been received by its peer as a response to a ResendRequest(35=2) message from that peer.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

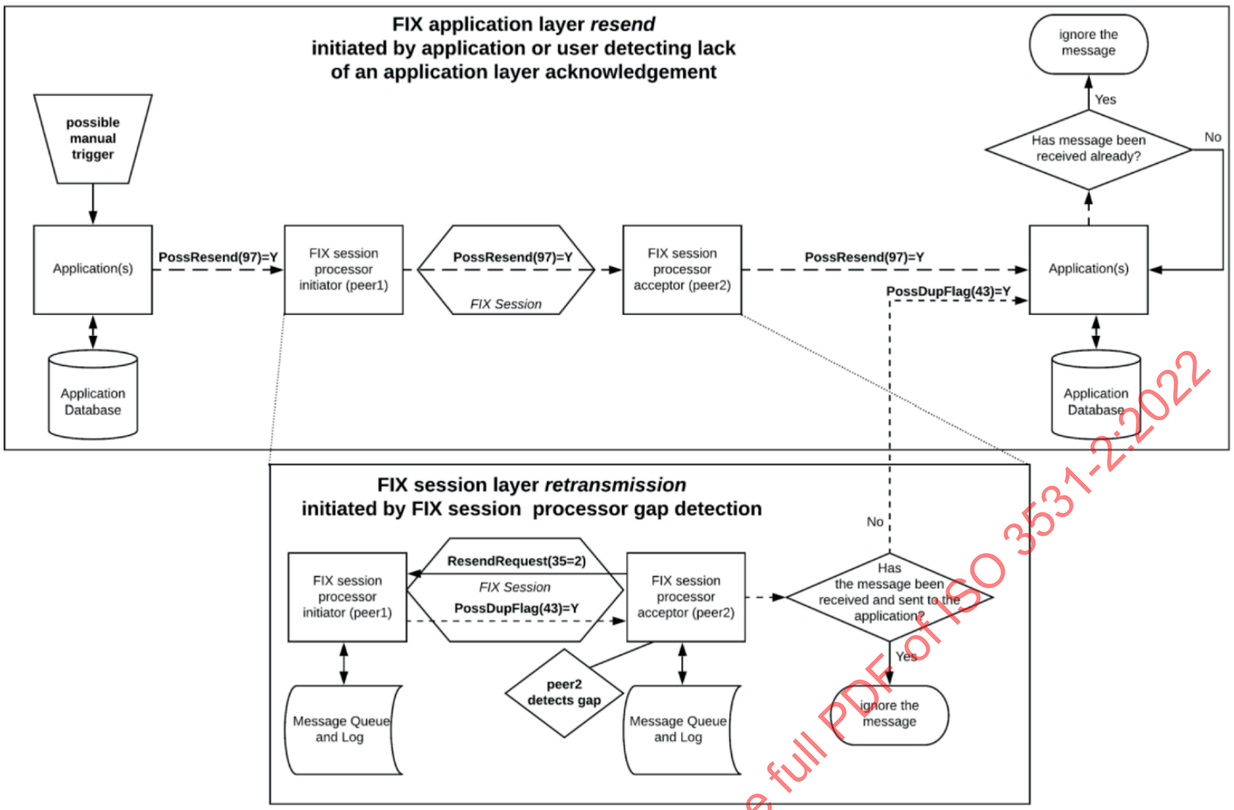

FIX application layer *resend* initiated by application or user detecting lack of an application layer acknowledgement
possible manual trigger
Application(s)
Application Database
PossResend(97)=Y
FIX session processor initiator (peer1)
PossResend(97)=Y
FIX Session
FIX session processor acceptor (peer2)
PossResend(97)=Y
PossDupFlag(43)=Y
Application(s)
Application Database
ignore the message
Yes
Has message been received already?
No
FIX session layer *retransmission* initiated by FIX session processor gap detection
FIX session processor initiator (peer1)
ResendRequest(35=2)
FIX Session
PossDupFlag(43)=Y
FIX session processor acceptor (peer2)
Message Queue and Log
peer2 detects gap
Message Queue and Log
No
Has the message been received and sent to the application?
Yes
ignore the message

Figure 18 — The difference between application layer resend and session layer retransmission

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

4.11 FIX session state matrix

Precedence	State	Initiator	Acceptor	Description
1	Disconnected – No Connection Today	Y	Y	Currently disconnected, have not attempted to establish a connection “today”, and no MsgSeqNum(34) has been consumed (next connection “today” will start at MsgSeqNum(34) of 1).
2	Disconnected – Connection Today	Y	Y	Currently disconnected, have attempted to establish a connection “today” and thus MsgSeqNum(34)has been consumed (next connection “today” will start at MsgSeqNum(34) of (last + 1)).
3	Detect Broken Network Connection	Y	Y	While connected, detect a broken network connection (e.g. TCP socket closed). Disconnect the network connection and “shutdown” configuration for this session.
4	Awaiting Connection	N	Y	Session acceptor Logon awaiting network connection from counterparty.
5	Initiate Connection	Y	N	Session initiator Logon establishing network connection with counterparty.
6	Network Connection Established	Y	Y	Network connection established between both parties.

© ISO 2022 – All rights reserved

Precedence	State	Initiator	Acceptor	Description
7	Initiation Logon(35=A) Sent	Y	N	Session initiator Logon send Logon(35=A) message. *** Exception: 24hr sessions.
8	Initiation Logon(35=A) Received	N	Y	Session acceptor Logon receive counterparty's Logon(35=A) message. *** Exception: 24hr sessions.
9	Initiation Logon(35=A) Response	N	Y	Session acceptor Logon respond to counterparty's Logon(35=A) message with Logon(35=A) message to handshake.
10	Handle ResendRequest(35=2)	Y	Y	Receive and respond to counterparty's ResendRequest(35=2) sending requested messages and/or SequenceReset(35=4) Gap Fill messages for the range of MsgSeqNum(34) requested. Updated to include rejecting ResendRequest(35=2) received with MsgSeqNum(34) that is <= LastSeqNum processed.
11	Receive MsgSeqNum(34) Too High	Y	Y	Receive too high of MsgSeqNum(34) from counterparty, queue message, and send ResendRequest(35=2).
12	Awaiting/ Processing Response to ResendRequest(35=2)	Y	Y	Process requested MsgSeqNum(34) with PossDupFlag(43)=Y resent messages and/or SequenceReset(35=4) Gap Fill messages from counterparty. Queue incoming messages with MsgSeqNum(34) too high.
13	No messages received in Interval	Y	Y	No inbound messages (non-garbled) received in (HeartBtInt(108) + "reasonable period of time"), send TestRequest(35=1).
14	Awaiting/ Processing Response to TestRequest(35=1)	Y	Y	Process inbound messages. Reset heartbeat interval-related timer when ANY inbound message (non-garbled) is received.
15	Receive Logout(35=5) message	Y	Y	Receive Logout(35=5) message from counterparty initiating logout/disconnect. If MsgSeqNum(34) too high, send ResendRequest(35=2). If sent, wait a reasonable period of time for complete response to ResendRequest(35=2). Note that depending upon the reason for the Logout(35=5), the counterparty may be unable to fulfill the request. Send Logout(35=5) message as response and wait a reasonable period of time for counterparty to disconnect the network connection. Note that counterparty may send a ResendRequest(35=2) message if Logout(35=5) message response has MsgSeqNum(34) too high and then re-initiate the Logout(35=5) process.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Precedence	State	Initiator	Acceptor	Description
16	Initiate Logout(35=5) Process	Y	Y	Identify condition or reason to gracefully disconnect (e.g. end of "day", no response after multiple TestRequest(35=1) messages, too low MsgSeqNum(34), etc.). Send Logout(35=5) message to counterparty. Wait a reasonable period of time for Logout(35=5) response. During this time, handle "new" inbound messages and/or ResendRequest(35=2) messages if possible. Note that some logout/termination conditions (e.g. loss of database/message safe-store) may require immediate termination of the network connection following the initial send of the Logout(35=5) message. Disconnect the network connection and "shutdown" configuration for this session.
17	Active/Normal Session	Y	Y	Network connection established, Logon(35=A) message exchange successfully completed, inbound and outbound MsgSeqNum(34) are in sequence as expected, and HeartBeat(35=0) or other messages are received within (HeartBtInt(108) + "reasonable period of time").
18	Waiting for a Logon(35=A)	Y	N	Session initiator waiting for session acceptor to send back a Logon(35=A) acknowledgement.

4.11.1 FIX logon process state transition diagram

Session Initiator (e.g. buyside) Action	Session Acceptor (e.g. sellside) Action	Session Initiator (e.g. buyside) State	Session Acceptor (e.g. sellside) State
Start		Disconnected – No Connection Today	Awaiting Connection
		Disconnected – Connection Today	
Connect		Initiate Connection	Awaiting Connection
		(Possible) Detect Broken Network Connection	
	Accept Connection	Network Connection Established	Network Connection Established
Initiate Logon(35=A)		Initiation Logon(35=A) Sent	Network Connection Established
	Receive Initiation Logon(35=A)	Initiation Logon(35=A) Sent	Initiation Logon(35=A) Received
	Send Initiation Logon(35=A) Response	Initiation Logon(35=A) Sent	Initiation Logon(35=A) Response

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

Session Initiator (e.g. buyside) Action	Session Acceptor (e.g. sellside) Action	Session Initiator (e.g. buyside) State	Session Acceptor (e.g. sellside) State
			(Possible) Initiate Logout(35=5) Process (e.g. if MsgSeqNum(34) too low) (Possible) Receive MsgSeqNum(34) Too High
	(Possible) Send ResendRequest(35=2)		Initiation Logon(35=A) Response (Possible) Receive MsgSeqNum(34) Too High
Receive Initiation Logon(35=A) Response		(Possible) Active/Normal Session (Possible) Initiate Logout(35=5) Process (e.g. if MsgSeqNum(34) too low)	Initiation Logon(35=A) Response
(Possible) Send ResendRequest(35=2)		(Possible) Active/Normal Session (Possible) Receive MsgSeqNum(34) Too High	(Possible) Active/Normal Session (Possible) Handle ResendRequest(35=2)
		Active/Normal Session	Active/Normal Session

4.11.2 FIX logout process state transition diagram

Logout Initiator Action	Logout Acceptor Action	Logout Initiator State	Logout Acceptor State
Start		Active/ Normal Session	Active/Normal Session
		No Messages Received in Interval	No Messages Received in Interval
		Awaiting/ Processing Response to TestRequest(35=1)	- Initiation Logon(35=A) Sent - Awaiting/Processing Response to TestRequest(35=1) - Awaiting validation of logon - Receive MsgSeqNum(34) Too High - Awaiting/Processing Response to ResendRequest(35=2) - Initiate Logout(35=5) Process - Waiting for a Logon(35=A) acknowledgement

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Logout Initiator Action	Logout Acceptor Action	Logout Initiator State	Logout Acceptor State
Send Logout(35=5) Request		Logout Pending	
	Receive Logout(35=5) Request	Logout Pending	Logout Pending (Possible) Receive MsgSeqNum(34) Too High
	Send Logout(35=5) Response	Logout Pending	Awaiting Disconnect
	(Possible) Send ResendRequest(35=2)	Logout Pending	(Possible) Awaiting / Processing Response to ResendRequest(35=2)
(Possible) Receive ResendRequest(35=2)		(Possible) Awaiting / Processing Response to ResendRequest(35=2)	(Possible) Awaiting Response to ResendRequest(35=2)
Receive Logout(35=5) Response		Disconnected – Connection Today	Awaiting Disconnect
Disconnect		Disconnected – Connection Today	Disconnected – Connection Today

5 FIX session profiles

The following session profiles are defined within the standard. Each session profile defines a standardized use of the FIX session layer for a specific context primarily related to application version usage.

5.1 FIX.4.2 session profile

FIX.4.2 is the FIX session profile that is used with the FIX 4.2 application layer. The FIX 4.2 application layer may use fields from subsequent versions of FIX.

The FIX.4.2 session profile may use extended features from the FIX session standard by counterparty agreement.

5.1.1 Profile identification

The value of BeginString(8) must be "FIX.4.2"

5.1.2 Application version identification

Field	Definition
DefaultApplVerID(1137)	Not used with FIX.4.2 session profile as FIX 4.2 is the mandatory application layer.
DefaultApplExtID(1407)	May be used by counterparty agreement in-band. Recommendation is to communicate this information out-of-band via a rules of engagement, possibly by using a FIX Orchestra rules of engagement.
DefaultCstmApplVerID(1408)	May be used by counterparty agreement in-band on the Logon(35=A) message to identify the version of the custom application or FIX Orchestra rules of engagement.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

5.2 FIX4 session profile

FIX4 is fully backward compatible with the FIX 4.4 session layer as defined in Volume 2 of the FIX 4.4 specification.

FIX4 adopters may use extended capabilities beyond the basic FIX 4.4 functionality by counterparty agreement.

The FIX4 session profile supports the communication of a single version of the FIX application layer.

5.2.1 Profile identification

The value of BeginString(8) must be "FIX.4.4"

5.2.2 Application version identification

The default FIX application version is *FIX Latest*.

Field	Definition
DefaultApplVerID(1137)	DefaultApplVerID(1137) may be used to specify the application version in the Logon(35=A) message. If the DefaultApplVerID(1137) is not specified the default will be assumed to be *FIX Latest*.
DegaultApplExtID(1407)	May be used by counterparty agreement in-band. Recommendation is to communicate this information out-of-band via a rules of engagement possibly by using a FIX Orchestra rules of engagement.
DefaultCstmApplVerID(1408)	May be used by counterparty agreement in-band on the Logon(35=A) message to identify the custom application or FIX Orchestra rules of engagement.

5.3 FIXT session profile

FIXT supports transmitting messages from multiple application versions over the same session layer. This capability is the primary differentiator between the FIX4 session profile and FIXT session profile. FIXT may be used with a single application version, such as *FIX Latest*. The benefit in using FIXT for a single application version is it provides an option to include additional application versions at the individual message level at a future time.

5.3.1 Profile identification

The value of BeginString(8) must be "FIXT.1.1".

5.3.2 Multiple application version support over a single FIXT session

The FIXT session profile (along with the LFIXT session profile based upon FIXT) permit application messages from different application versions of FIX to be used over a single FIX session. The benefit is to broaden service offerings, perform incremental enhancements at the individual message or message category level.

5.3.3 Session default application version identification

The session default application version must be present in DefaultApplVerID(1137) in the Logon(35=A) message.

The application version may be overridden by specifying the application version explicitly by specifying the ApplVerID(1128) in the StandardHeader component of any application message by counterparty agreement.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Field	Definition
DefaultApplVerID(1137)	DefaultApplVerID(1137) must be specified on the Logon(35=A) message.
DegaultApplExtID(1407)	May be used by counterparty agreement in-band. Recommendation is to communicate this information out-of-band via a rules of engagement possibly by using a FIX Orchestra rules of engagement.
DefaultCstmApplVerID(1408)	May be used by counterparty agreement in-band on the Logon(35=A) message to identify the custom application or FIX Orchestra rules of engagement.

5.3.4 Message type default application version

FIXT expands upon the ability to specify an application version on the Logon(35=A) message by allowing different application versions to be specified by message type in the MsgTypeGrp component.

The MsgTypeGrp component may be used to specify what application messages are supported over the FIX session being initiated.

Within the MsgTypeGrp component, a FIX session may specify a message type default application version on the Logon(35 = A) message. For each message type where the default application version is different from the session default application version, the message type default application version is specified using the RefMsgType(372) field to specify the message type, the RefApplVerID(1130) field to specify the application version, and the DefaultVerIndicator(1410) field to indicate if RefApplVerID(1130) is the default.

When present, the RefApplVerID(1130) in the RefMsgTypeGrp component has precedence over the session default application version specified in the DefaultApplVerID(1137) field of the Logon(35 = A) message.

5.3.5 Explicit application version per message

The application version may be specified on a message instance using the ApplVerID(1128) field from the standard header.

The explicit application version only applies to the message instance in which it is sent, default application versions remain unaltered.

The explicit application version has precedence over the message type default application version and the session default application version.

FIX session processors must maintain state information regarding the default application version used during the FIX session.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Table 4 — Application version precedence

Application version	Fields	Precedence
Session default	DefaultApplVerID(1137)	Lowest level of precedence.
Message type default	RefMsgType(372), RefApplVerID(1130), DefaultVerIndicator(1410)	Supersedes the session default application version.
Explicit	ApplVerID(1128)	Highest precedence over default application versions. Only applies to the message instance in which ApplVerID(1128) is present.

5.3.6 Use of extension packs

A FIX session may specify a default extension pack as part of the session default application version by inclusion of a valid extension pack number in the DefaultApplExtID(1407).

An extension pack can be specified for a specific message type using the RefApplExtID(1406) field within the MsgTypeGrp component. If provided, RefApplExtID(1406) becomes part of the message type default application version.

A message instance can explicitly include an extension pack in the ApplExtID(1156) field. If ApplExtID(1156) is specified it becomes part of the explicit application version.

An extension pack is considered incompatible with the application version if the extension pack number specified is less than the last extension pack that was used to create the application version.

Table 5 — Extension pack precedence

EP version	Fields	Precedence
Session default	DefaultApplExtID(1407)	Lowest level of precedence.
Message type default	RefMsgType(372), RefApplExtID(1406), DefaultVerIndicator(1410)	Supersedes the session default extension pack.
Explicit	ApplExtID(1156)	Highest precedence over default extension pack. Only applies to the message instance in which ApplExtID(1156) is present.

5.3.7 Use of a custom application version

A default custom application version may be specified in-band over a FIX session by inclusion of a user defined value in DefaultCstmApplVerID(1408) on the Logon(35=A) message.

A custom application version can be specified for a specific message type using the RefCStmApplVerID(1131) field within the MsgTypeGrp component. If provided, RefCStmApplVerID(1131) becomes part of the message type default application version.

A message instance can explicitly include a custom application version in the CstmApplVerID(1129) field. If CstmApplVerID(1129) is specified it becomes part of the explicit application version.

Table 6 — Custom application version precedence

EP version	Fields	Precedence
Session default	DefaultCstmApplVerID(1408)	Lowest level of precedence.
Message type default	RefMsgType(372), RefCStmApplVerID(1131), DefaultVerIndicator(1410)	Supersedes the session default custom application version.
Explicit	CstmApplVerID(1129)	Highest precedence over default custom application versions. Only applies to the message instance in which the CstmApplVerID(1129) is present.

5.4 Lightweight FIXT (LFIXT session profile)

The Lightweight FIXT session profile (LFIXT) is a restriction of the FIXT session profile. LFIXT eliminates the use of session layer retransmission of messages to simplify the session layer protocol.

LFIXT defines two different modes of operation. Under *LFIXT compatible mode*, LFIXT peer is interoperable with existing FIX session processors that support the FIXT session profile. Under *LFIXT succinct mode*, LFIXT peer is no longer interoperable with FIXT peer, but the complexity of the session layer protocol is cut down further.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

5.4.1 Profile identification

The value of BeginString(8) must be “FIXT.1.1”.

LFIXT shares the BeginString(8) value with the FIXT session profile in order to support its communication with the FIXT peer under the *LFIXT compatible mode*.

5.4.2 Application version identification

The LFIXT session profile follows the same application version identification rules as the FIXT session profile.

5.4.3 LFIXT transport layer requirements

LFIXT does not support retransmission of messages at the session layer and relies on the application layer to provide retransmission services during the FIX session when needed.

For LFIXT a FIX session exists over one FIX connection which exists over one transport layer. A loss of either the FIX connection or the transport layer connection constitutes the end of a FIX session.

Each establishment of a FIX connection creates a new FIX session, which means that NextNumIn is set to 1 and the NextNumOut is set to 1 in the FIX session processor.

5.4.4 LFIXT compatible mode

When operating under *LFIXT compatible mode*, a FIXT peer shall handle all possible types of FIXT inbound session layer messages, but only send out session layer messages of limited types to enable a seamless communication between the LFIXT peer and the FIXT peer. An LFIXT peer is compatible with its FIXT peer in this sense.

Similarly, two LFIXT peers have the ability to communicate with each other seamlessly when both operating under the *LFIXT compatible mode*.

FIXT peer
LFIXT peer

Figure 19 — FIXT peer connected to LFIXT peer operating under compatible mode

Table 7 — Session layer messages when peer uses LFIXT session profile under compatible mode

Session Layer Message	Received from peer	Sent to peer
HeartBeat(35=0)	Yes	Yes
Logon(35=A)	Yes	Yes
TestRequest(35=1)	Yes	No
ResendRequest(35=4)	Yes	No
Reject(35=3)	Yes	Yes
SequenceReset(35=4) GapFillFlag(123)=N	Yes	Yes, but only passively
SequenceResest(35=4) GapFillFlag(123)=Y	Yes	No
Logout(35=5)	Yes	Yes

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

5.4.5 LFIXT succinct mode

When operating in *LFIXT succinct mode*, an LFIXT peer shall only accept a subset of inbound session layer message types and shall only transmit a subset of session layer message types. The complexity of the session layer protocol is reduced to a minimum in this way, while sacrificing the compatibility with FIXT peers.

Upon receiving an unexpected type of session layer message, an LFIXT peer operating under the *LFIXT succinct mode* shall terminate the connection by sending a Logout(35 = 5) with a description of the error in Text(58) and then terminate the transport layer connection.

Succinct mode shall only be used by an LFIXT peer when its counterparty is already known to be using the LFIXT session profile, e.g. by prior out-of-band exchange of the rules of engagement[13]. Its counterparty can operate under *LFIXT compatible mode* or under *LFIXT succinct mode*, but cannot be a FIXT peer (using the FIXT session profile) since the two peers are not compatible in this case.

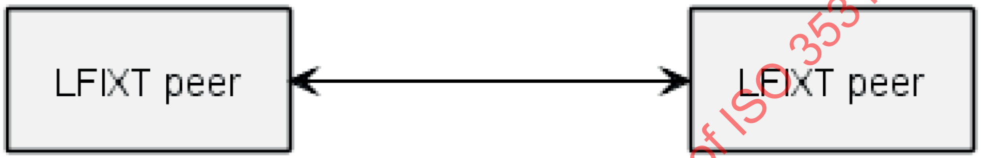


Figure 20 — LFIXT peer connected to LFIXT peer operating under succinct mode

Table 8 — Session layer messages when peer uses LFIXT session profile under succinct mode

Session Layer Message	Received from peer	Sent to peer
HeartBeat(35=0)	Yes	Yes
Logon(35=A)	Yes	Yes
Reject(35=3)	Yes	Yes
Logout(35=5)	Yes	Yes

5.4.6 LFIXT and FIXT operating mode interoperability

The following table shows the interoperability between LFIXT and FIXT peers for each operating mode of LFIXT.

Table 9 — Interoperability between LFIXT and FIXT peers by operating mode

Operating Mode	LFIXT acceptor using succinct mode	LFIXT acceptor using compatible mode	FIXT acceptor
LFIXT initiator using succinct mode	interoperable	interoperable	not interoperable
LFIXT initiator using compatible mode	interoperable	interoperable	interoperable
FIXT initiator	not interoperable	interoperable	interoperable

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

[13] FIX Orchestra may be used to communicate which FIX session profile and which operating mode is being used for a FIX enabled service.

5.4.7 Validation of message sequence numbers

If a received message has a MsgSeqNum(34) less than NextNumIn, a serious error exists. The FIX session and transport layer connection should be terminated unless the message received is:

- a SequenceReset(35=4) message with PossDupFlag(43) present with a value of "Y".
- any message of a message type that supports retransmission with a PossDupFlag(43) present with a value of "Y".
- a Logon(35=A) message with ResetSeqNumFlag(141) present with a value of "Y".

If a received message has a MsgSeqNum(34) greater than NextNumIn, a serious error exists. The FIX session and transport layer connection should be terminated.

5.4.8 Application layer recovery

The application layer shall be responsible for message recovery.

An LFIXT session processor must not perform message recovery.

5.4.9 LFIXT initiator connects to LFIXT acceptor

An LFIXT initiator shall send a Logon(35=A) message with the following fields present

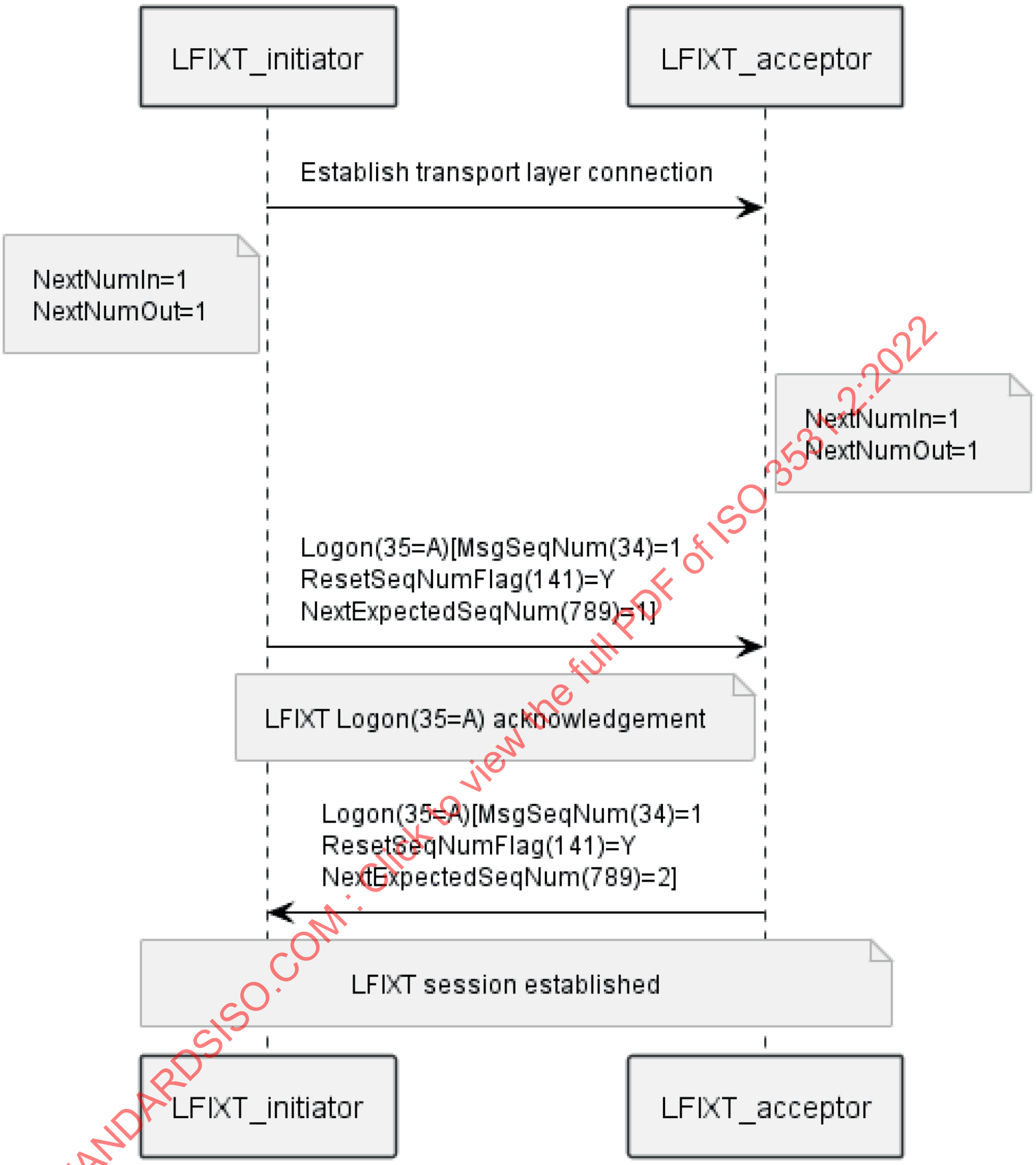
Table 10 — LFIXT Logon(35=A) initiator

Field	Value
MsgSeqNum(34)	1
ResetSeqNumFlag(141)	Y
NextExpectedSeqNum(789)	1

In the following diagram shows the standard LFIXT logon process.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved


LFIXT_initiator
LFIXT_acceptor
Establish transport layer connection
NextNumIn=1
NextNumOut=1
NextNumIn=1
NextNumOut=1
Logon(35=A)[MsgSeqNum(34)=1
ResetSeqNumFlag(141)=Y
NextExpectedSeqNum(789)=1]
LFIXT Logon(35=A) acknowledgement
Logon(35=A)[MsgSeqNum(34)=1
ResetSeqNumFlag(141)=Y
NextExpectedSeqNum(789)=2]
LFIXT session established
LFIXT_initiator
LFIXT_acceptor

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 21 — LFIXT logon process

5.4.10 FIXT initiator connects to LFIXT acceptor (compatible mode)

A FIXT initiator that is able to persist message history shall send a Logon(35=A) message with the following fields present. Upon receiving the Logon(35=A) request, an LFIXT acceptor shall adjust its own NextNumOut to the NextExpectedMsgSeqNum(789) contained in the Logon(35=A) request.

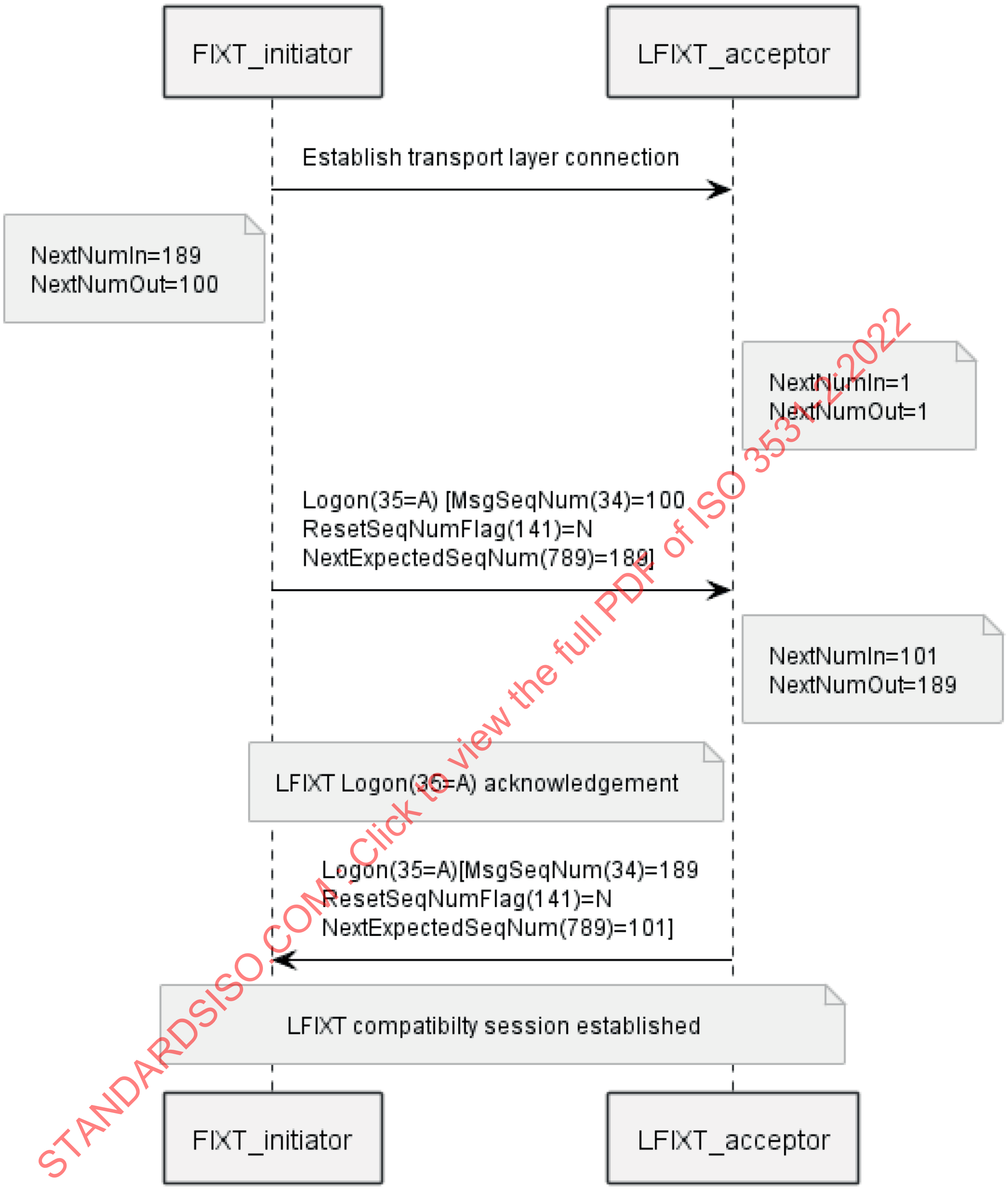
Table 11 — FIXT Logon(35=A) initiator

Field	Value
MsgSeqNum(34)	1
ResetSeqNumFlag(141)	N
NextExpectedSeqNum(789)	NextNumIn

In the following example, a FIXT initiator is establishing a new FIX connection to the LFIXT session. This example shows how the LFIXT acceptor adjusts its NextNumIn and NextNumOut values to the FIXT session state being maintained by a FIXT initiator.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved


FIXT_initiator
LFIXT_acceptor
Establish transport layer connection
NextNumIn=189
NextNumOut=100
NextNumIn=1
NextNumOut=1
Logon(35=A) [MsgSeqNum(34)=100
ResetSeqNumFlag(141)=N
NextExpectedSeqNum(789)=189]
NextNumIn=101
NextNumOut=189
LFIXT Logon(35=A) acknowledgement
Logon(35=A)[MsgSeqNum(34)=189
ResetSeqNumFlag(141)=N
NextExpectedSeqNum(789)=101]
LFIXT compatibilty session established
FIXT_initiator
LFIXT_acceptor

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Figure 22 — FIXT initiator connecting to LFIXT acceptor in compatible mode logon

In the following example, an LFIXT initiator is establishing a new FIX connection to the FIXT session. This example shows how the FIXT acceptor resets its NextNumIn and NextNumOut values when the LFIXT initiator sends the Logon(35 = A) request.

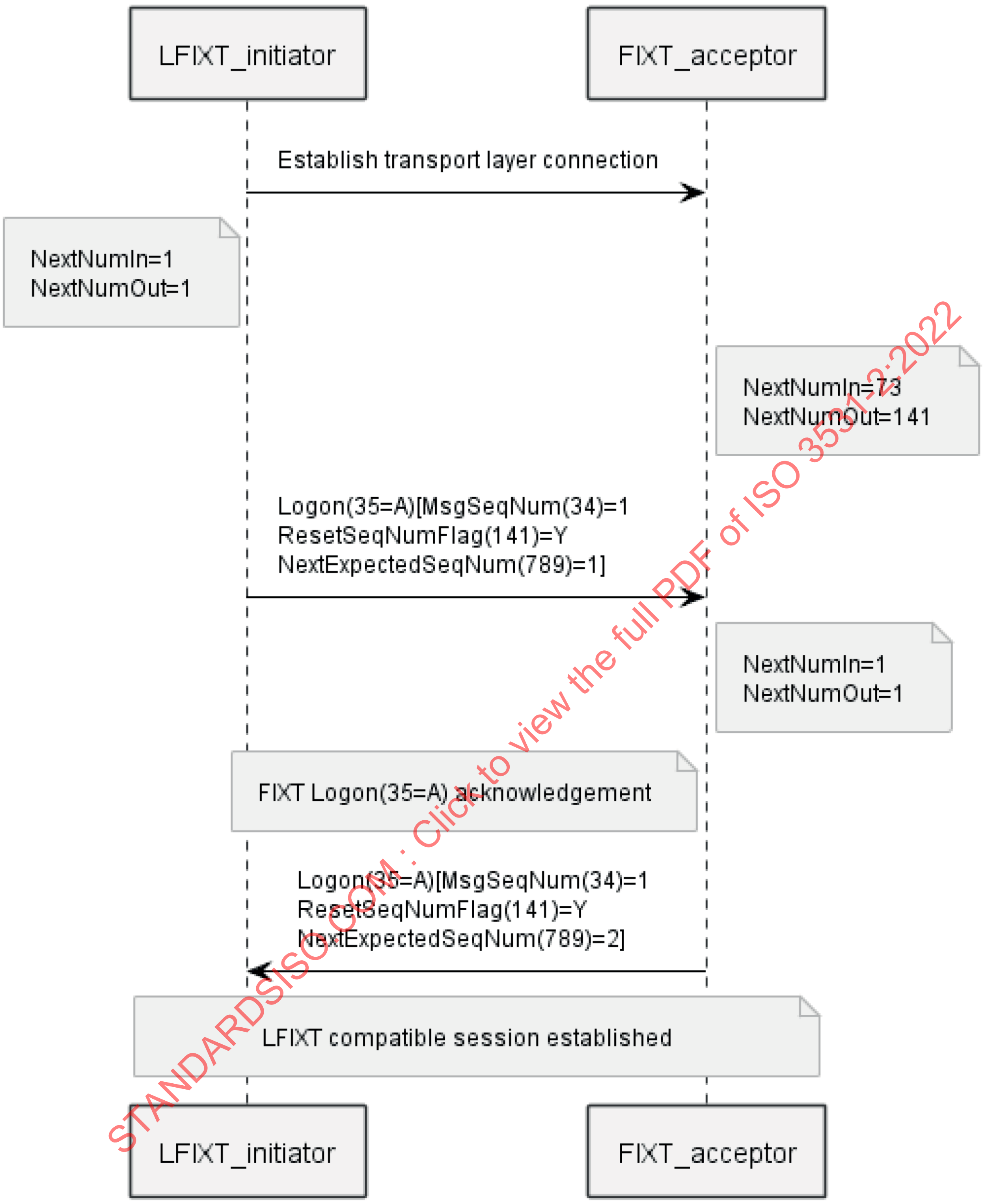

LFIXT_initiator
FIXT_acceptor
Establish transport layer connection
NextNumIn=1
NextNumOut=1
NextNumIn=73
NextNumOut=141
Logon(35=A)[MsgSeqNum(34)=1
ResetSeqNumFlag(141)=Y
NextExpectedSeqNum(789)=1]
NextNumIn=1
NextNumOut=1
FIXT Logon(35=A) acknowledgement
Logon(35=A)[MsgSeqNum(34)=1
ResetSeqNumFlag(141)=Y
NextExpectedSeqNum(789)=2]
LFIXT compatible session established
LFIXT_initiator
FIXT_acceptor

Figure 23 — LFIXT initiator connecting to FIXT acceptor in compatible mode logon

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

5.4.11 Receipt of ResendRequest(35 = 2) message from a FIXT peer

LFIXT peers are not expected to save outbound message history for retransmission.

If a ResendRequest(35 = 2) message is received, the LFIXT peer shall respond with a SequenceReset(35 = 4) with the GapFillFlag(123) set to “N” and a NewSeqNo(36) = NextNumOut.

The following example shows a FIXT initiator connected to an LFIXT acceptor. The FIXT initiator does not process messages with MsgSeqNum(34) = 98 and MsgSeqNum(34) = 99 for some reason. As a result, the FIXT initiator transmits a ResendRequest(35 = 2) asking for messages 98 and 99. The LFIXT acceptor responds with a SequenceReset(35 = 4) message with GapFillFlag(123) set to “N” and the NewSeqNo(36) set to the maximum of the EndSeqNo(16) from the ResendRequest(35 = 2) message and the LFIXT acceptor NextNumOut. This example demonstrates that an LFIXT session does not provide session layer message recovery.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

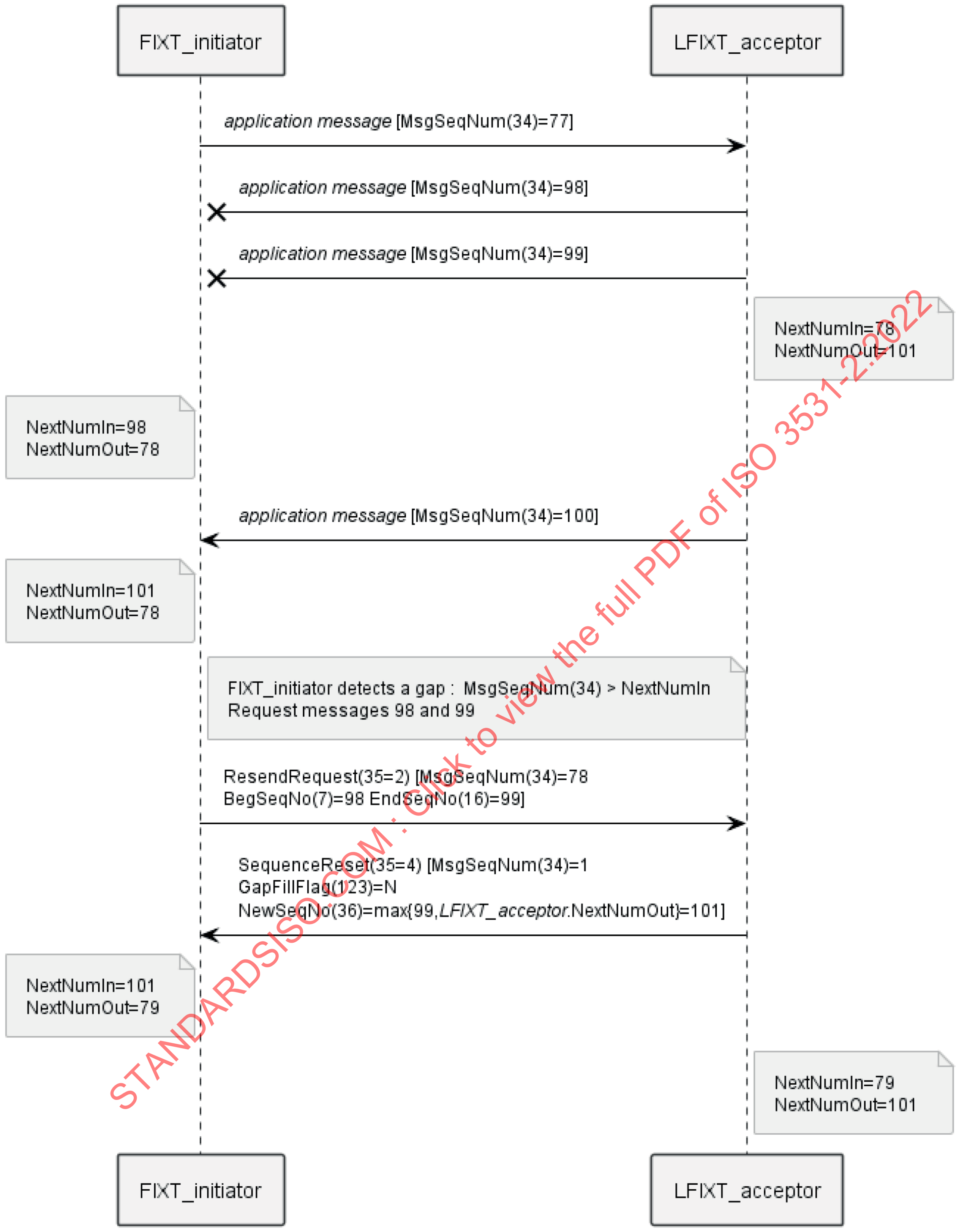

FIXT_initiator
LFIXT_acceptor
application message [MsgSeqNum(34)=77]
application message [MsgSeqNum(34)=98]
application message [MsgSeqNum(34)=99]
NextNumIn=78
NextNumOut=101
NextNumIn=98
NextNumOut=78
application message [MsgSeqNum(34)=100]
NextNumIn=101
NextNumOut=78
FIXT_initiator detects a gap : MsgSeqNum(34) > NextNumIn
Request messages 98 and 99
ResendRequest(35=2) [MsgSeqNum(34)=78
BegSeqNo(7)=98 EndSeqNo(16)=99]
SequenceReset(35=4) [MsgSeqNum(34)=1
GapFillFlag(123)=N
NewSeqNo(36)=max{99,*LFIXT_acceptor*.NextNumOut}=101]
NextNumIn=101
NextNumOut=79
NextNumIn=79
NextNumOut=101
FIXT_initiator
LFIXT_acceptor

Figure 24 — LFIXT resetting sequence numbers in LFIXT compatible mode

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

5.4.12 Processing invalid messages

If LFIXT encounters a garbled message or other data parsing issues, the LFIXT session processor should terminate the connection by sending a Logout(35=5) with a description of the error in Text(58) and terminate the transport layer connection.

6 FIX message routing

6.1 Message routing details – one firm-to-one firm (point-to-point)

The following table provides examples regarding the use of SenderCompID(49), TargetCompID(56), DeliverToCompID(128), and OnBehalfOfCompID(115) when using a single point-to-point FIX session between two firms. Assumption (A=sell-side, B =buy-side):

Table 12 — Message routing example, single session between two firms

	SenderCompID	OnBehalfOfCompID	TargetCompID	DeliverToCompID
A to B directly	A		B	
B to A directly	B		A	

6.2 Message routing details – third party message routing

The FIX session protocol supports the ability for a single FIX session to represent multiple counterparties. This can be in a 1-to-many, many-to-1, or 1-to-1 fashion. In addition, some third parties may be connected to other third parties effectively forming a "chain" of "hops" between the original message initiator and the final message receiver. The SenderCompID(49), OnBehalfOfCompID(115), TargetCompID(56), and DeliverToCompID(128) fields are used for routing purposes.

When a message travels over intermediary FIX sessions or an intermediary sends a message on behalf of another firm (using OnBehalfOfCompID(115)), that intermediary may add their details to the HopGrp component. The HopGrp may be used to build a "history" of FIX sessions through which the original message was transmitted. The HopGrp is not used to facilitate routing, rather it provides an audit trail of intermediary FIX sessions involved in transmitting the message to its final destination. An audit trail of intermediary involvement may be a requirement of some regulatory bodies or counterparties. When an intermediary FIX session forwards a message on to the next hop (may be the end point or another intermediary), that intermediary may add its hop details to the HopGrp (i.e. its SenderCompID(49) as HopCompID(628), its SendingTime(52) as HopSendingTime(629), and the received message's MsgSeqNum(34) or some other reference as HopRefID(630)).

Note that if OnBehalfOfCompID(115) or DeliverToCompID(128) message source identification/routing is used for a FIX session, then it must be used on all application messages transmitted via that session accordingly (reject the message if not).

The following figure provides examples regarding the use of SenderCompID(49), TargetCompID(56), DeliverToCompID(128), and OnBehalfOfCompID(115) when using a single FIX session to represent multiple firms. Assumptions are that A = sell-side, B = buy-side, and Hub = third party:

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

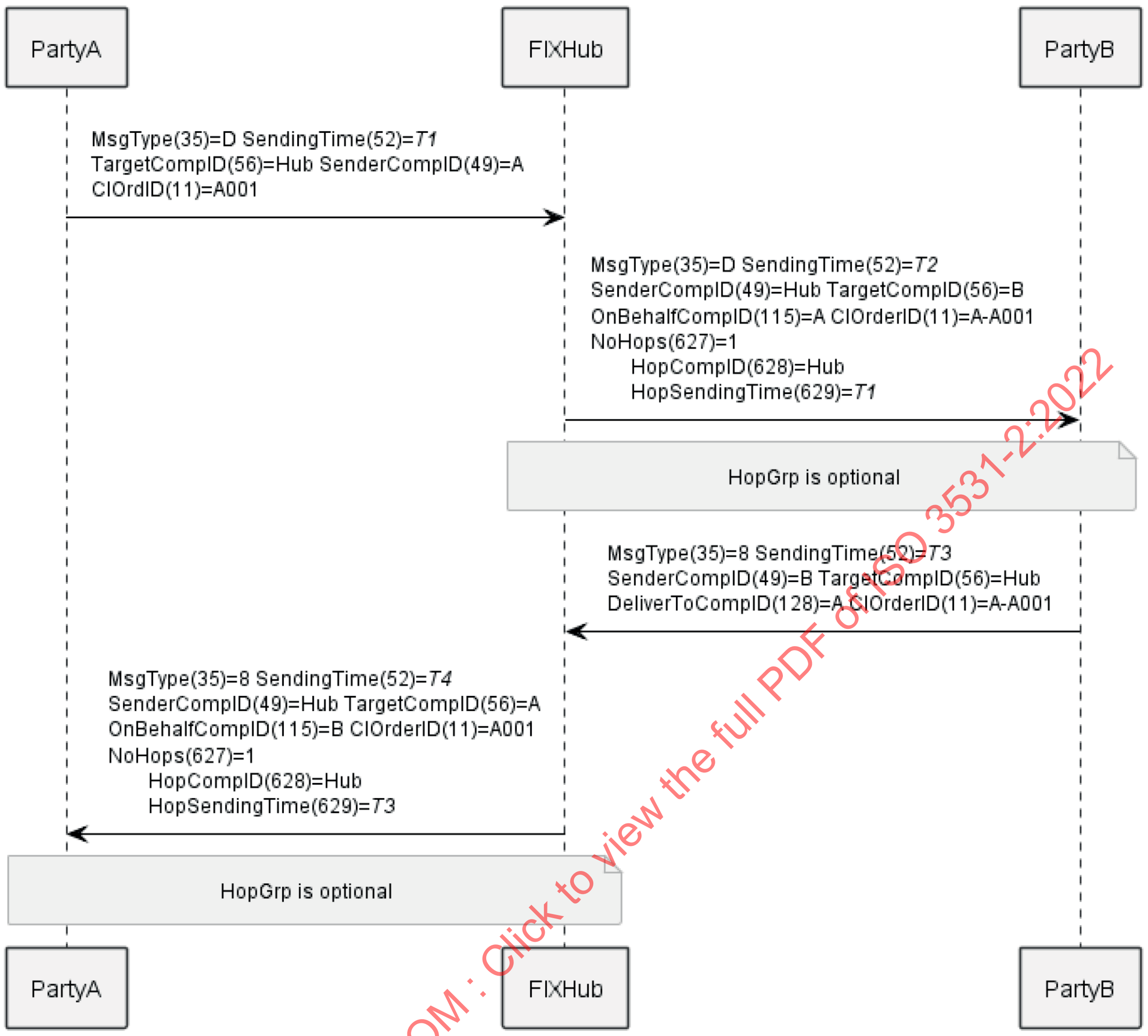

PartyA
FIXHub
PartyB
MsgType(35)=D SendingTime(52)=T1
TargetCompID(56)=Hub SenderCompID(49)=A
ClOrdID(11)=A001
MsgType(35)=D SendingTime(52)=T2
SenderCompID(49)=Hub TargetCompID(56)=B
OnBehalfCompID(115)=A ClOrderID(11)=A-A001
NoHops(627)=1
HopCompID(628)=Hub
HopSendingTime(629)=T1
HopGrp is optional
MsgType(35)=8 SendingTime(52)=T3
SenderCompID(49)=B TargetCompID(56)=Hub
DeliverToCompID(128)=A ClOrderID(11)=A-A001
MsgType(35)=8 SendingTime(52)=T4
SenderCompID(49)=Hub TargetCompID(56)=A
OnBehalfCompID(115)=B ClOrderID(11)=A001
NoHops(627)=1
HopCompID(628)=Hub
HopSendingTime(629)=T3
HopGrp is optional
PartyA
FIXHub
PartyB

Figure 25 — Message routing example, single session between multiple firms

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Note that some fields—for example, ClOrdID(11) on a NewOrderSingle(35 = D) message—must be unique for all orders on a given FIX session. Thus, when using OnBehalfOfCompID(115) or DeliverToCompID(128) addressing, a recommended approach is to prepend these fields to the original value. For example, if A sends the hub a ClOrdID(11) value of "A001", then the hub can specify a ClOrdID(11) of "A-A001" when sending the message to B to ensure uniqueness.

7 Transmitting alternatively encoded messages over a FIX session

XML content may be transmitted over a FIX session layer. FIXML or other XML-based content (such as ISO 20022 XML messages, FpML XML messages) within the XmlData(213) field in the StandardHeader of the XMLnonFIX(35=n) message.

The application message type is assumed to be contained in the XML document sent in XmlData(213) and is not processed by the FIX session layer.

© ISO 2022 – All rights reserved

7.1 Use of Attachment group

Other encodings, such as SBE, JSON, Google Protocol Buffers, PDF, jpeg should be transmitted using the AttachmentGrp component within the XMLnonFIX(35=n) message.[14]

The AttachmentGrp component provides the ability to attach other media type documents to a FIX message for transmission. The media type can be any of the media types (previously referred to as MIME types) that are listed by IANA RFC2046.

8 Components

8.1 AttachmentGrp

The AttachmentGrp component provides the ability to attach other media type documents to a FIX message for transmission. The media type can be any of the media types (previously referred to as MIME types) that are listed by IANA RFC2046. The AttachmentGrp is intended to be used to attach documents or payloads in other encodings, such as XML, JSON, TTL, SBE, GPB, PDF, TIFF, jpg to a FIX message. Note when the AttachmentGrp is used within a application message, such as the TradeCaptureReport(35=AE), the attachment should supplement the data already contained in the application message. It is not intended to replace the content of the business message. The standard fields within the application message should be populated, even if they duplicate data expressed within the attachment(s).

Tag	Name	Req'd	Description
2104	NoAttachments		
→2105	AttachmentName	N	Required if NoAttachments(2104) > 0. Specifies the file name of the attachment.
→2106	AttachmentMediaType	N	Required if EncodedAttachment(2112) is present. The MIME media type (and optional subtype) of the attachment. The values used are those assigned, listed and maintained by IANA (www.iana.org) [RFC2046].
→2107	AttachmentClassification	N	Specifies semantically the type of the attached document from a business perspective. The default classification scheme reuses the FIX standard classification scheme of a high level section (pretrade, trade, posttrade, etc.) and a category, then a specific application or document type. The expression follows {"section/category/application type"}.
→2108	AttachmentExternalURL	N	Either AttachmentExternalURL(2108) or EncodedAttachment(2112) must be specified if NoAttachments(2104) > 0.
→2109	AttachmentEncodingType	N	The encoding type of the content provided in EncodedAttachment(2112). Required if EncodedAttachment(2112) is present.
→2110	UnencodedAttachmentLen	N	Unencoded content length in octets. Can be used to validate successful unencoding.
→2111	EncodedAttachmentLen	N	Length in octets of EncodedAttachment(2112). Must be set if EncodedAttachment(2112) is specified

[14] AttachmentGrp component added in EP167

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Tag	Name	Req'd	Description
			and must immediately precede it.
→2112	EncodedAttachment	N	The content of the attachment in the encoding format specified in the AttachmentEncodingType(2109) field. Either AttachmentExternalURL(2108) or EncodedAttachment(2112) must be specified if NoAttachments(2104) > 0.
→Component	AttachmentKeywordGrp	N	Optional list of keywords associated with the EncodedAttachment(2112).

8.2 AttachmentKeywordGrp

The AttachmentKeywordGrp component provides a place to associate keywords with an attachment document to support the current approach of tagging to support metadata.

Tag	Name	Req'd	Description
2113	NoAttachmentKeywords		
→2114	AttachmentKeyword	N	Required if NoAttachmentKeywords(2113) > 0. Can be used to provide data or keyword tagging of the content in EncodedAttachment(2112).

8.3 HopGrp

The HopGrp is used to build a history of the various FIX sessions over which an application message was transmitted. The HopGrp is not used to facilitate message routing, rather it provides an audit trail of intermediary FIX sessions to the ultimate receiver of a message.

Tag	Name	Req'd	Description
627	NoHops		
→628	HopCompID	N	Assigned value used to identify the third party firm which delivered a specific message either from the firm which originated the message or from another third party (if multiple “hops” are performed). It is recommended that this value be the SenderCompID (49) of the third party.
→629	HopSendingTime	N	Time that HopCompID (628) sent the message. It is recommended that this value be the SendingTime (52) of the message sent by the third party.
→630	HopRefID	N	Reference identifier assigned by HopCompID(628) associated with the message sent. It is recommended that this value be the MsgSeqNum(34) of the message sent by the third party.

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

8.4 MsgTypeGrp

The MsgTypeGrp may be used to specify the application level messages supported by a peer over the FIX session.

Tag	Name	Req'd	Description
384	NoMsgTypes		
→372	RefMsgType	N	Specifies a specific, supported MsgType(35). Required if NoMsgTypes(384) is > 0. Should be specified from the point of view of the sender of the Logon(35=A) message.
→385	MsgDirection	N	Indicates direction (send vs. receive) of a supported MsgType(35). Required if NoMsgTypes(384) is > 0. Should be specified from the point of view of the sender of the Logon(35=A) message.
→1130	RefApplVerID	N	Specifies the service pack release being applied to an application message.
→1406	RefApplExtID	N	Specified the extension pack being applied to a message.
→1131	RefCstmApplVerID	N	Specifies a custom extension to a message being applied at the session level.
→1410	DefaultVerIndicator	N	Indicates that this Application Version (RefApplVerID(1130), RefApplExtID(1406),RefCstmApplVerID(1131)) is the default for the RefMsgType(372) field.

8.5 StandardHeader

Tag	Name	Req'd	Description
8	BeginString	Y	FIX.4.2 | FIX.4.4 | FIXT.1.1 BeginString(8) must be the first field in the message.
9	BodyLength	Y	Message length, in octets. BodyLength(9) must be the second field in the message.
35	MsgType	Y	Defines message type. MsgType(35) must be the third field in the message.
1128	ApplVerID	N	**FIXT.1.1 ONLY** Indicates application version being used for a message instance. ApplVerID(1128) applies to a specific message occurrence.
1156	ApplExtID	N	**FIXT.1.1 ONLY** Indicates an extension pack number being used for a message instance The ApplExtID(1156) applies to a specific message occurrence.
1129	CstmApplVerID	N	**FIXT.1.1 ONLY** Identifies a custom (user specified) version for a message instance.
49	SenderCompID	Y	(Always unencrypted)
56	TargetCompID	Y	(Always unencrypted)
115	OnBehalfOfCompID	N	Trading partner company ID used when sending messages via a third party (Can be embedded within encrypted data section.)
128	DeliverToCompID	N	Trading partner company ID used when sending messages via a third party (Can be embedded within encrypted data section.)
90	SecureDataLen	N	Length field for SecureData(91).

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

Tag	Name	Req'd	Description
			SecureDataLen(90) must be present and unencrypted if SecureData(91) is present in the message.
91	SecureData	N	Encrypted message content. Tag number, separator ("="), and delimiter (<SOH>) must be unencrypted. If present in the message, SecureData(91) must be immediately preceded by SecureDataLen(90). The use of SecureData(90) and its associated length field SecureDataLen(91) was deprecated as of the FIXT.1.1 session specification. Application layer encryption is still supported by counterparty agreement.
34	MsgSeqNum	Y	(Can be embedded within encrypted data section.)
50	SenderSubID	N	(Can be embedded within encrypted data section.)
142	SenderLocationID	N	Sender's LocationID (i.e. geographic location and/or desk) (Can be embedded within encrypted data section.)
57	TargetSubID	N	"ADMIN" reserved for administrative messages not intended for a specific user. (Can be embedded within encrypted data section.)
143	TargetLocationID	N	Trading partner LocationID (i.e. geographic location and/or desk) (Can be embedded within encrypted data section.)
116	OnBehalfOfSubID	N	Trading partner SubID used when delivering messages via a third party. (Can be embedded within encrypted data section.)
144	OnBehalfOfLocationID	N	Trading partner LocationID (i.e. geographic location and/or desk) used when delivering messages via a third party. (Can be embedded within encrypted data section.)
129	DeliverToSubID	N	Trading partner SubID used when delivering messages via a third party. (Can be embedded within encrypted data section.)
145	DeliverToLocationID	N	Trading partner LocationID (i.e. geographic location and/or desk) used when delivering messages via a third party. (Can be embedded within encrypted data section.)
43	PossDupFlag	N	Always required for retransmitted messages, whether prompted by the sending system or as the result of a resend request. (Can be embedded within encrypted data section.)
97	PossResend	N	Required when message may be duplicate of another message sent under a different sequence number. (Can be embedded within encrypted data section.)
52	SendingTime	Y	(Can be embedded within encrypted data section.)
122	OrigSendingTime	N	Required for message retransmitted as a result of a ResendRequest. If data is not available set to same value as SendingTime (Can be embedded within encrypted data section.)
212	XmlDataLen	N	Required when XmlData(231) is present. Length of the

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved

Tag	Name	Req'd	Description
			XmlData(231) field in octets.
213	XmlData	N	Can contain a XML formatted message block (e.g. FIXML, ISO20022, FpML).
347	MessageEncoding	N	Type of message encoding used for fields of datatype *data*, commonly referred to as encoded fields. MessageEncoding(347) must be present if any fields of datatype *data* are present in the message.
369	LastMsgSeqNumProcessed	N	The last MsgSeqNum(34) value received by the FIX session processor and processed by downstream application, such as trading system or order routing system. May be specified on every message sent. Useful for detecting a backlog with a counterparty.
Component	HopGrp	N	The HopGrp may be used to track the FIX sessions over which an application message has been sent.

8.6 StandardTrailer

Tag	Name	Req'd	Description
93	SignatureLength	N	Length field for Signature(89). SignatureLength(93) must be present and unencrypted if Signature(89) is present in the message. SignatureLength(93) must not be included as part of the encrypted content in SecureData(91).
89	Signature	N	If Signature(89) is present, it must be immediately preceded by SignatureLength(93). Signature(89) must not be included as part of the encrypted content in SecureData(91).
10	CheckSum	Y	Three-octet character representation of the modulo 256 checksum. Checksum(10) must be the last field in the message. The end of field delimiter (<SOH>) of Checksum(10) serves as the end of message delimiter.

9 Messages

9.1 Heartbeat message

The Heartbeat(35=0) message is sent unilaterally to keep a FIX connection active during periods of inactivity and it is also sent as a response to a TestRequest(35=1) from a peer.

Tag	Name	Req'd	Description
Component	**StandardHeader**	Y	MsgType = 0 T
112	TestReqID	N	Required when the heartbeat is the result of a Test Request message.
Component	**StandardTrailer**	Y	

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

9.2 TestRequest message

The TestRequest(35=1) is used to force a response from the peer. A HeartBeat(35=0) with the TestReqID(112) set to the value from the TestRequest(35=1)

Tag	Name	Req'd	Description
Component	**StandardHeader**	Y	MsgType = 1
112	TestReqID	Y	An arbitrary identifier provided by the sender of the TestRequest(35=1) message this must be returned by the peer in a HeartBeat(35=0) message response to confirm viability of the FIX connection.
Component	StandardTrailer	Y	

9.3 ResendRequest message

The ResendRequest(35=2) is sent by the receiving application to initiate the retransmission of messages.

Tag	Name	Req'd	Description
Component	StandardHeader	Y	MsgType = 2
7	BeginSeqNo	Y	The sequence number of the first message that requires retransmission.
16	EndSeqNo	Y	The sequence number of the last message sequence number. “0” specifies retransmission of all messages since BeginSeqNo(7)
Component	**StandardTrailer**	Y	

9.4 Reject message

The Reject(35=3) message should be issued when a message is received but cannot be properly processed due to a session laver or tagvalue encoding violation.

Tag	Name	Req'd	Description
Component	**StandardHeader**	Y	MsgType = 3
45	RefSeqNum	Y	MsgSeqNum of rejected message
371	RefTagID	N	The tag number of the FIX field being referenced.
372	RefMsgType	N	The MsgType of the FIX message being referenced.
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

STANDARDSISO.COM : Click to view the full PDF of ISO 3531-2:2022

© ISO 2022 – All rights reserved