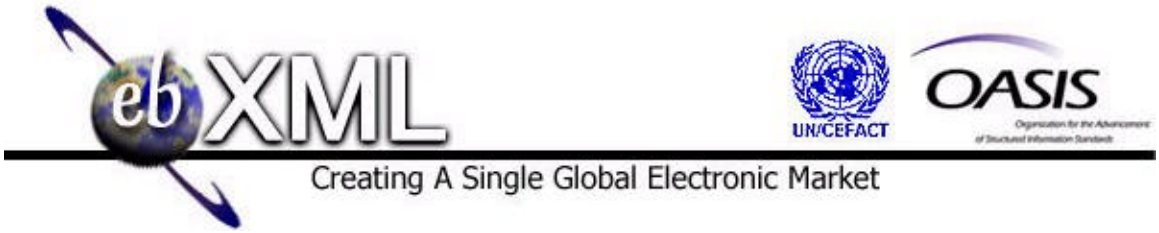



# ebXML Transport, Routing & Packaging Reliable Messaging Specification

## Working Draft 22-September-2000

**This version:**
ebXML Reliable Messaging Specification v0-078
**Latest version:**
N/A
**Previous version:**
v0-074
**Editor:**
Jim Hughes <jfh@fs.fujitsu.com>
**Authors:**
Masayoshi Shimamura <shima@rp.open.cs.fujitsu.co.jp>
**Contributors:**
See Acknowledgements

---

## Abstract

This document defines the structures and processes used to provide Reliable Messaging within the ebXML Transport, Routing and Packaging architecture.

## Status of this Document

This document represents work in progress and no reliance should be made on its content.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119.

*Editor Note 1: Significant changes since the previous version include:*
*- deleted RM-Group semantics*
*- added Recovery Sequences for each error*
*- added definition of "AtMostOnce" semantics and "Unspecified" semantics*
*- added definition of size of Sequence Number*
*- added informative text which describes the relationship between an MS ack and a Transport protocol ack*

# Table of Contents

# 1 Introduction

## 1.1 Purpose and Scope

This specification defines the Reliable Messaging function used between ebXML Messaging Services. It responds to the requirements for Reliable Messaging found in section 4.2(1) of Reference [1]. Material from this draft document will be incorporated into the Messaging Services Specification at a future date.

## 1.2 Goal and Policy

This ebXML Reliable Messaging Specification describes how to provide reliable message transmission between two Messaging Services when the "From" Party sending a message through these Messaging Services specifies "AtMostOnce" delivery semantics in the Message Header.

"Reliable Messaging" means that the Sending Party's Messaging Service Handler will obtain a positive confirmation, either through a Messaging Service Level Acknowledgment Message, or by time-out, that the message was or was not delivered into the Receiving Party Messaging Service Handler's persistent storage. Further message processing at the Receiving Party (including generation of Business Process Level Acknowledgment Messages) is not within the scope of this Reliable Messaging Specification.

All ebXML Messaging Service implementations SHALL support the Reliable Messaging function.

## 1.3 Features (this section will be completed once other parts are done)

- Item…
- Item…

# 2 Reliable Messaging Architecture

## 2.1 Basic Concepts

To achieve reliable messaging between Parties, this specification defines a process which enables the Parties' ebXML Messaging Services to communicate with each other using "at most once" semantics, coupled with a timeout to determine lost messages.

For the purposes of this document, the term "*Sender*" means the Sending Party's Messaging Service that sends the message on the underlying message transport, and "*Receiver*" means the Messaging Service used by the Receiving Party. The term "*From-Party*" means the party that originally prepared the message and provided the message to its Messaging Service, and the term "*To-Party*" means the party that was identified by the From-Party as the final recipient of the message.

For example, a simple message transmission using two Message Service Handlers and one transport is shown in Figure 2-1.

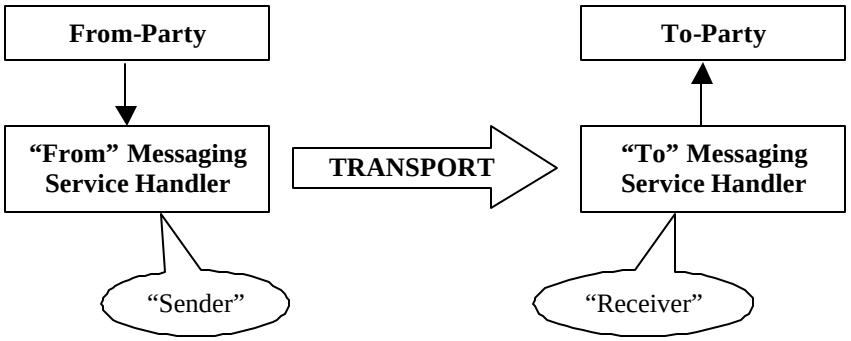



**Figure 2-1: Simple Message Transmission**

Reliable Messaging consists of the following basic concepts:

1) Messages are sent and received through Messaging Service Handlers (MSH), which function on behalf of their respective Parties (and Business Processes). With respect to a particular underlying transport, each MSH can be identified as a "Sender" or a "Receiver".

2) A message is identified by its **MessageId** field, which is contained in the Message Header's **MessageData** element created by the Sender.

3) When the From-Party requests Reliable Messaging semantics for the message, the Sender sets the **DeliverySemantics** field in the **ReliableMessagingInfo** element of the Message Header to "AtMostOnce".

4) Reliable Messaging processing requires no changes to the Message Header during transmission, once the Message Header is prepared by the From-Party's MSH.

5) Reliable Messaging uses a "Routing Header" contained in the Message Envelope.

> *Editor Note 2: In a later section of this draft, it is proposed that the single Routing Header Data Element is updated as the message moves between intermediate MSHs. An alternative is that one Routing Header Data Element could be added to the Routing Header for each Sender-Receiver-Transport triplet as the message moves from the From-Party to the To-Party through a sequence of MSHs. This would add complexity but would provide an audit trail.*

6) A Reliable message indicated by setting the **DeliverySemantics** field to **AtMostOnce.**

7) For each reliable message, the Sender generates a **Sequence Number** that is unique to the Sender-Receiver-Transport triplet. For subsequent reliable messages, the Sender increments the Sequence Number placed in that message. The **Sequence Number** is contained in the Routing Header Data Element.

8) When the Receiver receives a reliable message, the Receiver compares the received reliable message's **Sequence Number** with the previous reliable message's **Sequence Number**, if available. If the newly received reliable message's **Sequence Number** is one greater than the previous reliable message's **Sequence Number**, the Receiver signals a normal completion of the reliable message transmission back to the Sender by sending an "Acknowledgement" message. In any other case, the Receiver sends an error message to the Sender and the Sender re-sends (at least) the missing message.

9) Within a reliable message transmission, the Receiver can determine whether a received message is a duplicate message or not by using the **MessageId** and/or the Sender-Receiver-Transport unique **Sequence Number.** If the received message is a duplicate, the Receiver discards the message. If the message is not a duplicate, the Receiver stores the message in its persistent storage and delivers the message to a higher processing level.

10) To detect loss of a reliable message, the Sender sets a time-out for that message. If the transmitted reliable message is lost due to system or communication failure, the Sender will re-send this message the number of times specified in the Trading Partner Agreement (TPA) to the Receiver before reporting failure to the From-Party.

11) A Messaging Service level Acknowledgement is sent from the Receiver to the Sender for every received message.

## 2.2 Message Envelope Elements used for Reliable Messaging

### 2.2.1 Message Header – Message Data Element

Reliable Messaging uses the **MessageId** field to uniquely identify the message.

### 2.2.2 Message Header – Reliable Messaging Info Element

When the From-Party requests Reliable Messaging semantics for the message, the Sender sets the **DeliverySemantics** field to "AtMostOnce". In these semantics, an Acknowledgment Message and the recovery sequence described in this specification are used. The **Sequence Number** field in the Routing Header Data Element is utilized. All the messages are stored in temporary persistent store in the Sender and the Receiver for recovery. In Reliable Messaging semantics, the ordering of all reliable messages is guaranteed.

When the From-Party requests Unreliable Messaging semantics for the message, the Sender sets the **DeliverySemantics** field to "Unspecified". In these semantics, an Acknowledgment Message is not used during message transfer and lost messages are not recovered. The **Sequence Number** field does not have a value in the Routing Header Data Element, and Sender and Receiver are not required to use temporary persistent store.

*Editor Note 3: In Reliable Messaging semantics, message delivery is guaranteed (except for an error). Since a lost message is re-sent by the MSH automatically, the Sending Party does not need to re-send a lost message unless it wants to attempt a retry after all the MSH-level attempts fail. Should the semantics be called "ExactlyOnce" instead of "AtMostOnce"?*

*Editor Note 4: The semantics of "Unspecified" may be unclear, as the MSH can't decide whether the semantics are reliable or not. If the semantics mean unreliable messaging, should there be another name such as "BestEffort"?*

### 2.2.3 Routing Header

For each Sender-Receiver-Transport triple used to transmit the message, the Sender SHALL provide a Routing Header, which includes the mandatory elements shown in Table 2-1.

*Editor Note 5: There is a need to identify the particular Messaging Service instance that is processing the message on behalf of the From-Party or To-Party. A **MessageServiceId** is used for this. If a multi transport function and/or multi path function is really needed, MessageServiceId also might be needed.*

*Editor Note 6: The mandatory (but not optional) routing header elements are required in all instances, even when the message is not sent with Reliable Messaging semantics. This permits audit functions (to be further defined in the Messaging Service Specification).*

180 **Table 2-1: Mandatory Routing Header Data Elements**

| *Element* | *Outline Description* |
|---|---|
| **SenderID** | Sender's Messaging Service Handler logical address, using PartyID format (context and text value) |
| **ReceiverID** | Receiver's Messaging Service Handler logical address, using PartyID format (context and text value) |

181 When a Routing Header is used for a message sent with Reliable Messaging functions, two
182 additional Routing Header Data Elements SHALL be added to the Routing Header by the Sender.
183 They are described in Table 2-2.

184 **Table 2-2: Additional Routing Header Data Elements**

| *Element* | *Outline Description* |
|---|---|
| **Sequence Number** | Integer value which is incremented (e.g. 1, 2, 3, 4, ...) for each Sender-prepared message sent to the Receiver using a particular transport. The Sequence Number takes a value in the range 1 to $2^{32}$-1 (4,294,967,295). In following cases, the Sequence Number takes value "1":<br>• First message from a Sender to a Receiver using a particular transport<br>• First message after wraparound (next value after $2^{32}$-1)<br>• First message after removing Sequence Number information in the Sender (Sender MAY remove Sequence Number information when it has no messages which were sent to the Receiver for long time)<br>The Receiver may use this Sequence Number to check for repeated messages, or the Receiver may use the **MessageId**. |

185 ## 2.3 Message Transfer Sequence

186 A reliable message SHALL be sent and a single acknowledgement message returned to the
187 Sender once the reliable message has been received by the Receiver. As a message is received,
188 the Receiver MAY process it appropriately, usually by passing the message to the higher-level
189 "To-Party".

190 With respect to a particular Sender, Receiver and transport triple, transmission of one reliable
191 message SHALL be completed before another reliable message may be sent.

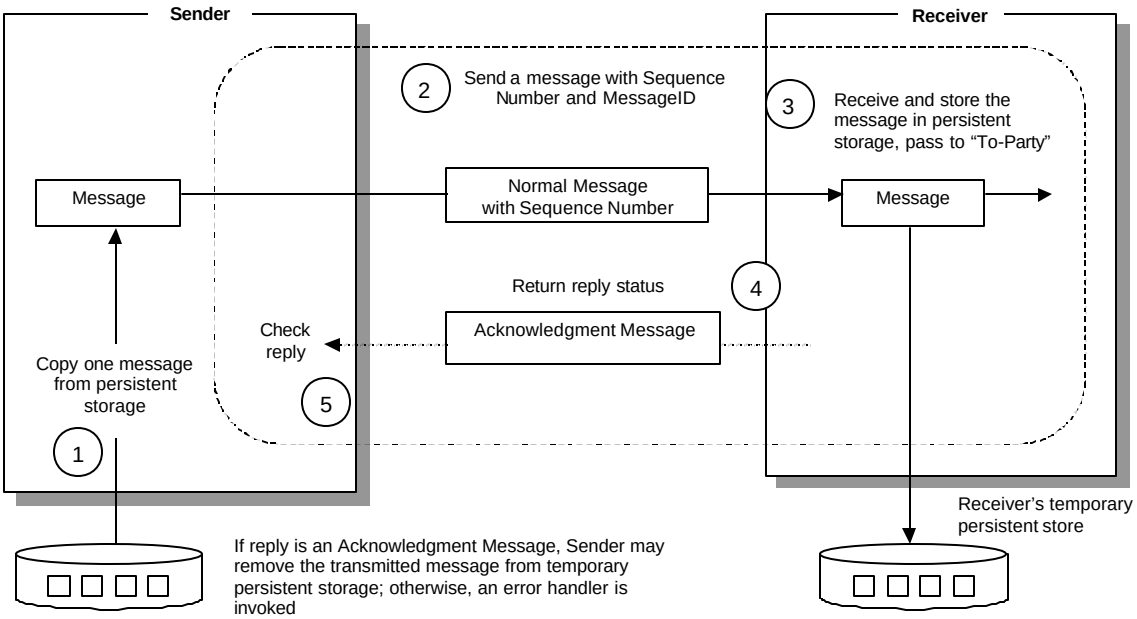



**Figure 2-2: Reliable Message Transfer Sequence**

Reliable Messaging processing is shown in the following sequence:

(1) Message copy

Sender initially stores messages passed from the ebXML "From-Party" in temporary persistent storage, and then copies the stored message for message transfer.

(2) Sending message

A Reliable message has DeliverySemantics = AtMostOnce set, and receipt of a message with this value notifies the Receiver of Reliable Messaging. The message is identified by a **Message** Identifier in the Message Header and a Sequence Number in the Routing Header.

(3) Receiving, checking and storing message

The Receiver receives the reliable message and then checks whether the message is repeated or not by using the Sequence Number and/or Message-Id. If it is not a repeated message, the Receiver stores the message into persistent storage and processes the message appropriately.

If it is a repeated message, the Receiver discards the message.

(4) Acknowledgment by Receiver

The Receiver returns an Acknowledgment Message to the Sender.

(5) Sender checks reply and removes transferred message

Sender checks the Acknowledgement Message from the Receiver. If the reply is the appropriate Acknowledgement Message for the transferred message, Sender may remove the transferred message from Sender's persistent storage if the message is no longer needed for some other Messaging Service function.

## 2.4 Recovery Sequence for Lost Messages

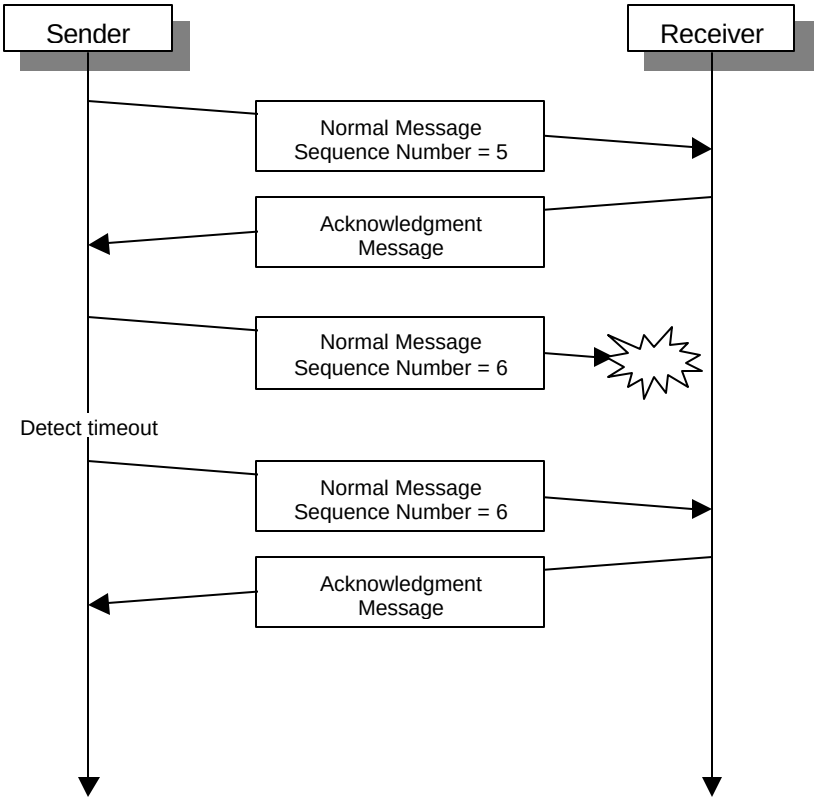



**Figure 2-3: Recovery Sequence for Lost Messages**

If the sent message does not reach the Receiver, the Sender detects a timeout while waiting for an Acknowledgement Message, since the Receiver does not return the Acknowledgement Message. The Sender's recovery handler re-sends last message again. When the Receiver receives the re-sent message, the Receiver checks whether the received message is a repeated message or not and returns an Acknowledgement Message to the Sender.

## 2.5 Detection of Repeated Messages by the Receiver (non-normative)

Detection of repeated messages in the Receiver using **Message Identifiers** and/or **Sequence Numbers** is implementation dependent. However, an effective detection logic can be suggested which uses **Sequence Numbers**.

The Receiver receives the reliable message and then compares the received reliable message's **Sequence Number** with the immediately previous reliable message's **Sequence Number**:

(1) Received message's **Sequence Number** == Previous message's **Sequence Number** + 1

The message is not a repeated message. The Receiver stores the message into persistent storage, processes the message appropriately and returns an Acknowledgement Message.

(2) Received message's **Sequence Number** == Previous message's **Sequence Number**

The message is a repeated message. The Receiver discards the message and returns Acknowledgement Message. [? Does this mean duplicate Ack messages sent?]

Note: When both the received message's **Sequence Number** and the previous message's **Sequence Number** are 1, the Receiver will have to use the **MessageId** instead of a Sequence Number to check for a repeated message. This situation happens only when following sequence occurs (this is a very unusual situation):

a) The Sender sends only one message to the Receiver, and it's the first message for the Sender-Receiver-Transport triplet. There is no subsequent message for long time.

b) Since there is no other message that should be sent to the Receiver for long time, the Sender might remove its **Sequence Number** information for the Receiver. But the Receiver does not remove its **Sequence Number** information for the Sender.

c) After that, the Sender is given a message from the sending Party that should be sent to the Receiver. Since the Sender's **Sequence Number** information was previously removed, the Sender sends the message to the Receiver using **Sequence Number** 1.

(3) Received message's **Sequence Number** == 1 && not case (2)

The message is not a repeated message. The Receiver stores the message into persistent storage, processes the message appropriately and returns an Acknowledgement Message.

(4) Any other case

It means that the Sequence Number value is invalid. The Receiver discards the message and returns an error message.

## 2.6 Reliable Messaging Acknowledgement and Error Messages

### 2.6.1 General

The Messaging Service handles the following errors:

(1) Communication Protocol Errors

(2) ebXML Messaging Errors

(3) Timeout

(4) Transient Errors

The recovery handler in the Sender executes a Messaging Service recovery sequence for all errors above except for "(2) ebXML Messaging Errors", because (2) is processed in a level higher than the recovery handler.

### 2.6.2 Reliable Messaging Formats

Messages used to report Reliable Messaging acknowledgements and errors between Messaging Service Handlers are formatted according to Reference [2], with specific fields completed as shown below. The SenderID and ReceiverID fields in the Router Header identify the Messaging Service Handlers that are using Reliable Messaging semantics.

Each reliable message is unique to a Sender-Receiver-Transport triple, as discussed in Base Concepts, above. For the purposes of describing error and acknowledgement messages, the following terms are used:

- *Original-Sender* means the "From" PartyID contained in the reliable message

- *Original-Recipient* means the "To" PartyID contained in the reliable message

> *Editor Note 7: Not sure if we will need these definitions above, but they might be useful when multi-node networks are discussed. The error/ack message is sent between MSHs and not the PartyIDs.*

All acknowledgement and error messages SHALL contain at least these values:

- "From" PartyID = ReceiverID shown in the Routing Header Data Element
- "To" PartyID = SenderID shown in the Routing Header Data Element
- TPAId and ConversationID are those used in the reliable message
- ServiceInterface and Action are not present in the reliable message
- RefToMessageId = MessageId of the reliable message
- DeliverySemantics = "Unspecified"

### 2.6.3 Communication Protocol Errors

When the Sender or the Receiver detects a transport protocol level error (such as an HTTP, SMTP or FTP error), the appropriate transport recovery handler will execute a recovery sequence. No Reliable Messaging functions are involved in this recovery sequence, since it happens at a lower level.

However, if the Sender detects a transport protocol level error that is unrecoverable at the transport protocol level, or receives the error message "Communication Protocol Errors", the appropriate recovery handler in the Sender executes a Messaging Service recovery sequence. This recovery sequence SHALL use a retry interval and SHALL re-send the last message to the Receiver. The format of the re-sent message is exactly the same as the original message. In the recovery sequence or after the recovery sequence:

- If the Sender detects a transport protocol level error, which is unrecoverable at the transport protocol level, or receives the error message "Communication Protocol Errors" again, the recovery handler repeats the recovery sequence before returning an Acknowledgment Message or an error message except for Communication Protocol Errors.
- If the Sender detects or receives another error, the recovery handler executes an appropriate recovery sequence for the error.
- If the Sender receives an Acknowledgment Message, the message transmission is completed.

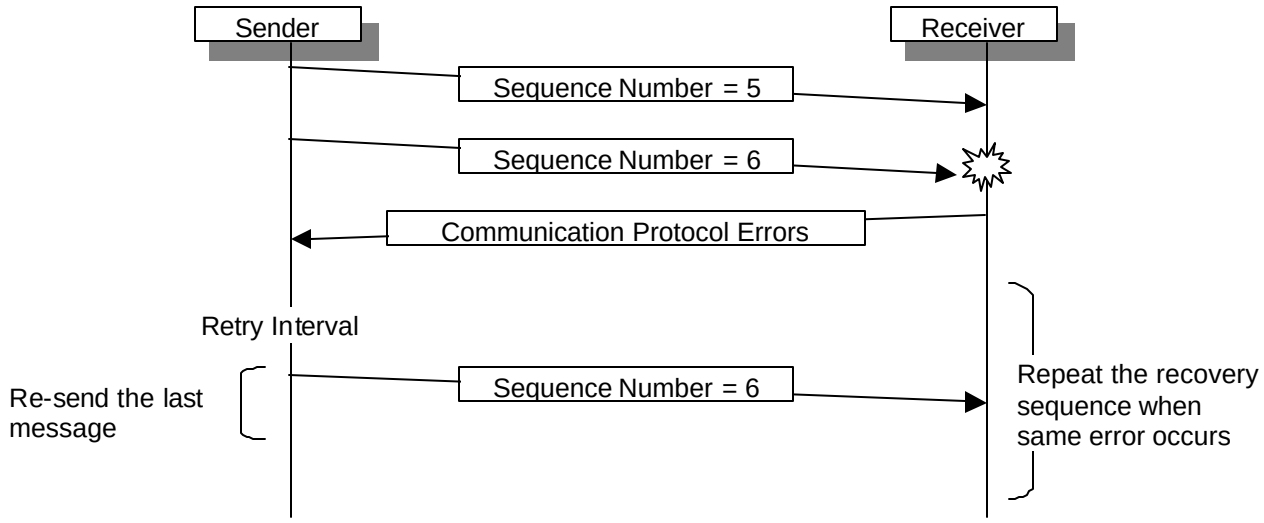



**Figure 2-4: Recovery Sequence for Communication Protocol Errors**

### 2.6.4 ebXML Messaging Errors

When the Sender receives error message “ebXML Messaging Errors”, the recovery handler passes this error to a higher-level and continues message transfer sequence.

### 2.6.5 Timeout

When the Sender detects a timeout while waiting for an Acknowledgement Message from the last sent message, the appropriate recovery handler in the Sender executes a Messaging Service recovery sequence. The timeout value is defined in the TPA as **Timeout**. This recovery sequence SHALL re-send the final message to the Receive and SHALL use a retry interval. The format of the re-sent message is exactly the same as the original message. In the recovery sequence or after the recovery sequence,

- If the Sender does not receive any error message or Acknowledgment Message in the retry interval, the recovery handler repeats the recovery sequence before returning an Acknowledgment Message or an error message.
- If the Sender detects or receives another error, the recovery handler executes the appropriate recovery sequence for the error.
- If the Sender receives an Acknowledgment Message in the recovery sequence, the message transmission is completed.

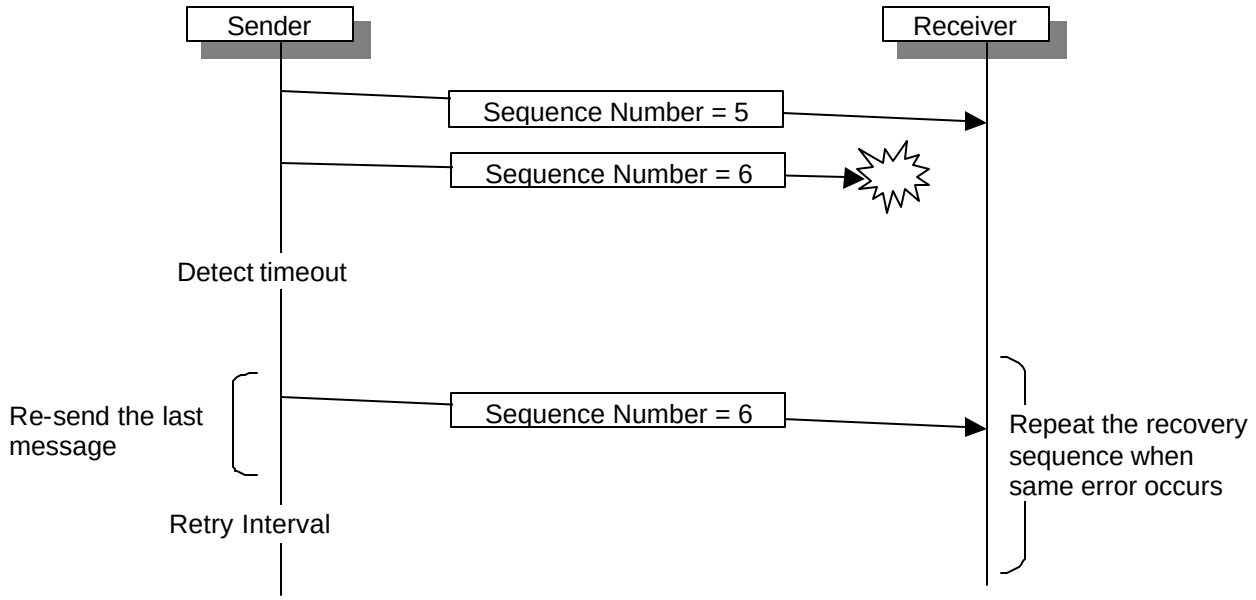



**Figure 2-5: Recovery Sequence for Timeout**

### 2.6.6 Transient Errors

When the Sender receives the error message "Transient Errors", the appropriate recovery handler in the Sender executes a Messaging Service recovery sequence. The recovery sequence SHALL suspend sending of further messages to the Receiver for the period specified in the **MinRetrySecs** field in the error message. If the **MinRetrySecs** field does not exist in the error message, **RetryInterval** specified in TPA is used as the suspending time. After the suspension, the Sender's recovery handler SHALL re-send the last sent message to the Receiver. The format of the re-sent message is exactly the same as the original message. In the recovery sequence or after the recovery sequence,

- If the Sender receives the error message "Transient Errors" again, the recovery handler repeats the recovery sequence before returning an Acknowledgment Message or an error message except for Transient Errors.
- If the Sender detects or receives another error, the recovery handler executes the appropriate recovery sequence for the error.
- If the Sender receives an Acknowledgment Message, the message transmission is completed.

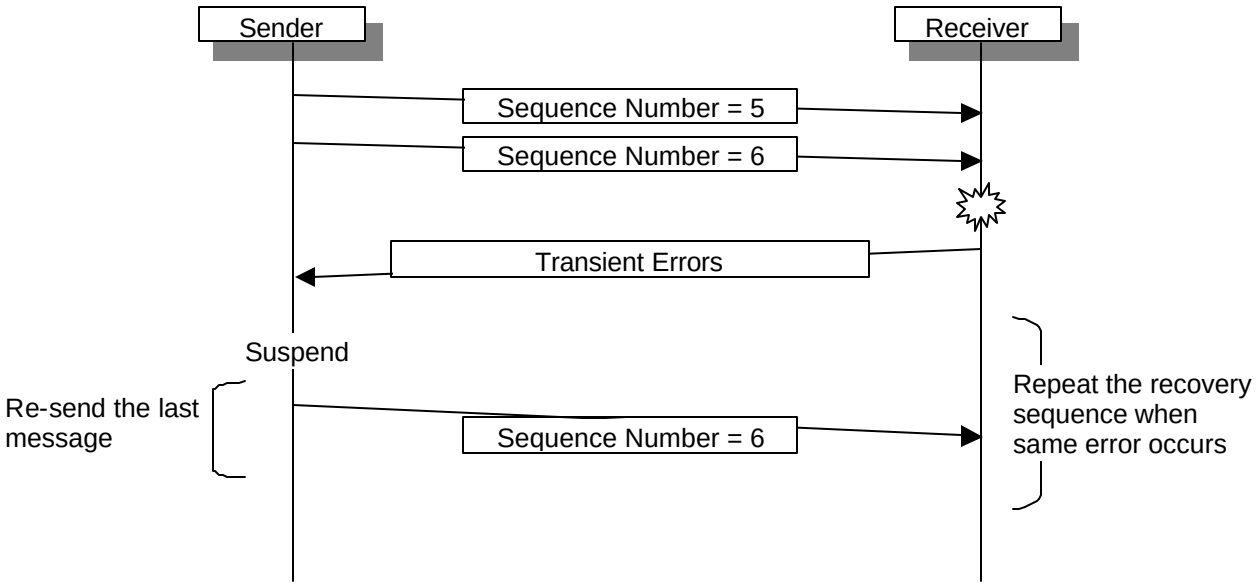



**Figure 2-6: Recovery Sequence for Transient Errors**

### 2.6.7 Acknowledgement Message

The Receiver's Messaging Service Handler sends this message to the Sender's Messaging Service Handler when the message is received. There is no reply to this message from the Sender's Messaging Service Handler. Since the Sender's Messaging Service Handler will not initiate a new message transmission until a current message transmission has been correctly received, there is no possibility that the Receiver will detect a non-sequential Sequence Number when determining if the received message is repeated message or not using Sequence Number. The **MessageType** SHALL be "Acknowledgement".

There is no Payload and no business level response information.

### 2.6.8 Maximum Number of Retries and Retry Interval

The retry interval is defined in the TPA as **RetryInterval**. When the total number of retries in a reliable message transmission reaches a maximum number defined in the TPA as **Retries** and the last error is still not resolved, the recovery handler will:

(1) Suspend sending messages to the Receiver

(2) Reports this incident to a higher-level so that system administrator can resolve this incident

When the System administrator resolves the incident, the recovery handler will reset the retry counter to zero and then re-start message transfer sequence from the uncompleted reliable message transmission.

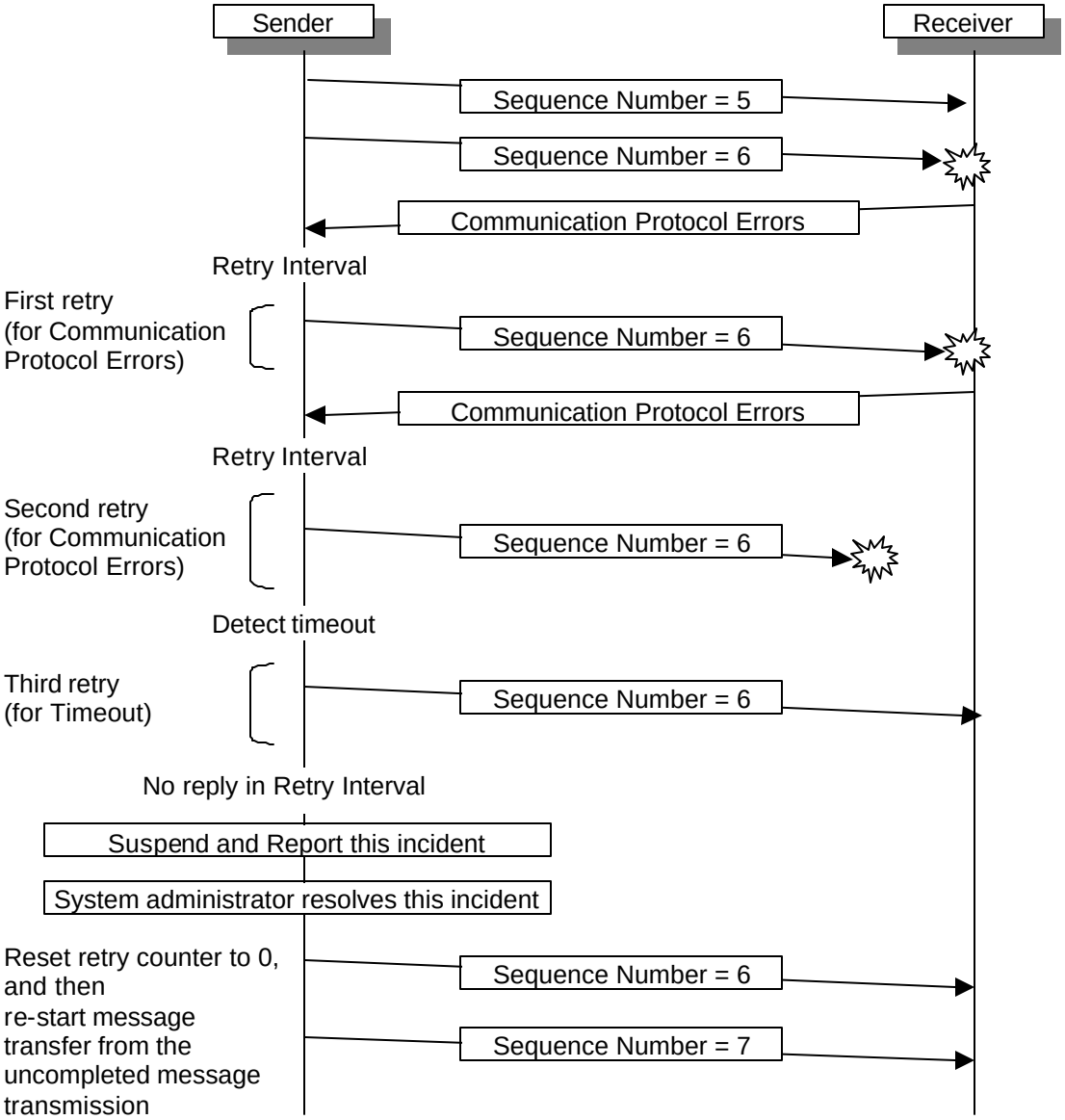



**Figure 2-7: Repeat of Recovery Sequence (specified maximum number of retries is 3)**

# 3 Relationship with Transport Protocols

The ebXML Messaging Service messages are carried by Transport Protocols as shown in the following sections.

## 3.1 HTTP

All ebXML Messaging Service messages are carried by an HTTP Request Message (POST method). The HTTP Response Message to the HTTP Request Message has no entity body.

359 **Table 3-1 Relationship with HTTP**

| *ebXML Messaging Service message* | *HTTP* |
|---|---|
| **Normal Message** | • Request Message (POST method) from Sender to Receiver<br>• Response Message to the Request Message has no entity body |
| **Acknowledgement Message** | • Request Message (POST method) from Receiver to Sender<br>• Response Message to the Request Message has no entity body |
| **Error Message** | • Request Message (POST method) from Receiver to Sender<br>• Response Message to the Request Message has no entity body |

360 ## 3.2 SMTP

361 All ebXML Messaging Service messages are carried as mail in an SMTP Mail Transaction.

362 **Table 3-2 Relationship with SMTP**

| *ebXML Messaging Service message* | *SMTP* |
|---|---|
| **Normal Message** | Mail Transaction from Sender to Receiver |
| **Acknowledgement Message** | Mail Transaction from Receiver to Sender |
| **Error Message** | Mail Transaction from Receiver to Sender |

363 ## 3.3 FTP

364 [TBD]

365 # 4 Reliable Routing

366 ## 4.1 Store and Forward Semantics

367 Reliable Routing consists of a series of individual simple Reliable Messaging transmissions
368 between a Sender and a Receiver. These *Store and Forward* semantics consist of the following
369 sequence:

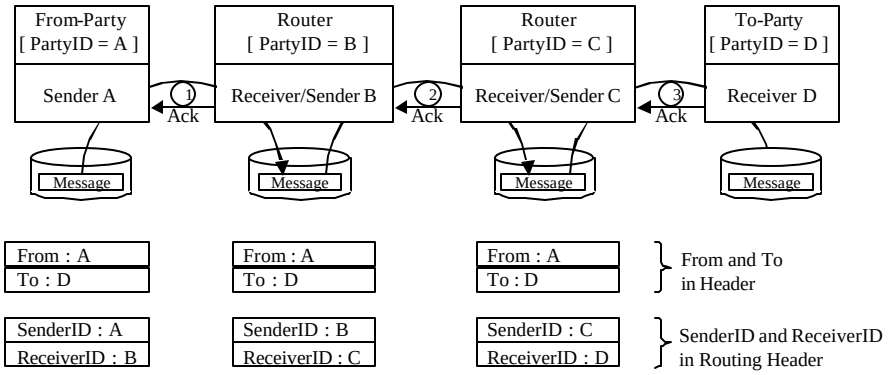
370 (1) Sender A transfers a message to Receiver B using Reliable Messaging.

371 (2) After completion of the reliable messaging transmission between Sender A and Receiver B,
372 Sender B transfers the received message to Receiver C using Reliable Messaging.

(3) After completion of the reliable messaging transmission between Sender B and Receiver C, Sender C transfers the received message to Receiver D using Reliable Messaging.

(4) [Repeat until end of routing]

**Figure 4-1 Reliable Routing**




## 4.2 Routing Information

The first Sender (From-Party's Sender) specifies the From/To elements in the Header, and the SenderID/ReceiverID elements in the Routing Header for first message transferred to the Router.

When the message is forwarded between Routers, the Router updates the SenderID/ReceiverID elements in the Routing Header for message forwarding to the next Router.

*Editor Note 8: As an alternative, subsequent Routers could add Router Headers to the message, which would permit auditing of the message transfer.*

## 4.3 Error Handling in Routing

When message forwarding to a subsequent Router is not available or fails from a particular Router, and if that Router received the message from previous Sender, the Router's Receiver returns an ErrorMessage (Transient Error) to the Sender instead of an Acknowledgement Message. By this rule, the Sending Party will not receive a Messaging Service acknowledgement of successful transmission until the message has actually been received by the Receiving Party's Message Service Handler.

# 5 Trading Partner Agreement (TPA) Considerations

Reliable Messaging uses the following arguments that are specified in the TPA.

**Table 5-1 Reliable Messaging related arguments specified in the TPA**

| *Argument* | *Outline Description* |
|---|---|

| | |
|---|---|
| **Timeout** | Wait time for any response from the Receiver.<br>• Integer value specifying a number of seconds<br>• After sending a Normal Message, the Sender SHALL wait for any response (MS Acknowledgement or Error Message) for specified time before start of retry |
| **Retries** | Maximum number of retries.<br>• Integer value specifying the number of retries<br>• The Sender SHALL repeat retries the specified number of times until the Sender receives an MS Acknowledgement Message<br>• If the Sender does not receive an MS Acknowledgement Message after the maximum number of retries, the Sender SHALL notify the incident to the higher level (application and/or system admin) |
| **RetryInterval** | Wait time between retries.<br>• Integer value specifying a number of seconds<br>• After a retry, the Sender SHALL wait for a response (MS Ack or Error Message) for specified time before start of next retry |

# 6 Changes to Current ebXML Specifications

*Editor Note 9: This section will be deleted when RM material is moved into the Messaging Service Specification.*

## 6.1 Changes to ebXML Messaging Service Specification v0-1

- Routing Header and Routing Header Data Elements will included in the Message Envelope.
- (others to be determined when the new specification is finished)

## 6.2 Changes to Other ebXML Specifications

There are no changes to other ebXML specifications.

# 7 Definition of terms

- **Exactly Once**

  A message delivery semantic that means:

  - Message delivery is guaranteed
  - A message reaches the Receiving Party only once

- Even if a message does not reach the Receiving Party, Sending Party does not need to execute retry by itself (retry is automatically executed by the messaging service).

- **At Most Once**

  A message delivery semantic that means:

  - Message delivery is not guaranteed
  - A message reaches the Receiving Party either once, or not at all
  - When a message is not delivered, the Sending Party can detect the incident
  - In the incident, if the Sending Party want to guarantee message delivery, the Sending Party has to execute retry by itself

- **Best Effort**

  A message delivery semantic that means:

  - Message delivery is not guaranteed
  - A message reaches the Receiving Party either once, or not at all
  - Even if message does not reach the Receiving Party, the Sending Party can not detect the incident

# 8 References


[1] ebXML Transport, Routing and Packaging: Overview and Requirements, version 0-96, 26 May 2000

[2] ebXML Transport, Routing and Packaging: Messaging Service Specification, version 0-21, 13 September 2000


# 9 Acknowledgements


The author wishes to acknowledge the members of the ebXML TR&P who commented on Fujitsu's proposal in the face-to-face meetings and in e-mail.


# 10 Authors' Address


Masayoshi Shimamura
Fujitsu Limited
Shinyokohama Nikko Bldg., 15-16, Shinyokohama 2-chome
Kohoku-ku, Yokohama 222-0033, Japan
Telephone: +81-45-476-4590
E-mail: shima@rp.open.cs.fujitsu.co.jp