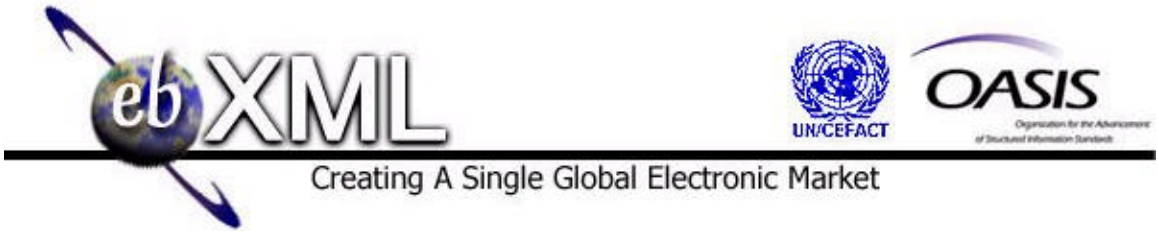


ebXML Transport, Routing & Packaging Reliable Messaging Specification

Working Draft 26-July-2000

This version:
ebXML Reliable Messaging Specification v0-06.doc
Latest version:
N/A
Previous version:
This is the first version.
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 is the first draft specification of ebXML Reliable Messaging and has been prepared for internal discussion within the ebXML Transport, Routing and Packaging Team. It is based on a presentation made in a Team meeting in Burlington, MA during 11-12 July, 2000. 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.

Table of Contents

1 Introduction

1.1 Purpose and Scope

This specification defines the Reliable Messaging function used within the ebXML messaging handling layer between ebXML Messaging Systems. It responds to the requirements for Reliable Messaging found in section 4.2(1) of Reference [1], and proposes changes to be incorporated into the Message Header Specification.

1.2 Goal and Policy

This ebXML Reliable Messaging Specification describes how to provide reliable message transmission between two Parties when the Parties use "at most once" communication semantics.

A policy of this specification is to limit changes to existing ebXML specifications to only those changes required to define reliable messaging. Section 3 of this document proposes minimal changes to the ebXML Message Header Specification [2].

2 Reliability Architecture

2.1 Base Concept

To achieve reliable messaging between Parties, this specification defines a process which enables the Parties' Messaging Systems to communicate with each other using "at most once" semantics.

[For the purposes of this document, the term "*Sender*" means the *Sending Party's Messaging System* which interacts with the underlying message transport, and "*Receiver*" means the comparable Messaging System used by the Receiving Party.]

Reliable Messaging consists of the following fundamental concepts:

- A transmitted message is identified by a **Message Identifier** and a Sender-generated **Recovery Number**. The **Recovery Number** is generated from a Recovery Counter which is unique to the Sender-Receiver pair.
- When a transmitted message is lost due to a system or communication failure, the Sender will re-send the lost message to the Receiver.
- The Receiver receives the re-sent message and determines whether the message is a repeated message or not by using information in the **Message Identifier** and/or a Sender-generated **Recovery Number** located in the Message Header**.**
- If the re-sent message is repeated, the Receiver discards the message. If the message is not repeated, the Receiver stores the message in its persistent storage.

2.2 Features

Reliable Messaging has following features:

- Minimum changes to existing ebXML specifications

To implement reliable ebXML messaging, this specification adds one new element, **Recovery Number,** to the "Message Data" element, and it adds two new elements, **Message Expiration Timestamp** and **Window Count,** to the "Reliable Messaging Info" element in the ebXML Message Header.

- No dependence on specific transport protocols

  Since Reliable Messaging does not use a transport protocol specific function, it works on any transport protocol, such as HTTP or SMTP.

- Simple and lightweight protocol

  Reliable Messaging uses existing ebXML message types (Normal Message, Acknowledgement Message and Error Message) which are defined in [2]. It does not require additional message types.

- High Performance

  Reliable Messaging includes a Sliding Window method for high performance communication between Parties. This technique allows the Sender to batch multiple messages and receive only one Acknowledgement Message in reply.

- No dependence on a TP monitor

  Reliable Messaging does not need any other service which provides a transaction function, such as a TP monitor.

2.3 Message Header Elements used for Reliable Messaging

2.3.1 Message Data

Reliable Messaging uses the **Message Identifier,** which uniquely identifies that message, and a **Recovery Number**. They are placed in the Message Data Element of the Message Header for each message and are defined as shown in Table 2-1.

Table 2-1 Message Data Element

Message Header Element	***Outline Description***
6) Message Data	
a) **Message Identifier**	This is a globally unique identifier for an individual Message. It is a string that consists of the Sender's globally unique address on the Internet (e.g. IP address, URL, etc.), time stamp, sequential number, etc. In Reliable Messaging, it may be used to detect repeated messages by the Receiver. <Note> Need to define detailed format </Note>
b) **Recovery Number**	Integer value which is sequentially incremented (e.g. 1, 2, 3, 4, ...) for each Sender-prepared message to the Receiver. The Sender uses a different Recovery Counter for each Receiver. In Reliable Messaging, it may be used for an effective check for repeated messages by the Receiver. *<Note> May use one Recovery Counter for all*

	Receivers?; Need to define size of Recovery Counter </Note>
c) **Message Creation Timestamp**	<see definition in Message Header specification>
d) **Ref To Message Identifier**	<see definition in Message Header specification>

Each Message Data element includes a **Recovery Number** for an efficient check of repeated messages by the Receiver. The Sender uses individual persistent Recovery Counters to generate **Recovery Numbers** for each Receiver.

2.3.2 Reliable Messaging Info

Reliable Messaging uses a **Message Expiration Timestamp** in the Reliable Messaging Info element to allow the Sender to specify a message lifetime and to support garbage collection. **Window Count** in the Reliable Messaging Info element supports a "Sliding Window" method for the message transfer sequence. They are defined as shown in Table 2-2.

Table 2-2 Reliable Messaging Info Element

Message Header Element	***Outline Description***
10) Reliable Messaging Info	
a) **Guaranteed Delivery**	<see definition in Message Header specification>
b) **Message Expiration Timestamp**	Shows message lifetime when used with the Message Creation Timestamp. If a message expires before it reaches the receiving Party, Sender and Receiver may remove the message from their persistent storage.
c) **Window Count**	Indicates the number of messages in the Window. This element appears only when the "Guaranteed Delivery" element indicates "AtMostOnce" semantics and this message is the last Normal Message in the Window. When the Receiver receives a Normal Message which includes this element, the Receiver returns an Acknowledgment Message or an Error Message to notify the Sender whether or not all messages in the Window were successfully received.

2.4 Message Transfer Sequence

A sequence of messages, or "Window", may be sent and a single acknowledgement message returned to the Sender once all the messages in the Window have been received. With respect to a particular Sender and Receiver pair, transmission of one Window must be completed before another Window may be sent.

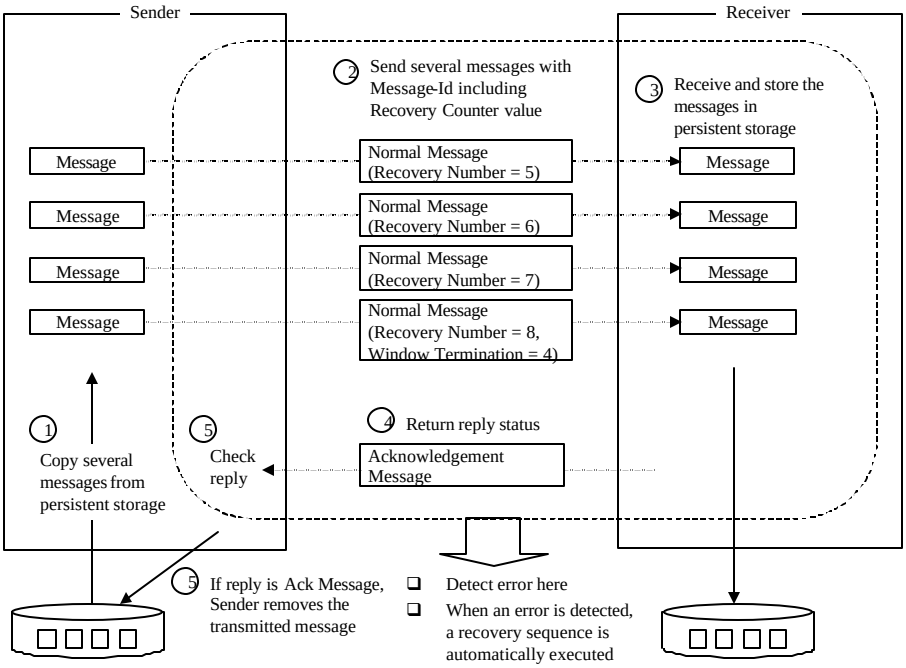


Figure 2-1 Message Transfer Sequence in a Window

Reliable Messaging utilizes persistent storage and a variation of Sliding Window method as shown in the following sequence:

(1) Message copy

Sender initially stores messages passed from the ebXML application in persistent storage, and then copies the stored messages for message transfer.

(2) Sending messages

Sender sends several messages to Receiver before receiving an Acknowledgement Message from the Receiver. Each message is identified by a **Message Identifier** and a **Recovery Number** in the Message Header. Sender notifies Receiver of the end of the Window by using the **Window Count** element in the Reliable Messaging Info element of the last message in the Window.

(3) Receiving and storing message

The Receiver receives and stores the all messages of the Window in persistent storage.

(4) Acknowledgment by Receiver

When the Receiver receives a message which includes the **Window Count** element in the Reliable Messaging Info element, the Receiver compares the **Window Count** value with the number of messages received for this Window. If both numbers are the same, the Receiver returns an Acknowledgment Message to the Sender; otherwise, an Error Message is returned.

(5) Reply check and removal of transferred messages by Sender

Sender checks the Acknowledgement Message from the Receiver. If the reply is the appropriate Acknowledgement Message for the transferred messages, Sender removes the transferred messages from Sender's persistent storage.

<note>

- *Need to define maximum message size in this specification or another specification*
- *Need to define the format and content of Error and Acknowledgement Messages used in Reliable Messaging*
- *Need to define how to split and transfer large files in this specification or another specification*

</note>

2.5 Error Detection

- Transport protocol level error

  When the Sender detects a transport protocol level error (such as an HTTP, SMTP or FTP error), the Sender's recovery handler will execute a recovery sequence.

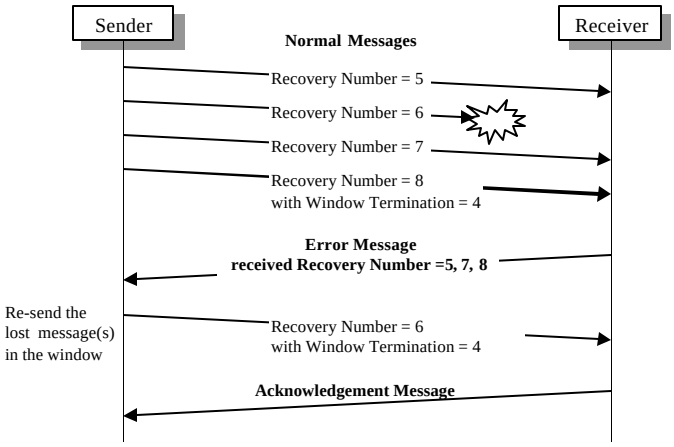
- ebXML Message level error

  When the Sender detects a timeout while waiting for an Acknowledgement Message, the appropriate recovery handler in the Sender executes a recovery sequence.

  When the Sender receives an Error Message from the Receiver, the Sender's recovery handler checks the error status of the Error Message. If the error status indicates the possibility of a lost message, the recovery handler executes a Window Recovery Sequence as shown in the next section. If the error status indicates another error, such as an invalid message header format or an invalid payload format, the recovery handler passes the Error Message to a higher process because such an error is out of the scope of the Reliable Messaging.

2.6 Window Recovery Sequence

Figure 2-2 Recovery Sequence for Lost Message in Window



- Transport protocol level error

  In the case of a transport protocol level error, Sender re-sends all the messages in the Window to the Receiver.

- Acknowledgement Message Timeout

  In the case of an Acknowledgement Message timeout, Sender re-sends all the messages in the Window to Receiver.

- Error Message indicating lost message

  After receiving a **Window Count** field, the Receiver checks for a difference between the number of messages indicated by **Window Count** in last Normal Message and the number of messages received in the Window. If there is a discrepancy, the Receiver returns an Error Message, which indicates which messages have been received in the Window using the message Recovery Numbers. The Sender can then determine which messages were lost and only re-send lost messages to the Receiver.

2.7 Detection of Repeated Messages by the Receiver

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

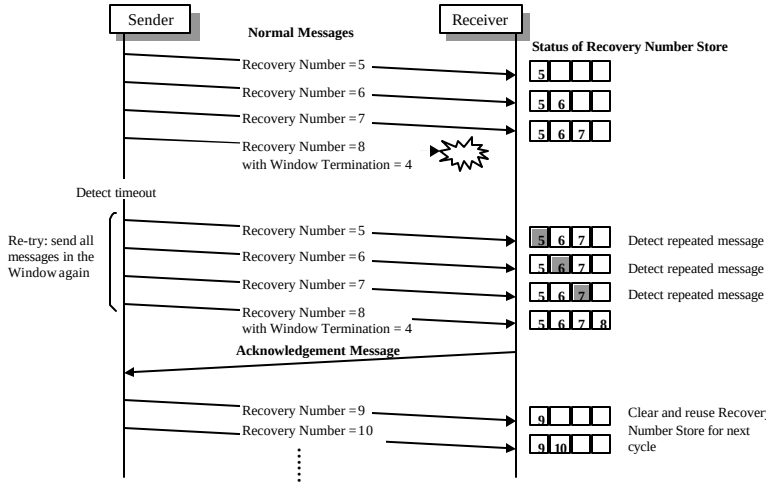


Figure 2-3 Detection of Repeated Messages in a Window

- Receiver creates a Recovery Number Store for each Sender. One Recovery Number Store holds the **Recovery Numbers** for all messages within one Window which are received from a Sender.
- Whenever a message is received, the Receiver compares the message's **Recovery Number** with the all numbers recorded in the Recovery Number Store.
  - If the same number is detected, the message was repeated and the Receiver discards the message
  - If the number is not duplicated, the Receiver records the message's **Recovery Number** in Recovery Number Store and stores the message in its persistent storage
- When the first message of the next Window is received, the Recovery Number Store is cleared and reused for messages in next Window.

2.8 Garbage Collection

Sender and Receiver may automatically remove unused Recovery Counters and Recovery Number Stores to reduce system resources required for Reliable Messaging.

Each Recovery Counter used by the Sender has a Recovery Counter Expiration in persistent storage. Whenever a message is sent, the **Message Expiration Timestamp** from the message's Reliable Messaging Info element is stored in the Recovery Counter Expiration, overwriting the previous value. The Sender may remove the Recovery Counter and its associated Recovery Counter Expiration when all the following conditions are satisfied at same time:

- Recovery Counter for the Receiver has expired using the Recovery Counter Expiration
- Sender is not sending a Normal Message to the Receiver
- Sender is not waiting for an Acknowledgement Message from the Receiver
- There is no subsequent message in persistent storage which should be sent to the Receiver

Similarly on the Receiver side, each Recovery Number Store has Recovery Number Expiration in persistent storage. Whenever a message is received, the **Message Expiration Timestamp** in the received message's Reliable Messaging Info element is stored in the Recovery Number Expiration, overwriting the previous value. The Receiver may remove the Recovery Number Store and its Recovery Number Expiration when the all the following conditions are satisfied at same time:

- Recovery Number Store for the Sender has expired using the Recovery Number Expiration
- The Receiver is not receiving any Normal Messages from the Sender

3 Changes to Existing ebXML Specifications

3.1 Changes to ebXML Message Header Specification v0-5

- Add a **Recovery Number** element to the Message Data element (see "Table 2-1 Message Data Element")
- Add **Message Expiration Timestamp** and "**Window Count** elements to the Reliable Messaging Info element (see "Table 2-2 Reliable Messaging Info Element").

3.2 Changes to Other ebXML Specifications

There are no changes to other ebXML specifications.

4 Open Issues

The following issues will need further discussion in the Transport, Routing and Packaging Team.

4.1 Reliability in Routing through Intermediate Nodes

[TBD]

4.2 Trading Partner Agreement (TPA) Considerations

[TBD]

4.3 Codeset Conversions

[TBD]

4.4 Time synchronization

[TBD]

4.5 Definition of terms

[TBD]

5 References

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

[2] ebXML Transport, Routing and Packaging: Message Header Specification, version 0-5, 26 May 2000

[3] ebXML Transport, Routing and Packaging: Message Envelope Specification, version 0-6, 24 July 2000

6 Acknowledgements

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

7 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