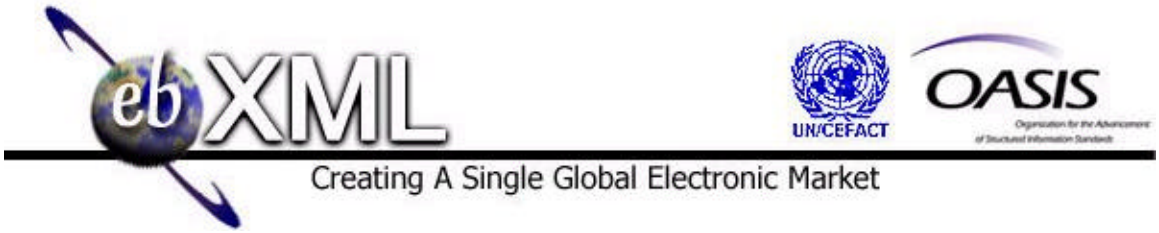

ebXML Transport, Routing & Packaging Reliable Messaging Specification

Working Draft 16-August-2000

This version:
ebXML Reliable Messaging Specification v0-074.doc
Latest version:
N/A
Previous version:
v0-073
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: Editor comments like this will be removed before this material is incorporated into the Messaging Service Specification (or the text will be moved to some non-normative part of the specification).

Editor Note 2: Significant changes since the previous version include:
- changed "Window" to "RM-Group" (Reliable Messaging Group)
- deleted section 2.8, Garbage Collection, since it is not normative
- comments from San Jose meetings

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].

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) are 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 which 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.

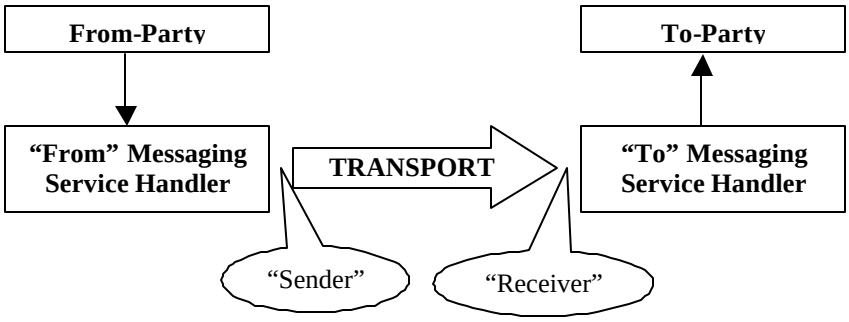

Figure 2-1: Simple Message Transmission

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. With respect to a particular underlying transport, the MSHs can be identified as a "Sender" and a "Receiver".

2) A message is identified by its globally unique **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. One Routing Header Data Element is 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.

6) One or more messages between the Sender and Receiver on the same Sender-Receiver-Transport triplet may be sent using Reliable Messaging semantics. This sequence messages is termed an "RM-Group".

Editor Note 3: The Sender will need to know how many reliable messages will form the RM-Group. This could be determined from the TPA, or from characteristics of the underlying transport, or in negotiations with the "From-Party", or…? If there is interaction with the Receiver to determine this value, then some normative interface is needed.

Editor Note 4: Restricting messages to the same transport for a particular RM-Group will simplify error management. At a later date, we may consider how to transmit an RM-Group over several transports, allow ACKs on an alternate transport, etc.

7) The beginning of an RM-Group is indicated by setting the **DeliverySemantics** field to **AtMostOnce** in the first message in the group; all subsequent messages in the RM-Group will also have this field set. The end of the RM-Group is indicated by setting the **RM-Group Count** field to greater than 0, and all messages in the group except the final message will have this field set to 0.

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

9) The Sender sends messages in the RM-Group to the Receiver, and indicates the last message in the RM-Group by setting the RM-Group Count field to the number of messages in the RM-Group. The Receiver compares this value with the number of messages actually received and, if they are equal, signals a normal completion of the RM-Group transmission back to the Sender by sending an "RM-Group Complete Acknowledgement" message. If fewer messages were actually received than the number shown in the RM-Group Count field, the Receiver sends an error message to the Sender and the Sender resends (at least) the missing messages.

10) Within an RM-Group, the Receiver can determine whether a received message is a duplicate message or not by using the globally unique **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.

Editor Note 5: There is a need to describe the relationship between this persistent store and the higher-level application, which uses the ebXML Messaging System Handler interface. Probably this discussion should happen in the material describing how the higher-level service invokes MSF functions and receives reports.

11) To detect loss of the last message in an RM-Group, the Sender sets a time-out for that message. If the final transmitted message in an RM-Group is lost due to system or communication failure, a Sender will re-send this last message [TBD] times to the Receiver before reporting failure to the From-Party.

Editor Note 6: How many times should the Sender attempt this re-try before reporting a failure to the From-Party? Is there a TPA element which describes this recovery semantics?

Editor Note 7: Should there be an inquiry function in the Messaging Service Handler, so that the Sender can obtain information from the Receiver?

12) An "RM-Group Complete Acknowledgement" Messaging Service level acknowledgement is sent from the Receiver to the Sender for every received message which has an RM-Group Count value greater than 0.

Editor Note 8: If the Sender is supposed to report completion of this RM-Group to the From-Party, how is the Sender notified to do this ? in the TPA? by the From-Party?

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

Reliable Messaging uses the **DeliverySemantics** field to either initiate an RM-Group or continue an RM-Group in transmission to the Receiver.

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 9: There is a need to identify the particular Messaging Service instance which is processing the message on behalf of the From-Party or To-Party. A **MessageServiceId** is used for this.*

Editor Note 10: The mandatory 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).

Editor Note 11: There is a request to add an RM-Group Id field to identify a particular RM-Group, plus specific flags to denote the start/end of the RM-Group… Do we really need these?

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)

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

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 Receiver may use this Sequence Number to check for repeated messages, or the Receiver may use the globally unique **MessageId**.
RM-Group Count	If this message is the last Message in the RM-Group, this field indicates the number of messages in the RM-Group. For other than the last message in the RM-Group, the value of this field is set to "0".

2.3 Message Transfer Sequence

A sequence of reliable messages, or "RM-Group", SHALL be sent and a single acknowledgement message returned to the Sender once all the messages in the RM-Group have been received by the Receiver. As messages are received, the Receiver MAY process them appropriately, usually by passing the message to the higher-level "To-Party".

With respect to a particular Sender, Receiver and transport triple, transmission of one RM-Group SHALL be completed before another RM-Group may be sent.

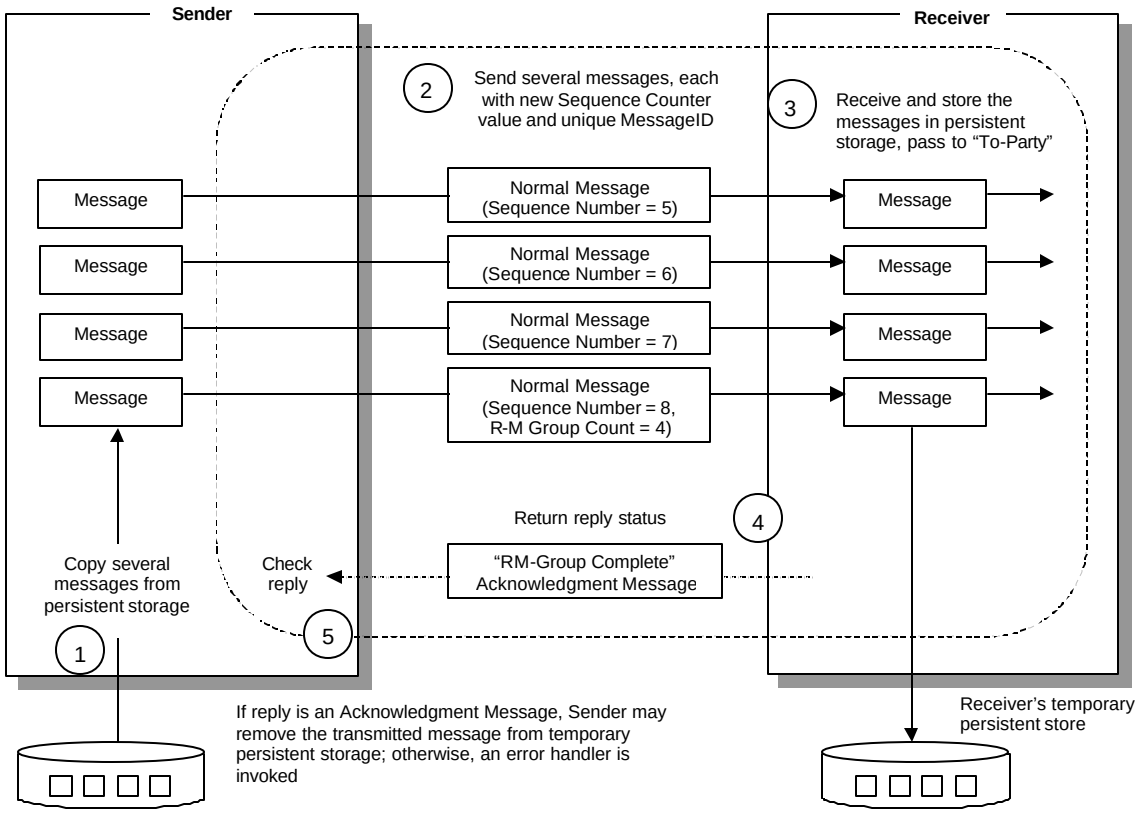

Figure 2-2: Reliable Message Transfer Sequence

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 messages for message transfer.

(2) Sending messages

All messages in the RM-Group have DeliverySemantics = AtMostOnce set, and receipt of a message with this value notifies the Receiver of the start of the RM-Group. Sender sends several messages to the Receiver before receiving an Acknowledgement Message when the RM-Group is complete. Each message is identified by a **Message Identifier** in the Message Header and a **Sequence Number** in the Routing Header. Sender notifies Receiver of the end of the RM-Group by setting a non-zero value for the **RM-Group Count** element in the last message of the RM-Group.

(3) Receiving and storing message

The Receiver receives and stores the all messages of the RM-Group into persistent storage and processes messages appropriately.

(4) Acknowledgment by Receiver

When the Receiver receives a message which includes a non-zero value for the **RM-Group Count** element in the Routing Header Data Element, the Receiver compares the **RM-Group Count** value with the number of messages received for this RM-Group. If both numbers are the same, the Receiver returns an "RM-Group Complete" Acknowledgment Message to the Sender; otherwise, an Error Message is returned. The Error Message shows the Sequence Numbers for the messages actually received by the Receiver for this RM-Group, and the Sender uses this information to decide which messages must be re-sent to the Receiver.

(5) Sender checks reply and removes transferred messages by

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

2.4 Recovery Sequence for Lost Messages

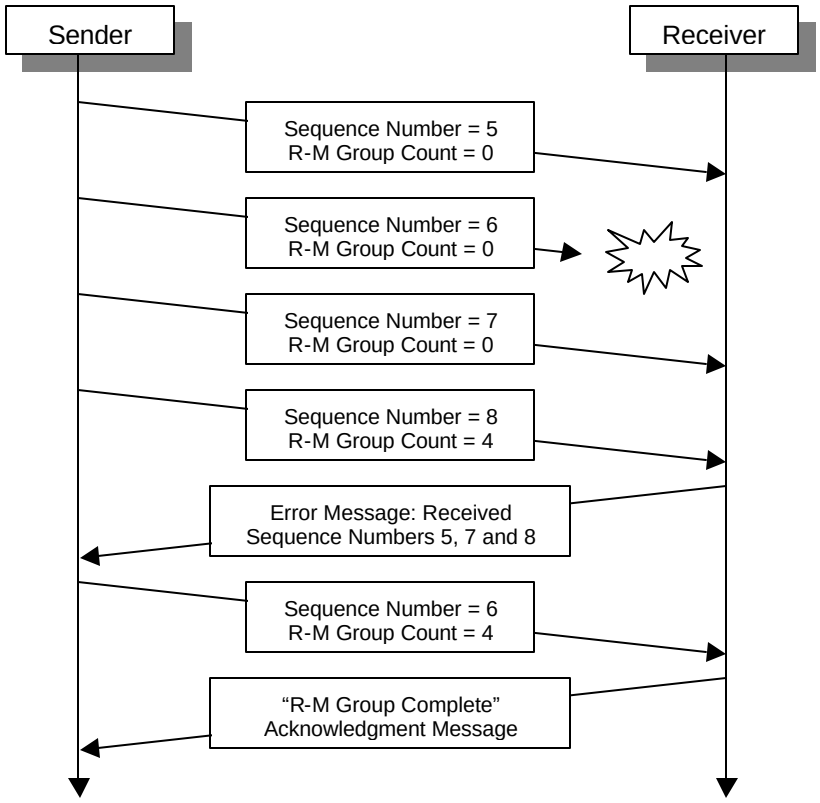

Figure 2-3: Recovery Sequence for Lost Messages

After receiving an **RM-Group Count** value that is greater than 0, the Receiver checks for a difference between this value and number of messages actually received in the RM-Group. If there is a discrepancy, the Receiver returns an Error Message to the Sender which indicates Sequence Numbers for the messages which were received. The Sender may then determine which messages were lost and only re-send lost messages to the Receiver (or, alternatively, the Sender may resend all messages in the RM-Group since the Receiver will always discard duplicates).

Editor Note 12: TBD: What happens at the Sender if the Error Message or Acknowledgement Message is lost?

2.5 Detection of Repeated Messages by the Receiver

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**. Figure 2-4 shows detection of duplicated messages when the final message of an RM-Group is lost.

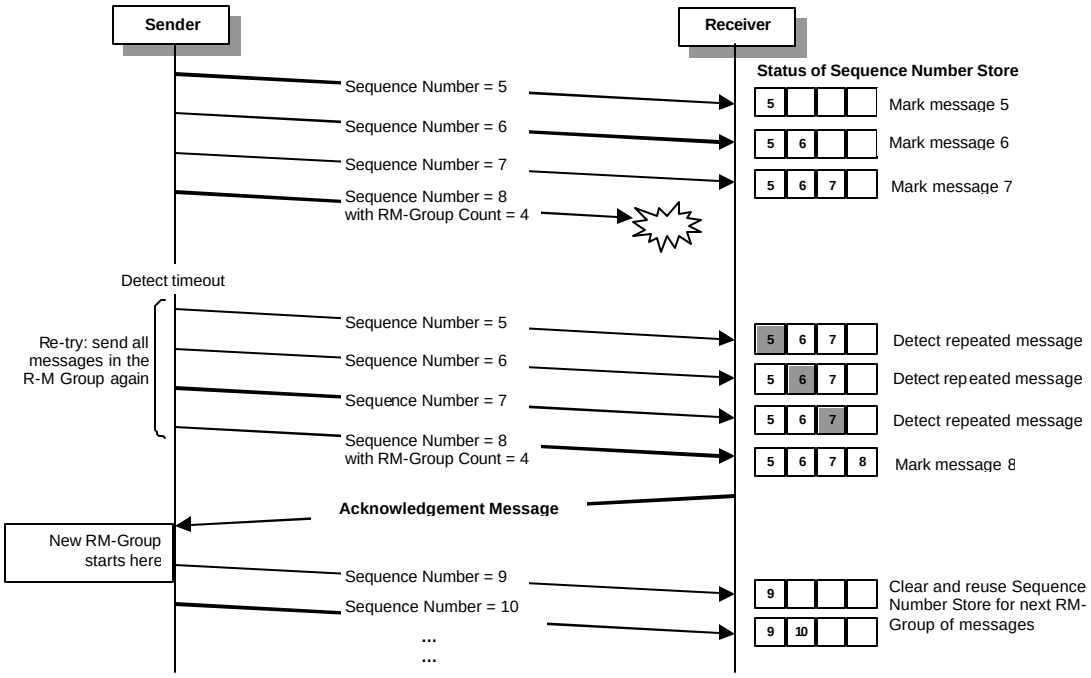

Figure 2-4: Detection of Repeated Messages in an RM-Group

- Receiver creates a Sequence Number Store for each Sender-Transport, which holds the **Sequence Numbers** for all messages within one RM-Group which are received from a Sender on a particular transport.
- Whenever a message is received, the Receiver compares the message's **Sequence Number** with the all numbers recorded in the Sequence Number Store.
  - If the same number is detected, the message was repeated and the Receiver may discard the message.

- If the number is not duplicated, the Receiver records the message's **Sequence Number** in Sequence Number Store, stores the message in its persistent storage and begins processing of the message for the Receiver's higher-level process.

- When the first message of the next RM-Group is received, the Sequence Number Store is cleared and reused for messages in next RM-Group.

2.6 Reliable Messaging Acknowledgement and Error Messages

2.6.1 General

When the Sender or 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.

When the Sender detects a timeout while waiting for an Acknowledgement Message from the last message in an RM-Group, the appropriate recovery handler in the Sender executes a Messaging Service recovery sequence. This recovery sequence SHALL re-send the final message to the Receiver [TBD] times before returning an error message to the From-Party. The format of the re-sent message is exactly the same as the original message.

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 which are using Reliable Messaging semantics for this RM-Group.

Each RM-Group contains messages that are 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 first message of the RM-Group
- *Original-Recipient* means the "To" PartyID contained in the first message of the RM-Group

> *Editor Note 13: 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 RM 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 first message of the RM-Group
- ServiceInterface and Action are not present in the message
- RefToMessageId = MessageId of the first message in the RM-Group
- DeliverySemantics = "Unspecified"

2.6.3 Error: Missing Message(s) in RM-Group

The MessageType SHALL be ”Error”.

The Payload for this error message SHALL contain the two characters “MM”, a space, and a comma-delimited list of the SequenceNumbers corresponding to messages which were received by the Receiving Messaging Service Handler for this RM-Group. There is always at least one SequenceNumber contained in the list, since the error message is generated when the last message in the RM-Group is received.

The Sender SHALL respond to this message by re-sending messages in the RM-Group. The Sender MAY choose to resend either all the messages or those messages which are not identified by the SequenceNumbers in the payload of the error message. The Receiver will check again on the completeness of the RM-Group by using the **RM-Group Count** field:

- If there is one message to re-send, that message is sent exactly as it was sent previously, and the **RM-Group Count** field is set to the number of messages in the RM-Group.
- If there are more than one message to re-send, all messages except the last message are sent exactly as they were previously sent (**RM-Group Count** field set to 0), and the final message has its **RM-Group Count** field set to the number of messages in the RM-Group.

2.6.4 RM-Group Complete Acknowledgement

The Receiver’s Messaging Service Handler sends this message to the Sender’s Messaging Service Handler when the final message of a RM-Group is received, and the correct number of messages for this RM-Group have been 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 RM-Group until a current RM-Group has been correctly received, there is no possibility that the Receiver will count incorrect messages when determining if all RM-Group messages were properly received.

The MessageType SHALL be “Acknowledgement”.

There is no Payload.

Editor Note 14: The receiving MSH knows that this is an RM-Group ACK because it is addressed directly to the MSH and references the beginning of the open RM-Group. Optionally, we might specify a flag or field to denote clearly that this message is a MSH-level ACK for RM.

Editor Note 15: What if the Sender never receives the ACK from the Receiver? The Receiver has processed all messages in the group and should be ready to start a new group. However, the Sender will timeout and (possibly) re-send the last message. We need to instruct the Receiver on what to do if it gets duplicate final messages for a group. Recognizing this second transmission as a duplicate and ignoring it won’t work, since the Sender is expecting an ACK to the message to complete the group.

3 Changes to Current ebXML Specifications

Editor Note 16: This section will be deleted when RM material is moved into the Messaging Service Specification.

3.1 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)

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

How is the MSH-level ACK from the final Receiver prepared and sent back to first Sender? Should there even be any Reliable Messaging semantics between subsequent Sender-Receiver pairs in the network?

What does an intermediate node do if a link is not available to the next node, yet previous links have already concluded that the message was reliably delivered to the subsequent link? (answer: must break the transmission… but what happens if there has been some partial transmission on reliable links in the network?)

"RM-Group-ACK" is different from "final-node-ACK"…

What does From_Party do when it never gets a higher-level message ACK from the network?

4.2 Trading Partner Agreement (TPA) Considerations

[TBD] discuss TPA terms used in Reliable Messaging.

4.3 Definition of terms

[TBD] eventually add the new Reliable Messaging Terms and move to Glossary

5 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 0-1, 11 August 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 meetings 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