Web Services Reliable Messaging TC WS-Reliability

Working Draft 0.84, 15 December 2003

Document identifier:

wd-web services reliable messaging tc-ws-reliability-0.84

Location:

http://www.oasis-open.org/(TBD)/(TBD)/

Editor:

Kazunori Iwasa, Fujitsu Limited <kiwasa@jp.fujitsu.com>

Abstract:

Web Services Reliability (WS-Reliability) is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions, and is independent of the underlying protocol. This specification contains a binding to HTTP.

Status:

This document is updated aperiodically on no particular schedule.

Committee members should send comments on this specification to the wsrm@lists.oasis-open.org list. Others should subscribe to and send comments to the wsrm-comment@lists.oasis-open.org list. To subscribe, send an email message to wsrm-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Web Services Reliable Messaging TC web page (http://www.oasis-open.org/committees/wsrm/).

The errata page for this specification is at http://www.oasis-open.org/committees/wsrm/xxx. (TBD)

Table of Contents

1 Introduction (Needs Updating)

1.1 Purpose of WS-Reliability

The purpose of WS-Reliability is to address reliable messaging requirements, which become critical, for example, when using Web Services in B2B applications. SOAP [SOAP1.1] or [SOAP1.2] over HTTP [RFC2616] is not sufficient when an application-level messaging protocol must also address reliability and security. While security is getting traction in the development of Web Services standards, reliability is not. This specification is intended as an initial proposal for defining reliability in the context of current Web Services standards. The specification borrows from previous work in messaging and transport protocols, e.g., SOAP, and the ebXML Message Service [ebMS]. It proposes appropriate modifications to apply this work to Web Services.

1.2 Scope and Definition of Reliable Messaging

The focus of this specification is on the SOAP layer and envelope. The OASIS WS-RM TC does not presume to cover all aspects of Reliable Messaging. Several fundamental questions on reliability need to be addressed in subsequent work, and are not addressed in this specification:

- Assuming that reliability objectives cannot always be guaranteed or attainable, should a reliability contract include advanced quality of service elements (which may translate into specifying quantitative thresholds, e.g. how large a message archive or time period a duplicate check should cover)?
- Beyond the specified qualities of message delivery (guaranteed delivery, duplicate elimination, and message ordering), should reliability also define the degree of synchronization between sender and receiver applications (i.e. the degree to which both sender and receiver parties will have same understanding of whether a request was properly received or not)?

Within the scope of this specification, the following features are investigated:

- Asynchronous messaging at the application level.
- Three reliability features: Guaranteed Delivery, Duplicate Elimination, and Guaranteed Message Ordering.

In the current specification, we will define reliable messaging as the mechanism supporting the following requirements at the application level:

- Guaranteed message delivery, or At-Least-Once semantics.
- Guaranteed message duplicate elimination, or At-Most-Once semantics.
- Guaranteed message delivery and duplicate elimination, or Exactly-Once semantics.
- Guaranteed message ordering, within a context delimited using a group id.

1.3 Limits of Scope

The out of scope of this specification are:

- Application level synchronous messaging. Synchronous messaging applications that require knowledge of the error status immediately rather than waiting for the messaging layer to resend the message when an error returns are out of scope of this specification.
- Routing. Other mechanisms can be used in conjunction with an implementation of this specification.
- Security. Other mechanisms can be used in conjunction with an implementation of this specification.

1.4 Goal of This Specification

The goal of this specification is to define:

- A mechanism to guarantee message delivery and its expression in SOAP messages.
- A mechanism to eliminate duplicate messages and its expression in SOAP messages.
- A mechanism to guarantee received message order (within a context) and its expression in SOAP messages.

1.5 Notational Conventions

This document occasionally uses terms that appear in capital letters. When the terms "MUST", “REQUIRED”, “SHALL”, "SHOULD", "RECOMMENDED", “MAY”, “OPTIONAL”, "MUST NOT", “NOT REQUIRED”, “SHALL NOT”, and "SHOULD NOT" appear capitalized, they are being used to indicate particular requirements of this specification. An interpretation of the meanings of these terms appears in [RFC2119].

1.6 Relation to Other Specifications

(1) W3C SOAP1.1/1.2:

SOAP1.1 [SOAP1.1] and SOAP1.2 [SOAP1.2] are the base protocol for this specification. This specification defines extensions to SOAP Header and Body elements.

(2) OASIS ebXML Message Service Specification 2.0:

The reliable message mechanism defined in the ebXML Message Service Specification 2.0 [ebMS] is implemented in a number of products and open source efforts, many of which have undergone interoperability testing. WS-Reliability borrows from this technology.

(3) OASIS WS-Security:

This specification can be used with WS-Security [WSS] when that effort is completed in OASIS.

(4) WS-I Basic Profile 1.0

(TBD)

1.7 Examples of Messages Compliant with WS-Reliability

(To be added later.)

1.8 Terminology

Reliable Messaging:

The set of mechanisms and procedures required to send messages reliably. This includes the processing of Acknowledgment messages, re-sending of messages, duplicate message elimination, and message ordering.

Reliable Messaging Processor (RMP):

A module capable of processing and enforcing Reliable Messaging as described in this specification.

Message Delivery:

Message delivery is the action of transferring the responsibility of processing further a message, from the RMP and onto the next processor entity. This action marks the end of the RMP processing for this message. The time at which this action occurs must be clearly identifiable so that the next message processor can always establish in which order two deliveries are made.

Examples of message delivery are:

- pushing the message in a queue accessible by an application,
- calling back an application component,
- storing the message in a database where it is accessible by the next processor.

Reliable Message:

A message for which the sender requires some level of reliable delivery, typically requiring acknowledgment for notification of delivery.

PollRequest Message:

(TBD)

Acknowledgment Message:

A signal message sent by a SOAP node to notify the initial sender of delivery of the message. The Acknowledgment message indicates the receiver has received the message and the sender is no longer responsible to persist the message. (i.e. the receiver now has responsibility for message persistence)

An Acknowledgment Message is containing an RM:Response element referring to at least one previous message (and containing no RM:Fault element).

An Acknowledgment Message means that the acknowledged message has been successfully delivered, meaning that it has satisfied all the reliability requirements placed on it for delivery, and that the RMP, having made the message available to its next processor, is no longer responsible for processing it further.

Reliable Messaging Fault Message:

A message to notify the sender of the reliable message that there was a failure to receive or process the message.

Reliable Messaging Reply (RM-Reply):

A message which is either Acknowledge Message or Reliable Messaging Fault message.

Response RM-Reply Pattern:

We say that a response RM-Reply pattern is in use if the outbound Reliable Message is sent in the underlying protocol request and the Acknowledgment message (or Fault message) is contained in the underlying protocol response message corresponding to the original request.

Callback RM-Reply Pattern:

We say that a callback RM-Reply pattern is in use if the Acknowledgment message (or Fault message) is contained in an underlying protocol request of a second request/response exchange (or a second one-way message), operating in the opposite direction to the message containingthe outbound Reliable Message.

Polling RM-Reply Pattern:

We say that the polling RM-Reply pattern is being used if a second underlying protocol request is issued in the same direction as the one containing the outbound Reliable Message to act as a request for acknowledgment. The Acknowledgment message (or Fault message) is contained in the underlying protocol response to this request. This polling pattern is expected to be used in situations where it is inappropriate for the sender of reliable messages to receive underlying protocol requests.

2 Messaging Model

The following sections provide an overview of the WS-Reliability Messaging Model.

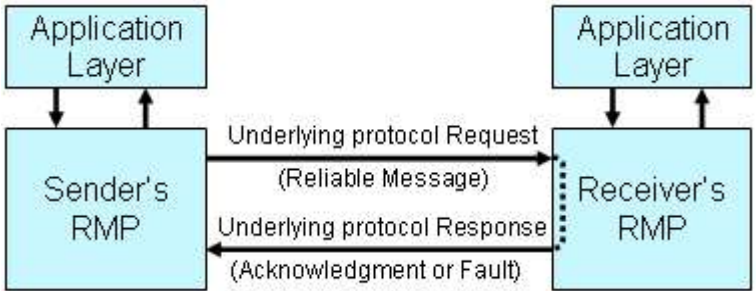
2.1 Overview of Messaging Model

In the Reliable Messaging Model described in this document, the sender node sends a message to the receiver node directly (i.e., intermediaries are assumed to be transparent in this specification). The receiver node sends back an Acknowledgment message or Fault message to the sender node. There are three ways how to send back Acknowledgment message or Fault message as described as follows:

(1) Request/Response Messaging Model

With this model, the outbound Reliable Message is sent in the underlying protocol request and the Acknowledgment Message is contained in the underlying protocol response message corresponding to the original request. The figure 1 shows this model.

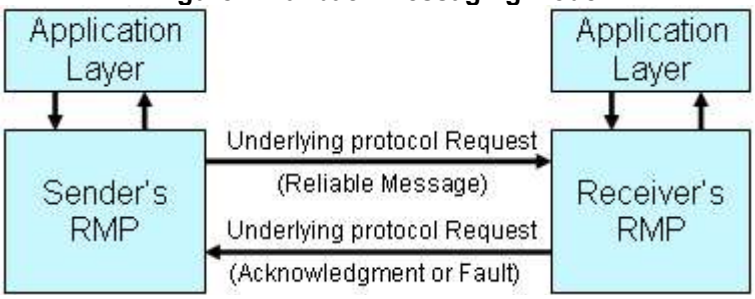
Figure 1 Request/Response Messaging Model



(2) Callback Messaging Model

With model, Acknowledgment Message is contained in an underlying protocol request of a second request/response exchange (or a second one-way message), operating in the opposite direction to the message containing the outbound Reliable Message. The figure 2 shows this model.

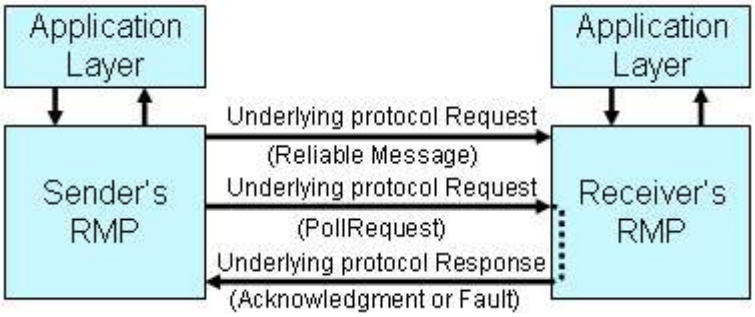
Figure 2 Callback Messaging Model



(3) Poll Messaging Model

With this messaging model, a second underlying protocol request is issued in the same direction as the one containing the outbound Reliable Message to act as a request for acknowledgment. The Acknowledgment Message is contained in the underlying protocol response to this request. This messaging model may be used in situations where it is inappropriate for the sender of reliable messages to receive underlying protocol requests. The figure 3 shows this model.

Figure 3 Poll Messaging Model



2.2 Message Identifier

Message Identifier is a combination of GroupId element and a possible SequenceNumber element. Every Reliable Message MUST contain a globally unique Message Identifier. The Acknowledgment message MUST contain a reference to the Message Identifier of the acknowledged message, confirming that the receiver SOAP node has received the message. Presence of SequeceNumber indicates the Group has more than one message.

2.3 Retry

If the SOAP node sending a Reliable Message does not receive an Acknowledgment message, that sender MUST resend the same message with same Message Identifier to the receiver node until (1) the sender gets an Acknowledgment message from the receiver, or (2) a specified number of resend attempts have been made without success. If the sender SOAP node fails to send the message (i.e., no Acknowledgment is received), the node MUST report the error to the application layer in some way.

2.4 Message Persistence (Update after resolving timing issues)

With Reliable Messaging, the sender is REQUIRED to persist the message until one of the following conditions are met:

- Receipt of an Acknowledgment message from receiver, indicating the message has been successfully delivered.
- All retry attempts have failed, and a delivery failure is reported to the application layer.
- The span of time indicated by the ExpiryTime element has expired.

The receiver MUST persist out of order messages to support Guaranteed Message Ordering.

The receiver MUST persist the Message Identifier to support Duplicate Elimination. Both sender and receiver MUST behave as if there was no system failure or system down after recovery.

2.5 Guaranteed Delivery

To deliver a message from a sender RMP to a receiver RMP without failure, or to report failure to the sender's application. To realize guaranteed delivery,the message MUST be persisted in the sender RMP until it delivery to it's receiver is acknowledged, or until the ultimate failure is reported to it's requester. There is a requirement of the underlying transport protocol that the message MUST be transported without corruption.

When message persistence is lost for any reason, it is no longer possible to continue to guarantee message delivery. Since the reliability of message persistence is A property of the system implementation, the conditions under which guaranteed message delivery holds is also a property of the system implementation..

Example 1). A PC Server may use a HDD for it's persistent Storage, and those messages persisted in the HDD are reliably maintained even if the the system software crashes and the system is rebooted. However, if the HDD itself crashes, it is no longer possible to guarantee message delivery

Example 2) . A message persisted in a mobile phone may be lost when it's battery is detached. In this case, message delivery is only guaranteed by proper battery maintenance of the mobile phone.

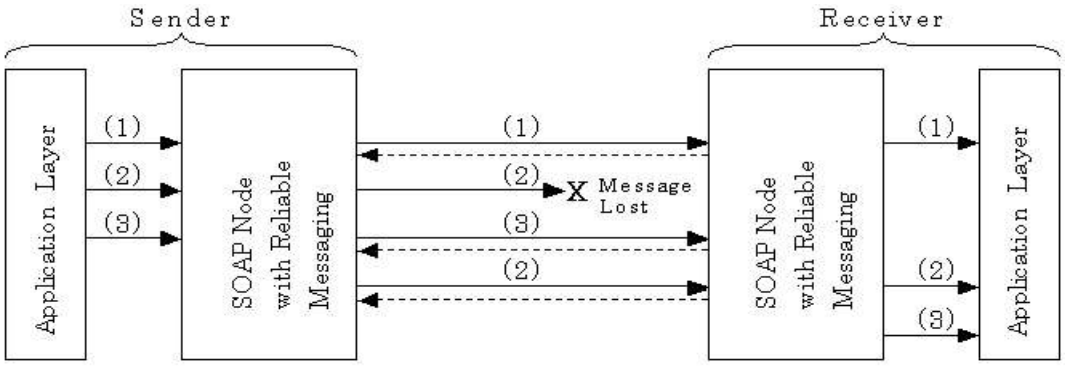
2.6 Duplicate Elimination

A number of conditions may result in transmission of duplicate message(s), e.g., temporary downtime of the sender or receiver, a routing problem between the sender and receiver, etc. In order to provide at-most-once semantics, the ultimate receiver MUST eliminate duplicate messages. Messages with the same Message Identifier MUST be treated as duplicates.

2.7 Guaranteed Message Ordering

Some applications will expect to receive a sequence of messages from the same sender in the same order these messages were sent. Although there are often means to enforce this at the application layer, this is not always possible or practical. In such cases, the messaging layer is required to guarantee the message order. This specification defines a model described in Figure 3 to meet this requirement. When the sender application sent three messages (1), (2), and (3) with Guaranteed Message Ordering, Receiver's RMP MUST guarantee the message order when it makes those messages available to the receiver's application. With the case of Figure 3, the receiver's RMP received message (1) and (3), the receiver's RMP makes message (1) available to the application, but it persists message (3) until message (2) is received. When receiver's RMP received message (2), it makes message (2) and (3) available to the application in order.

Figure 3 Ordering Model



299

300

301 2.8 Sequence Number (Needs updating)

302 A sequence number mechanism is used to track and enforce the order of a sequence of
303 messages having a common grouping identifier value. Such a mechanism has been widely used
304 in the past. In the Figure 3 above, assume the sender application layer generates three
305 messages in order of (1), (2), and (3). The sender SOAP node, with the message ordering
306 function enabled, sends those messages in order of (1), (2), and (3), sequentially and
307 asynchronously, with respective sequence numbers 1, 2, and 3. If the message (2) was not
308 properly received for any reason, the sender will resend the (2) message after a timeout has
309 occurred. The receiver's SOAP node will finally receive these messages as a sequence: (1), (3),
310 and (2). The receiver SOAP node, with the message ordering function enabled, may provide the
311 application layer with message (1), but not (3). Sequence numbering allows the receiver node to
312 easily detect a missing message in a sequence, that is (2), as soon as receiving (3). This
313 condition is recognized by the receiver when the sequence numbers of the messages it receives
314 are not contiguous (e.g., 1, 3, 2). The receiver SOAP node will wait for a message with sequence
315 number 2, and then provide message (2) and then message (3) to the application layer, in order.
316 This behavior can be subject to variants and additional rules to deal with specific failure use
317 cases, such as when a node cannot deliver the proper-sequence of messages due to a message
318 being lost.

319

320 2.9 Attachments

321 When this spec is used with W3C note SOAP messages with Attachments specification, the
322 following rules MUST be met:

323 1) The first MIME part MUST include whole SOAP envelope with WS-Reliability header
324 elements.

325 2) The charset of the Content-Header of the first MIME part MUST be either UTF-8 or UTF-16.

326 3) Zero or more additional MIME parts MAY be included in a reliable message.

327 4) The receiver RMP MUST make available, to the receiving application, all MIME parts in a
328 reliable message

3 Message Format

Figure 4 shows the structure of WS-Reliability elements embedded in the SOAP Envelope.

Figure 4 Structure of WS-Reliability elements

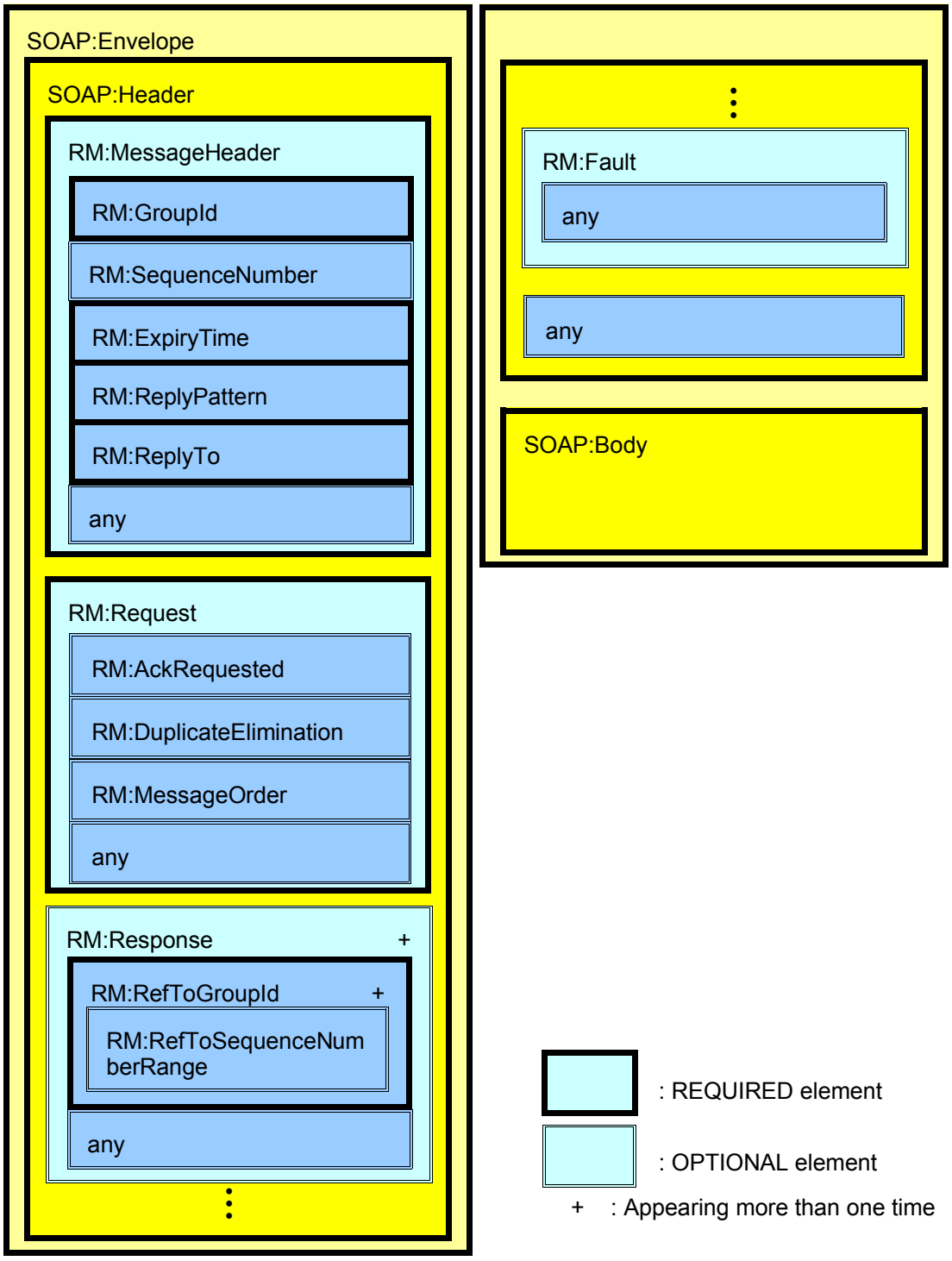
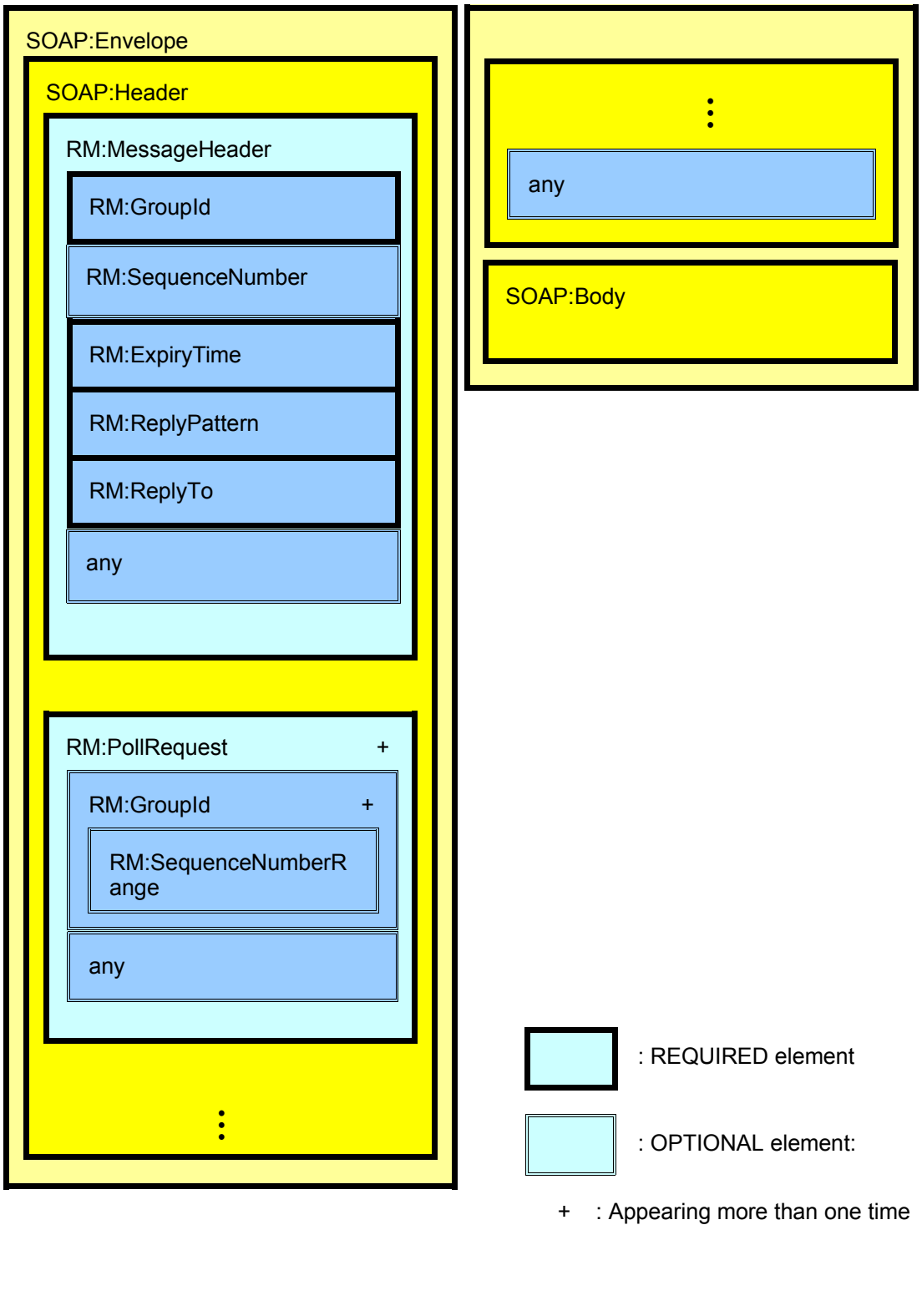


Figure 5 shows the structure of PollRequest message embedded in the SOAP Envelope.

Figure 5 Structure of PollRequest message elements



393

394 The namespaces [XML namespaces] for reliable messaging defined in this specification are:

395 http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1 for SOAP1.1 and

396 http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.2 for SOAP1.2

397

398 If there are additional elements that are not described in this specification present in a message,
399 the Reliable Messaging Processor MUST ignore those elements.

400 In a reliable message, the following four elements are direct children of SOAP Header:

401 - **MessageHeader** element

402 - **Request** element

403 - **PollRequest** element

404 - **Response** element

405 - **Fault** element

406

407 3.1 MessageHeader Element

408 The MessageHeader element includes basic information to be used for a reliable message. This
409 element includes the following attributes and child elements:

410 - SOAP **mustUnderstand** attribute with a value of “1”

411 - **GroupId** element

412 - **SequenceNumber** element

413 - **ExpiryTime** element

414 - **ReplyPattern** element

415 Table1 MessageHeader Element

Required	Yes
Value	None
Attributes	mustUnderstand
Child elements	GroupId SequenceNumber Timestamp ExpiryTime ReplyPattern
Fault	InvalidMessageHeader

416

Example 4 shows an example of a MessageHeader element.

Example 4 MessageHeader Element

(Example are included later)

3.1.1 GroupId Element

This REQUIRED element is to identify a sequence of messages, where each sequence is of length 1 or more. This element MUST have a globally unique identifier as its value. The syntax of this identification is URI, as defined in [RFC2396]. It is RECOMMENDED to use a syntax of Message-ID, as defined in [RFC2392]. This element contains the following attributes:

- a **removeAfter** attribute

Table2 GroupId Element

Required	Yes
Value	RFC2396 *See 3.1.1 for details
Attributes	removeAfter
Child elements	None
Fault	InvalidGroupId

(1) removeAfter attribute

This is an OPTIONAL attribute. This attribute is used to specify the time the GroupId can be removed from the RMP tracking mechanism for GroupId and SequenceNumber elements. Both sender and receiver MUST maintain the value of a GroupId element until either one of the following two events occur:

- The sender sends a Message with the value of “End” in the status attribute.

- The time specified in the removeAfter attribute has passed.

The format MUST be expressed as UTC and MUST conform to a [XML Schema] dateTime. If omitted, the value SHOULD be considered as ‘forever’.

3.1.2 SequenceNumber Element

The SequenceNumber element is a REQUIRED element for Group of more than one message. The value of this element MUST be unique within the same GroupId, and the combination of GroupId and SequenceNumber MUST be globally unique to be used for Message Identifier.

When a message includes a MessageOrder element, the SequenceNumber element is used for guaranteeing the message order within the group of messages specified by the same GroupId value. In other words, the sequence of numbered messages that the receiver node presents to the application MUST be in the same order as the sequence of numbered messages that the sender application has produced, within the group of messages having the same GroupId value.

When the sender requests Guaranteed Message Ordering, the sender MUST use Guaranteed Message Delivery and Duplicate Elimination for that message as well. In particular, the sender MUST include both an AckRequested element and a DuplicateElimination element, as well as the MessageOrder element for Guaranteed Message Ordering.

This element includes the following attribute:

- a **status** attribute

If the MessageOrder element appears in the message sent, the receiver of the message is REQUIRED to make messages available to the application layer only after all messages with lower sequence number with the same GroupId have been made available to the application. Example 5 illustrates this:

Example 5 SequenceNumber Element

(Example will be added later, when the schema is decided)

When a sender node communicates with a receiver node across several GroupId values, the sender MUST maintain an independent counter of the value of SequenceNumber for each GroupId. When sending a message containing a MessageOrder element with a new GroupId, the sender is REQUIRED to start with “0” for the SequenceNumber element in the GroupId.

The value of SequenceNumber MUST conform to [XMLSchema] unsignedLong. For the initial message with a specific GroupId that is sent to the receiver, the SequenceNumber value MUST be “0”. After the initial message has been sent to the receiver, the sender MUST increment the value by one for each message sent. When the value of a SequenceNumber reaches the maximum value, the sender MUST generate a new GroupId for any following messages. This begins a new sequence that could overlap with the old in rare circumstances. From the receiver's perspective, no link exists between the two sequences. To improve the chances that the message ordering is maintained across this change, the sender SHOULD wait until all Acknowledgment messages have been received for the old GroupId before starting the new sequence.

Table3 SequenceNumber Element

Required	No *See 3.1.2 for details
Value	unsignedLong
Attributes	status
Child elements	None
Fault	InvalidSequenceNumber

(1) status attribute

This OPTIONAL attribute is used to specify status of the group of messages. When this attribute is present, its value MUST be one of the following three:

- **Start**: Indicating the message is the first message for a group of messages.
- **Continue**: Indicating the message is in the middle of a group of messages.
- **End**: Indicating the message is the last message for a group of messages.

The sender node MUST send a very first message, to guarantee the order, with “Start” for this attribute. Also, the sender MUST send subsequent messages for the same series of messages with “Continue”, until the message sent is the last one for the series of messages, for which case the value MUST be “End”. When omitted, the default value for this attribute is “Continue.”

NOTE: Because delivery between the Reliable Messaging Processor and the application is not specified, this is not a complete guarantee of ordering to the application.

3.1.3 ExpiryTime Element (Modify after resolution)

The ExpiryTime element is used to indicate the ultimate time after which the receiver RMP MUST NOT deliver a received message to the application. This is an REQUIRED element. After a message has been sent for the first time, the value of the ExpiryTime in a message MUST NOT be modified in any case by Sender, when resending the message. So two or more messages with same Message Identifier (duplicates) MUST have the same value for ExpiryTime. When a message expires on the Sender side before being successfully sent, a Sender RMP MUST NOT send it or resend it, and MUST communicate a delivery failure to the Sender application. The time MUST be expressed as UTC and MUST conform to a [XML Schema] dateTime. The message is considered expired if the current time, in UTC, is greater than the value of the ExpiryTime element. If a receiver receives an expired message, it MUST send the sender a Fault message with Error code of “InvalidExpiryTime”.

NOTES: Given the above definition of ExpiryTime, in case duplicate elimination is required, when a received message is processed, it is sufficient to only check for its duplicates among IDs of past messages that have not expired yet at the time of the duplicate check.

Table5 ExpiryTime Element

Required	Yes
Value	dateTime
Attributes	None
Child elements	None
Fault	InvalidExpiryTime

3.1.4 ReplyPattern Element

The ReplyPattern element is used for a sender to indicate what reply pattern is requested. The ReplyPattern element is a REQUIRED element. This element is used to specify whether the Acknowledgment message (or Fault message) should be sent back directly in the reply to the reliable message, in a separate callback request, or in the response to a separate poll request. This element MUST have one of the following three values:

- **Response** : An Acknowledgment message or Fault message MUST be sent back directly in the response to the Reliable Message. This pattern is not applicable for one-way application level MEP.
- **Callback**: An Acknowledgment message (or Fault message) MUST be sent as a

callback request, using the address in the ReplyTo attribute. This pattern is not applicable for request-response application level MEP.

- **Poll**: An Acknowledgment message (or Fault message) MUST be sent as a response to a poll request. This pattern is not applicable for request-response application level MEP.

The value of this element in MessageHeader of the reply MUST be the same as that of the Request.

The ReplyPattern element contains the following OPTIONAL attribute:

– a **ReplyTo** attribute

Table6 ReplyPattern Element

Required	Yes
Value	String : Response, Callback, or Poll
Attributes	ReplyTo
Child elements	None
Fault	InvalidReplyPattern

(1) ReplyTo attribute

This is an OPTIONAL attribute, used to specify the initial sender's endpoint to receive a callback Acknowledgment message or Fault Message. A value of this attribute MUST be present if the ReplyPattern element value indicates that the Callback reply pattern is requested.

If present, the ReplyTo attribute is required to be URL as defined in [RFC 1738].

3.2 Request Element

The ReliableMessage element is a REQUIRED element. It includes specific information to be used for a reliable message and includes the following attributes and child elements:

- SOAP **mustUnderstand** attribute with a value of "1"
- **AckRequested** element
- **DuplicateElimination** element
- **MessageOrder** element

Table7 Request Element

Required	No * See 3.2 for details
Value	None

Required	No * See 3.2 for details
Attributes	mustUnderstand
Child elements	AckRequested DuplicateElimination MessageOrder
Fault	InvalidRequest

543

544 Example 6 shows an example of Request element.

545 **Example 6 Request Element**

546 (Example to be included later)

547 3.2.1 AckRequested Element

548 The AckRequested element is an OPTIONAL element. It is REQUIRED for guaranteeing
549 message delivery and message ordering. If the MessageOrder element is present, the
550 AckRequested element MUST also be present. This element is used by a sender to request the
551 receiver to send back an Acknowledgment or Fault message for the message sent. If a receiver
552 receives a message with AckRequested element, the receiver is REQUIRED to send an
553 Acknowledgment message even when the message is a duplicate.

554 The pattern used to send the Acknowledgment or Fault message is based on the value of the
555 ReplyPattern element.

556 Table8 AckRequested Element

Required	No
Value	None
Attributes	None
Child elements	None
Fault	InvalidAckRequested

557

558 3.2.2 DuplicateElimination Element

559 The DuplicateElimination element is used to require the receiver node to identify duplicate
560 messages it has received and process them accordingly (see section 2.6). A duplicate message
561 is a message with the same Message Identifier as another message. This element is
562 OPTIONAL. It is REQUIRED when duplicate elimination is mandated. If the MessageOrder
563 element is present, the DuplicateElimination element MUST also be present.

564 Table9 DuplicateElimination Element

Required	No
Value	None

Required	No
Attributes	None
Child elements	None
Fault	InvalidDuplicateElimination

565

566 3.2.3 MessageOrder Element

567 The MessageOrder element is OPTIONAL element. This element is used to request the receiver
568 node to deliver received messages to it's application layer with the same order that the sender
569 has sent. When a sender sends multiple messages with Guaranteed Message Ordering, the
570 sender is REQUIRED to include the MessageOrder element in the message. All messages to be
571 delivered in order MUST have same GroupID and MUST have sequence number as a value of
572 SequenceNumber element in order of the message to be delivered to receiver's application.
573 When the sender requests Guaranteed Message Ordering, the sender MUST use Guaranteed
574 message delivery and duplicate elimination for that message as well. In particular, the sender
575 MUST include both AckRequested element and DuplicateElimination element, as well as the
576 MessageOrder element for Guaranteed Message Ordering.

577 Table10 MessageOrder Element

Required	No
Value	None
Attributes	None
Child elements	None
Fault	InvalidMessageOrder

578

579 3.3 PollRequest Element

580 (To be added later)

581 The PollRequest Element is an OPTIONAL element. This element is used only in the
582 PollRequest message as shown in the Figure5. The PollRequest message contains two direct
583 child elements for SOAP Header element. The one is MessageHeader element, and the other is
584 the PollRequest element. The PollRequest message is used to query Acknowledgment message
585 for specific message. Typically, the PollRequest message is to receive Acknowledgment
586 message for a message sent with Polling RM-Reply Pattern.

587 This element includes the following child element:

588 - a **GroupId** element

589 Table11 PollRequest Element

Required	No
Value	None

Required	No
Attributes	None
Child elements	GroupId
Fault	InvalidPollRequest InvalidGroupId InvalidSequenceNumber

3.3.1GroupId element

This is the same element with GroupId element described in Section3.1.1, except its child element. This element contains the following child element:

- a **SequenceNumberRange** element

3.3.1.1SequenceNumberRange element

The SequenceNumber element is a OPTIONAL element. This element MUST contain the value of the SequenceNumber of the message. (To be added)

Table12 RefToSequenceNumberRange Element

Required	No
Value	unsignedLong
Attributes	None
Child elements	None
Fault	InvalidSequeceNumberRange

3.4 Response Element

The Response element includes response information to be used for an Acknowledgment message or Fault message. It is REQUIRED only when the message includes an Acknowledgment message or a Fault message. This element includes the following attribute and child elements:

- SOAP **mustUnderstand** attribute with a value of “1”

- **RefToGroupId** element

- **RefToSequenceNumberRange** element

This Response element can co-exist with Request element, and it enables to send back Acknowledgment message with the business response to the original message. It also enables the receiver sending an another independent message to the sender with an Acknowledgment message to reduce network traffic.

Table13 Response Element

Required	No
Value	None
Attributes	mustUnderstand
Child elements	RefToGroupId RefToSequenceNumberRange
Fault	InvalidResponse

612

613

614 Example 7 shows an example of the Response element.

615

616 **Example 7 Response Element**

617 (Example will be added later, when the schema is decided)

618 3.3.1 RefToGroupId Element(Needs updating)

619 The RefToGroupId element is a REQUIRED element. This element MUST contain the value of
620 the original GroupId of the message received successfully when used in an Acknowledgment
621 message, or for the message in error, when used in a Fault Message.

622 Table14 RefToGroupId Element

Required	Yes
Value	RFC2396
Attributes	None
Child elements	RefToSequenceNumberRange
Fault	InvalidRefToGroupId

623

624 3.3.1.1 RefToSequenceNumberRange Element(Needs updating)

625 The RefToSequenceNumber element is a REQUIRED element for an Acknowledgment or Fault
626 message when the original message was delivered with Guaranteed Message Ordering. This
627 element MUST contain the value of the original SequenceNumber of the message received
628 successfully when used in an Acknowledgment message, or for the message in error, when used
629 in a Fault Message.

630 Table15 RefToSequenceNumberRange Element

Required	Yes
Value	unsignedLong
Attributes	None

Required	Yes
Child elements	None
Fault	InvalidRefToSequeceNumber Range

631

4 SOAP Fault (Needs updating)

This section describes extensions to the fault codes defined in the SOAP 1.1 specification. Intended to carry error or status information for the SOAP layer, these fault code extensions MUST comply with SOAP Fault as defined in SOAP 1.1. The SOAP Fault is used in this model for notification of only SOAP level errors and Reliable Messaging errors. Errors specific to Reliable Messaging are described in the following sections.

4.1 SOAP Fault Extension for Reliable Messaging

To describe the details of the Reliable Messaging error, an additional Fault element is defined as a child element of SOAP Header.

4.1.1 Fault Element

This element is OPTIONAL and if present MUST appear within a SOAP Header element.

646 **Chart 1 FaultCode Values**

Value of FaultCode	Description
InvalidMessageHeader	Content or format of the Message Header element is invalid, or it was impossible to process the MessageHeader element for some reason.
InvalidMessageIdentifier	Content or format of the Message Identifier is invalid, or it was impossible to process the Message Identifier for some reason.
InvalidRefToGroupId	Content or format of the RefToGroupId element is invalid, or it was impossible to process the RefToGroupId element for some reason.
InvalidRefToSequenceNumber Range	Content or format of the RefToSequenceNumberRange element is invalid, or it was impossible to process the RefToSequenceNumberRange element for some reason.
InvalidTimestamp	Content or format of the Timestamp element is invalid, or it was impossible to process the Timestamp element for some reason.
InvalidExpiryTime	Content or format of the ExpiryTime element is invalid, or it was impossible to process the ExpiryTime element for some reason.
InvalidRequest	Content or format of the Request element is invalid, or it was impossible to process the Request element for some reason.
InvalidAckRequested	Content or format of the AckRequested element is invalid, it was impossible to process the AckRequested element for some reason, or the receiver couldn't send back Acknowledgment message for some reason.
InvalidMessageOrder	Content or format of the MessageOrder element is invalid, or it was impossible to process the MessageOrder element for some reason.

647

Example 8 Fault Message for Reliable Messaging

(Add examples when schema is completed)

4.2 Fault Codes (Update with Poll resolution and Timing resolution. Resolve redundancy with Chart 1)

The following sections describe, in more detail, use of the error codes in Chart 1 .

4.2.1 InvalidMessageHeader

This is an error message to be used when the content or format of the MessageHeader is invalid.

4.2.2 InvalidMessageIdentifier

This is an error message to be used when the content or format of the Message Identifier is invalid.

4.2.3 InvalidRefToGroupId

This is an error message to be used when the content or format of the RefToGroupId element is invalid. This is also for use when no message with a specific Message Identifier, as referred to by the RefToGroupId element, is found.

4.2.4 InvalidRefToSequenceNumber

This is an error message to be used when the content or format of the RefToSequenceNumber element is invalid. This is also for use when no message with a specific Message Identifier, as referred to by the RefToSequenceNumber element, is found.

4.2.5 InvalidTimestamp

This is an error message to be used when the content or format of the Timestamp element is invalid.

4.2.6 InvalidExpiryTime

This is an error message to be used when the content or format of the ExpiryTime element is invalid. This will be used also when a message is expired according to the value of ExpiryTime element.

4.2.7 InvalidRequest

This is an error message to be used when the content or format of the Request element is invalid.

4.2.8 InvalidAckRequested

This is an error message to be used when the content or format of the AckRequested element is invalid.

4.2.9 InvalidMessageOrder

This is an error message to be used when a content or format of MessageOrder element is invalid. This includes an error for wrong SequenceNumber element or its attributes, and the value of the SequenceNumber.

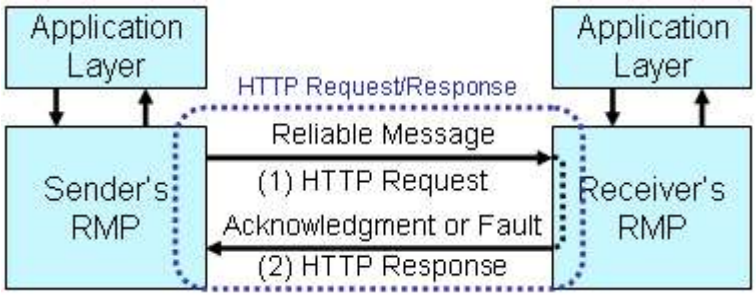
5 HTTP Binding (Needs to include examples)

This section describes the three binding pattern “Reponse”, “Callback”, and “Poll” binding pattern, when the underlying protocol is the HTTP. These binding pattern is identified by the value of ReplyPattern element(See Section3.1.5 for detail). This specification is expecting that the transport layer will not deliver a corrupted message to the reliability layer. When a request message contains AckRequested element, upon receipt of a reliable message, the receiver's RMP MUST send a reply. This reply MUST be either an Acknowledgment or a Fault message. This reply MUST be sent by specified binding pattern in the ReplyPattern element of the request message.

5.1 Reliable Messaging with “Response” binding pattern

The Reliable Messaging Acknowledgment or Fault message MUST be sent back on the same HTTP connection with the HTTP Request that the sender initiated to send the Message. This is illustrated in Figures 7. Both Acknowledgment Message and Fault message MUST be sent back to the sender on the same HTTP connection the sender sent a message.

Figure 7 Response binding pattern



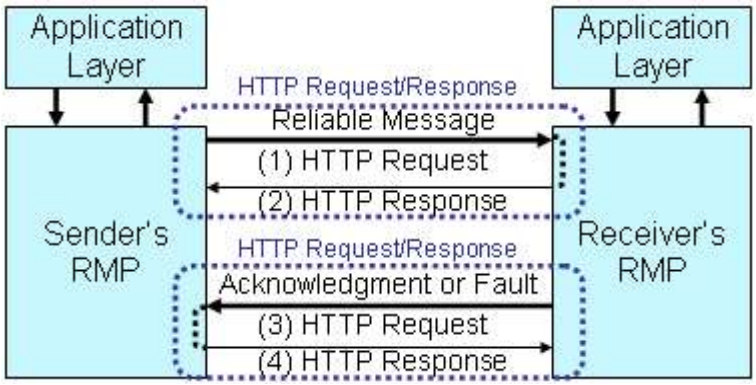
1) The sender initiates an HTTP connection, and sends a Message using the HTTP POST Request. Example 9 is an example of such a message.

2) The receiver sends back an Acknowledgment message to the sender on the same connection, with the HTTP response.

5.2 Reliable Messaging with “Callback” binding pattern

The Reliable Messaging Acknowledgment or Fault message MUST be sent back on a different HTTP connection from the HTTP connection that the sender initiated to send the message. The direction of the HTTP connection that receiver initiates is from the receiver to the sender. This is illustrated in Figure 8.

Figure 8 Callback binding pattern



(1) The sender initiates a HTTP connection, and sends a Message using HTTP POST Request. Example 9 is an example of this message.

(2) The HTTP response to the (1) has no content. Example 10 is an example of this HTTP response.

(3) The Acknowledgment Message is sent with another HTTP connection from the receiver to the sender. Example 11 is an example of this message.

(4) The HTTP response for (3) has no content. Example 10 is an example for this HTTP Response.

Example 9 Request Message with Callback binding pattern

(To be added later after schema is fixed.)

Example 10 HTTP response with no content

(To be added later after schema is fixed.)

Example 11 Response Message with Callback binding pattern

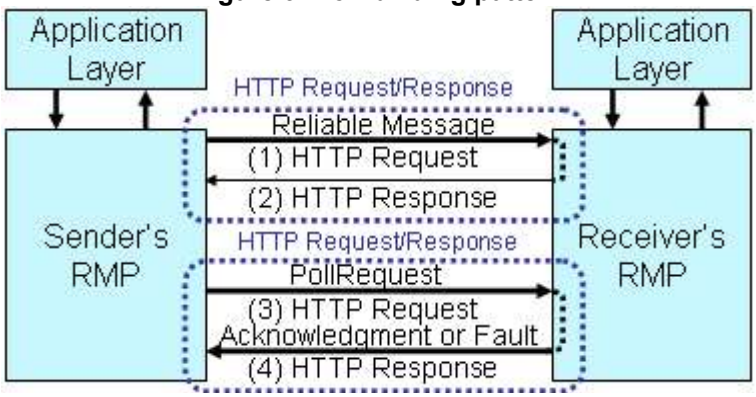
(To be added later after schema is fixed.)

5.3 Reliable Messaging with "Poll" binding pattern

The Reliable Messaging Acknowledgment message MAY also be sent back on a different HTTP connection from the HTTP connection used to send the message being acknowledged. This is illustrated in Figure 8 and 9.

745

746 **Figure 9 Poll binding pattern**



748

749 **Example 12 PollRequest with Poll binding pattern**

750 (To be added later after schema was fixed.)

751

752 **Example 13 Response with Poll binding pattern**

753 (To be added later after schema was fixed.)

754

755

756

757

6 References

6.1 Normative

[RFC1738] "Uniform Resource Locators (URL)", T. Berners-Lee et al, RFC 1738, IESG and IETF, December 1994. Available at

http://www.ietf.org/rfc/rfc1738.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, Bradner, S., IESG and IETF, March 1997. Available at

http://www.ietf.org/rfc/rfc2119.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, Tim Berners-Lee et al, IESG and IETF, August 1998. Available at:

http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, R. Fielding et al, IESG and IETF, June 1999. Available at

http://www.ietf.org/rfc/rfc2616.txt

[RFC2822] "Internet Message Format", RFC 2822, P. Resnick, Editor, IESG and IETF, April 2001. Available at

http://www.ietf.org/rfc/rfc2822.txt

[SOAP 1.1] "Simple Object Access Protocol (SOAP) 1.1", Don Box et al, W3C Note, 8 May, 2000. Available at

http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

"Extensible Markup Language (XML) 1.0, Second Edition", Tim Bray et al, eds., W3C Recommendation, 6 October 2000. Available at

http://www.w3.org/TR/2000/REC-xml-20001006/

[XML Namespaces] "Namespaces in XML", Tim Bray et al., eds., W3C Recommendation, 14 January 1999. Available at

http://www.w3.org/TR/1999/REC-xml-names-19990114/

792

793 [XML Schema] "XML Schema Part 2: Datatypes", Paul V. Biron and Ashok Malhotra, eds. W3C
794 Recommendation, 2 May 2001. Available at

795 http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

796

797 6.2 Non-normative References

798 [ebMS] "Message Service Specification Version 2.0", OASIS ebXML Messaging Services
799 Technical Committee, OASIS Standard, 1 April 2002. Available at

800 http://www.ebxml.org/specs/ebMS2.pdf

801

802 [SOAP 1.2] "SOAP 1.2 Part 1: Messaging Framework", Martin Gudgin, Marc Hadley, Noah
803 Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, eds., W3C Recommendation, 24
804 June 2003. Available at

805 http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

806

807 [WSS] "OASIS Web Services Security TC", documentation in progress, OASIS Technical
808 Committee. Committee home page at

809 http://www.oasis-open.org/committees/wss/

810

811 "XML Schema Part 1: Structures", Henry S. Thompson, David Beech, Murray Maloney, Noah
812 Mendelsohn, eds., W3C Recommendation, 2 May 2001. Available at

813 http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this specification:

David Ingham, Arjuna Technologies Limited

Joseph Chiusano, Booz Allen Hamilton

Peter Furniss, Choreology Ltd

Jeff Turpin, Cyclone Commerce

Pramila Mullan, France Telecom

Jacques Durand, Fujitsu

Kazunori Iwasa, Fujitsu

Tom Rutt (chair), Fujitsu

Jishnu Mukerji, Hewlett-Packard

Robert Freund, Hitachi

Eisaku Nishiyama, Hitachi

Nobuyuki Yamamoto, Hitachi

Ben Bloch, Individual

Mark Hansen, Individual

Paolo Romano, Individual

Dock Allen, Mitre Corporation

Junichi Tatemura, NEC Corporation

Alan Weissberger, NEC Corporation

Magdolna Gerendai, Nokia

Szabolcs Payrits, Nokia

Sunil Kunisetty, Oracle

Jeff Mischkinsky, Oracle

Marc Goodner, SAP

Pete Wenzel, SeeBeyond Technology Corporation

Doug Bunting, Sun Microsystems

Chi-Yuen Ng, University of Hong Kong

Patrick Yee, University of Hong Kong

844 Prasad Yendluri, webMethods, Inc.

845 Scott Werden, WRQ, Inc.

846 In addition, the following people made contributions to this specification:

847 (TBD)

Appendix B. Revision History

[This appendix is optional, but helpful. It should be removed for specifications that are at OASIS Standard level.]

Rev	Date	By Whom	What
WD-0.5	2003-09-04	Kazunori Iwasa	Initial version
WD-0.51		Kazunori Iwasa	Editorial update
WD-0.52		Kazunori Iwasa	Editorial update
WD-0.54	-2003-10-23	Kazunori Iwasa	Issue Rel-38 : Section 3.1.3 Timestamp Issue Rel-98 : Section 3.1.2 and 3.2.3 Issue Rel-40 : Section 3.1.4 Issue Rel-88 : Section 3.1.1 Issue Rel-16 : Section 3.2.1 to 3.2.3 Issue Rel-14 : Appendix C Editorial update
WD-0.60	-2003-10-28	Kazunori Iwasa	Editorial update at F2F in South SF.
WD-0.70	-2003-10-30	Kazunori Iwasa	Section2: Messaging models Section3: Message Format, and others Section4: PollRequest Section5: Binding patterns Editorial update

Rev	Date	By Whom	What
WD-0.83	-2003-11-18	Kazunori Iwasa	Section2.6: Added description of Figure3 Section3: Added tables for each element Rel-31: Section2.5 Rel-38: Timestamp was removed from Section 3 Rel-100: Added Section2.9 Attachments Rel-32: Added definitions to Section1.8 Rel-94: Figure5 and Section 3.3 (Needs additional descriptions and examples in Section3.3) Editorial updates, especially for : http://lists.oasis-open.org/archives/wsrm/200310/msg00054.html All editorial comments above are incorporated except one, which is a comment for line 357, to keep consistency with other sections.
WD-0.84	-2003-12-15	Kazunori Iwasa	Rel-33:Section 1.8: Update on Message Delivery and Acknowledgment Message Rel-50:Section 3.1.3 ExpiryTime Editorial updates

851

852

Appendix C. Futures List

The features and issues in the table below are listed as forward-looking statements regarding possible enhancements or the evolution of this specification.

	Category	***Details***
1	WSDL	Define WSDL extensions profiling the use of RM SOAP extensions.

Appendix D. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.