Web Services Reliable Messaging TC WS-Reliability

Committee Draft 0.992, 17 March 2004

Document identifier:
wd-web services reliable messaging tc-ws-reliability-0.992

Location:
http://www.oasis-open.org/committees/wsrm/documents/specs/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 a Committee Draft for public review.

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/documents/errata/1.1/index.html.

Table of Contents

1 Introduction

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 [SOAP1.2] over HTTP [RFC2616] is not sufficient when an application-level messaging protocol must also address reliability and security. 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].

1.2 Scope and Definition of Reliable Messaging

The focus of this specification is on the SOAP layer and envelope. In the current specification, we will define reliable messaging as the mechanism supporting any of the following requirements:

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

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.

Some messaging features are not mentioned in this specification. They are considered out of scope, yet the design of this specification is preserving compatibility with some of them. They are:

- Application level synchronous messaging. Synchronous messaging applications that require immediate knowledge of the error status instead of waiting for the messaging layer to resend the message when an error is returned.
- Routing features. This specification addresses end-to-end reliability, and is not concerned with intermediaries. The mechanisms described are orthogonal to routing techniques, and can be used in combination with these.

The OASIS WS-RM TC does not attempt to cover all aspects of Reliable Messaging. Several fundamental questions on reliability need to be addressed in subsequent work, and are only partially addressed in this specification:

- Given that some 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., Rate of delivery success, scope of a duplicate check, size of a message archive)? How could these quantitative parameters adjust to resource availability - memory, storage, computing - which depends on the communication system (mobile device, messaging hub, etc.)?

- Beyond the specified qualities of message delivery (Guaranteed Delivery, Duplicate Elimination, and Guaranteed Message Ordering), how much of the synchronization between sender and receiver applications can and should be supported (i.e., the degree to which both sender and receiver parties share the same understanding about the outcome of a reliable exchange)?

1.3 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].

Section 4 includes tables to explain each element. The meaning of labels in the table are follows:

- **Cardinality** : A constraint on the number of instances of an item type which may be present in an enclosing item. (e.g. “Cardinality = 0 or 1” means the message may not include the element, or it may include the element only once.)
- **Value** : A type or format for a value of the element.
- **Attributes** : Attribute names for the element. And type or format for its value is also included in parentheses.
- **Child elements**: Child element for the element.

This specification uses the following namespace prefixes:

Prefix	*Namespace*
soap	http://schemas.xmlsoap.org/soap/envelope/
wsrm	http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1

The choice of any namespace prefix is arbitrary and not semantically significant.

1.4 Relation to Other Specifications

- **W3C SOAP1.1/1.2**: SOAP1.1 [SOAP1.1] and SOAP1.2 [SOAP1.2] are the base protocols for this specification. This specification defines reliable messaging protocol embedded in the SOAP Header.
- **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.
- **OASIS WS-Security**: This specification defines reliability independently from security, each of these features mapping to different SOAP header extensions. Although both features can be used in combination, the specification does not attempt to compose them in a more intricate way, nor does it attempt to profile their combination. This specification can be used with WS-Security [WSS] when that effort is completed in OASIS.

- **WS-I Basic Profile 1.0**: This specification is compliant with WS-I Basic Profile 1.0a [WS-I BP1.0] for use of other technologies including SOAP, WSDL [WSDL1.1], and XML schema [XML Schema].

1.5 Examples of Messages Compliant with WS-Reliability

Example 1 Reliable Message embedded in HTTP Request

```
POST /abc/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.100
SOAPAction: ""
Content-Length: 1214

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/”>
   <soap:Header>
      <Request
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1">
          <MessageId groupId="mid://20040202.103832@oasis-open.org">
            <SequenceNum number="0" status="Start"
             groupExpiryTime="2005-02-02T03:00:33-31:00" />
          </MessageId>
          <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
          <ReplyPattern>Poll</ReplyPattern>
          <AckRequested/>
          <DuplicateElimination/>
          <MessageOrder/>
      </Request>
   </soap:Header>
   <soap:Body>
      <Request xmlns=”http://wsr-example.org/”>Request Message</Request>
   </soap:Body>
</soap:Envelope>
```

The message above uses the Request reliability element, which specifies among other things, that all three features should be used: Guaranteed delivery ("AckRequested" element), No Duplicate Delivery ("DuplicateElimination" element) and Ordered Delivery ("MessageOrder" element).

Example 2 PollRequest Message embedded in HTTP Request

```
POST /abc/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.100
SOAPAction: ""
Content-Length: 1021

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <PollRequest
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1">
          <RefToMessageIds groupId="mid://20040202.103832@oasis-open.org">
            <SequenceNumberRange from="0" to="20"/>
          </RefToMessageIds>
      </PollRequest>
   </soap:Header>
   <soap:Body />
</soap:Envelope>
```

The message above uses the PollRequest reliability element, which is polling the receiver for the status of messages within the range of sequence numbers 0 to 20 of a particular group. The expected response will tell which of these messages have been delivered (Acknowledged).

Example 3 Acknowledgment Message embedded in HTTP Response

```
HTTP/1.0 200 OK
Server: WS-ReliabilityServer
Date: Mon, 02 Feb 2004 10:38:32 GMT
Content-Language: en
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: 924

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Response
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1" replyPattern=”Poll”>
       <NonSequenceReply groupId="mid://20040202.103811@oasis-open.org">
       <SequenceReplies groupId="mid://20040202.103832@oasis-open.org/">
            <ReplyRange from="0" to="14"/>
            <ReplyRange from="16" to="20"/>
       </SequenceReplies>
      </Response>
   </soap:Header>
   <soap:Body />
</soap:Envelope>
```

The message above uses the Response reliability element, which in this case is carrying the response of a previous PollRequest element. The response acknowledges messages for a particular group within the ranges of sequence numbers 0 to 14 and 16 to 20 (meaning that 15 has not been delivered yet, possibly because it was not received.)

Example 4 Fault Message embedded in HTTP Response

```
HTTP/1.0 200 OK
Server: WS-ReliabilityServer
Date: Mon, 02 Feb 2004 10:38:32 GMT
Content-Language: en
Content-Type: text/xml; charset=utf-8
Content-Length: 624

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
     <Response
       xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
       soap:mustUnderstand="1" replyPattern=”Poll” >
         <SequenceReplies groupId="mid://20040202.103832@oasis-open.org">
           <ReplyRange from="15" to="15" fault=”InvalidRequest”/>
         </SequenceReplies>
     </Response>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

The message above uses the Response reliability element, which in this case is carrying the response of a previous PollRequest element. The response is reporting a reliability Fault for message with sequence number 15 within a particular group.

1.6 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. With regard to the transmission of a message from one RMP to another, the former will be act in the role of “sender” and the latter in the role of “receiver”.

Deliver:

An abstract operation the Receiving RMP may invoke per Reliable Message (e.g, a request to the application layer to take responsibility for the Reliable message).

Submit:

An abstract operation the Sending RMP supports, invoked per Reliable message (e.g., a request to the sending RMP to take responsibility for the reliable message. The time at which this operation is invoked must be clearly identifiable so that the RMP can always establish in which order two submissions are made.

Notify:

294 An abstract operation the Sending RMP may invoke per Reliable Message (e.g, a notification that
295 the Sending RMP cannot insure that the Requested Reliability feature were realized).

296

297 **Message Identifier:**

298 A Message Identifier is a value or a combination of values in the message header, that uniquely
299 identifies reliable messages. This identifier is only meaningful to the reliability features described
300 here.

301

302 **Message Delivery:**

303 Message delivery is the action of invoking the deliver operation for a Reliable Message. This
304 action marks the end of the RMP processing for this message. The time at which this action
305 occurs must be clearly identifiable so that the next message processor (application) can always
306 establish in which order two deliveries are made.

307 Examples of message delivery are:

308 · pushing the message in a queue accessible by an application,

309 · calling back an application component,

310 · storing the message in a database where it is accessible by the next processor.

311

312 **Reliable Message:**

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

315

316 **PollRequest Message:**

317 A polling message for Acknowledgment message(s). A sender RMP may send a PollRequest
318 Message for polling of Acknowledgment message(s) regardless of RM-Reply Pattern of the
319 original Reliable Message. E.g., Sender RMP may send PollRequest Message to retrieve
320 Acknowledgment message for a message originally sent with Callback ReplyPattern.

321

322 **Acknowledgment Indication:**

323 An indication which refers to a previous message delivered by the Receiving RMP. An
324 Acknowledgment signals that the acknowledged message has been successfully delivered,
325 meaning that it has satisfied all the reliability requirements placed on it for delivery.

326

327 **Reliable Messaging Fault Indication:**

328 An indication which refers to a previous message which encountered a Reliable Messaging fault
329 condition at the Receiving RMP. It signals to the sender of the referred message that there was a
330 failure to receive or process the message.

331

332

Duplicate Message:

A message is duplicate of another message if it has same message identifier.

Reliable Messaging Reply (RM-Reply):

An indication referring to a previous message, that is either an Acknowledgment Indication or a Reliable Messaging Fault Indication.

Response RM-Reply Pattern:

The Response RM-Reply pattern is used if the outbound Reliable Message is sent in a request of the underlying protocol and the RM-Reply is sent in the response message of the underlying protocol that corresponds to the request.

Callback RM-Reply Pattern:

The Callback RM-Reply pattern is used if the RM-Reply of a previous message is contained in an underlying protocol request of a second request/response exchange (or a second one-way message).

Polling RM-Reply Pattern:

The Polling RM-Reply pattern is used if a second underlying protocol request is issued to the receiver of a previous message, in order to obtain a RM-Reply. The RM-Reply can be either contained in the underlying protocol response to this request or in a separate underlying request from the receiver to the sender. This polling pattern is generally expected to be used in situations where it is inappropriate for the sender of reliable messages to receive underlying protocol requests (behind the firewall cases) or to avoid resending bulk messages often.

1.7 The Reliability agreement

1.7.1 Definition

A Reliability agreement for messaging, or RM Agreement, describes an agreed contract between a sender RMP and a receiver RMP regarding:

- The nature, content and occurrence of exchanged messages.
- The timing, content and occurrence of the submit, deliver, notify operations on these RMPs.

In so far as the submit, notify and deliver operations are interpreted as implementing communication between an RMP and an application, the above contract can be seen as a contract between the application layer, the sender and receiver RMPs.

The way such a contract is established or communicated to each party is out of scope, although the assumption is that only the sender RMP needs to initially have knowledge of the RM Agreement. No prior communication of the contract to the receiving party (RMP and its application) is required. I.e., the Receiver RMP does not need other input than the header of

received messages to get knowledge of the reliability requirements to which these messages are subject.

1.7.2 RM Agreement Items

An RM Agreement is a list of Agreement Items. An RMP implementation MUST be capable of:

(1) taking knowledge of a set of values that represent the RM Agreement Items described in this specification,

E.g., via configuration, or

via an API call, or

via a message, or

via the result of an algorithm.

(2) processing them according to the semantics described in this specification.

Some of these items will appear in the message protocol (I.e., map to some message header field), and some will not.

The following list of Agreement Items is considered by this specification. Each item is listed with its possible values:

- GuaranteedDelivery (enabled/disabled): for setting Guaranteed Delivery. (See Section 3.1 for details)
- NoDuplicateDelivery (enabled/disabled): for setting message delivery without duplicates, or Duplicate Elimination. (See Section 3.2 for details)
- OrderedDelivery (enabled/disabled): for setting Guaranteed Message Ordering. (See Section 3.3 for details)
- GroupMaxIdleDuration (number of seconds): For setting the elapsed time limit from the last message sent or received in a group, after which the group can be terminated. The value MUST NOT be zero or smaller.
- GroupExpiryTime (number of seconds): For setting the date and time after which the group can be terminated. The value MUST NOT be zero or smaller.
- ExpiryTime (number of seconds): For setting the date and time after which a message must not be delivered to the receiving application.
- RetryMaxTimes (integer number): For setting the maximum number of times a message must be resent if not acknowledged. The value MUST be zero or larger.
- RetryTimeInterval (number of seconds): For setting the minimal elapsed time between two re-sending of the same message. The value MUST NOT be zero or smaller.
- ReplyPattern ("Response", "Callback", "Poll") For setting the mode of response for Acknowledgments or Faults.

1.7.3 Messaging Scope of Agreement Items

The messaging scope of these agreement items may vary, as messages may be associated with a group. There are three scopes to consider:

- (s1) All messages sent over a connection between a Sender RMP and a Receiver RMP (default).
- (s2) All messages sent within a group.
- (s3) A single message, standalone (singleton) or within a group of several messages (non-singleton group).

Some agreement items obviously relate to a particular scope, e.g. ExpiryTime is affecting each message separately, while GroupExpiryTime is an agreement item about groups.

The smallest required scope for each RM Agreement item is:

Message scope (s3):

- ExpiryTime
- RetryMaxTimes
- RetryTimeInterval
- ReplyPattern

Group scope (s2):

- GuaranteedDelivery
- NoDuplicateDelivery
- OrderedDelivery
- GroupExpiryTime
- GroupMaxIdleDuration

NOTE: Although a RMP must support each agreement item at the scope level shown, the RMP implementation may also provide a way to assign a broader scope to these items.

Example: a RMP implementation may decide to provide a way to specify the same ExpiryTime value for all messages of a group.

1.7.4 Rules about Agreement Items

When defining an RM Agreement instance, there are some dependencies between the items of the agreement that must be respected:

- If GuaranteedOrdering is enabled for a messaging scope, then GuaranteedDelivery and NoDuplicateDelivery MUST also be enabled for that messaging scope.
- If GroupExpiryTime is enabled for a messaging scope, then the item GroupMaxIdleTime MUST NOT be enabled, and vice versa.

2 Messaging Model

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

2.1 Messaging Context

The Reliable Messaging Model described in this document makes the following assumptions:

- Reliability is a contract between two messaging nodes, with respective roles of sender and receiver: (1) the sender RMP on which the submit message operation is invoked, and (2) the receiver RMP which invokes the deliver message operation. Intermediaries are transparent to this specification. Signal messages resulting from a reliable exchange, such as Acknowledgment message or Reliable Messaging Fault message are sent from the receiving RMP to the sender RMP.
- The underlying protocol is a request-response protocol. In other words, this specification assumes the underlying protocol distinguishes two kinds of messages: requests and responses. Under normal conditions, a response is always sent back for each request. This assumption is not essential to the reliable features described here: these could be reformulated without this assumption.

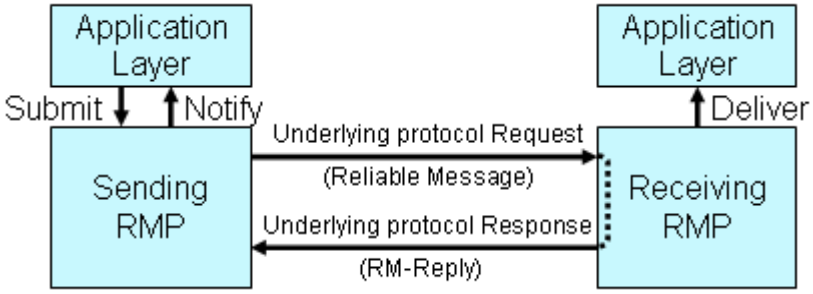
2.2 Message Reply Patterns

There are three ways to send back an Acknowledgment message or a Fault message as described as follows:

(1) Response Message Reply Pattern

With this message reply pattern, the outbound Reliable Message is sent in the underlying protocol request and the RM-Reply is contained in the underlying protocol response message corresponding to the original request. The figure 1 shows this reply pattern.

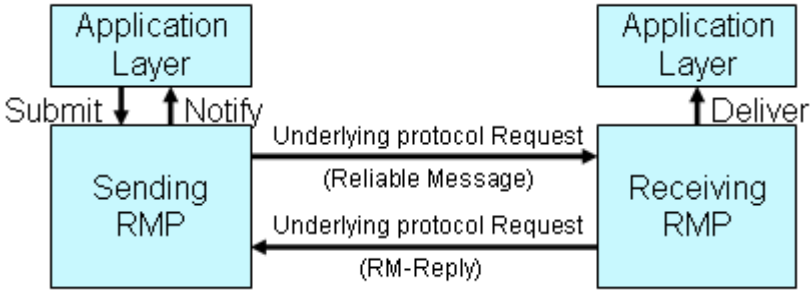
Figure 1 Response Message Reply Pattern



(2) Callback Message Reply Pattern

With this message reply pattern, the RM-Reply 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 reply pattern.

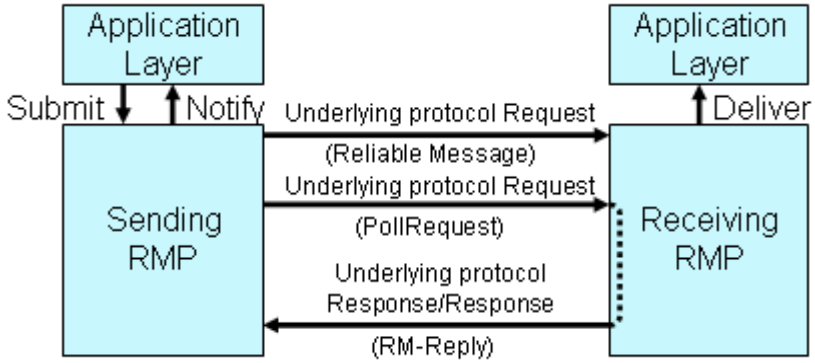
Figure 2 Callback Message Reply Pattern



(3) Poll Message Reply Pattern

With this message reply pattern, 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 RM-Reply is contained in the underlying protocol response to this request. This reply pattern 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 reply pattern.

Figure 3 Poll Message Reply Pattern



2.3 Message Identification and Grouping

Every Reliable Message MUST contain a globally unique Message Identifier. This Message Identifier relies on the notion of group. A message always belongs to a group. A group of messages is sent from the sender RMP to the receiver RMP as a sequence of individual messages. The Message Identifier is a combination of a group ID and of an optional sequence number which is an integer, and which is unique within a group. More precisely, a message is identified as follows:

(1) In case there is only one message in the group (singleton): the group ID, which is a globally unique group identifier, may be used alone as Message Identifier. No sequence number is required, although allowed.

(2) In case the message belongs to a group of several messages: the message is identified by the group ID and a sequence number. The group is submitted to the sender RMP as a sequence of messages, each sequence number value MUST be numbered with consecutive values starting with 0, in the submission order, and MUST be sent in the same order.

3 Reliability Features

3.1 Guaranteed Delivery

When a business payload is submitted to the sender RMP, the GuaranteedDelivery agreement item requires that either: (1) the payload is successfully delivered by the receiver RMP, or (2) the Sender RMP notifies a delivery failure.

The guaranteed delivery mechanism will however do its best to get the message delivered, e.g. resend a message in case of previous failure. In order for the mechanism described here to operate reliably, it is assumed that the underlying transport protocol prevents message corruption.

If the RMP sending a Reliable Message does not receive an Acknowledgment or Fault for a sent message that has not yet expired, it MUST resend the same message with same MessageId to the receiver RMP until either one of the following occurs (whichever occurs first):

- The sender gets a RM-Reply for the message from the receiver.
- The number of resending attempts specified by the RetryMaxTimes agreement item is exhausted.
- The message expires (ExpiryTime is past).

The time interval between two retries is specified by the RetryTimeInterval agreement item. If the sender RMP cannot guarantee that the message has been successfully delivered by the receiving RMP, the sender RMP MUST notify a delivery error.

The sending RMP MUST NOT send retries with a MessageId, for which it received an RM-Reply with one of the following Fault types:

- An Invalid Message Format fault code (Table 16)
- A NonSupportedFeature fault code
- A PermanentProcessingFailure fault code

The RMP MUST NOT return an Reliable Messaging Fault for a delivered MessageId. The RMP MUST NOT deliver a message which encounters an Reliable Messaging Fault.

Guaranteed Delivery assumes also that the RMP functions are operational.

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 neither possible to deliver the message on the receiver side, nor to notify failure on the sender side.

Example 2) . A message persisted in a sending mobile phone may be lost when it's battery is detached. In this case, neither successful message transmission and delivery, nor failure notification will be possible.

3.2 Duplicate Elimination

When an RMP delivers a received business payload, the NoDuplicateDelivery agreement item requires that no future business payload from a message with same identity as the message containing the first payload will ever be delivered.

A number of conditions may result in reception 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 Duplicate Elimination (At-Most-Once) semantics, the receiver RMP MUST NOT deliver a message that is a duplicate of a previously delivered message.

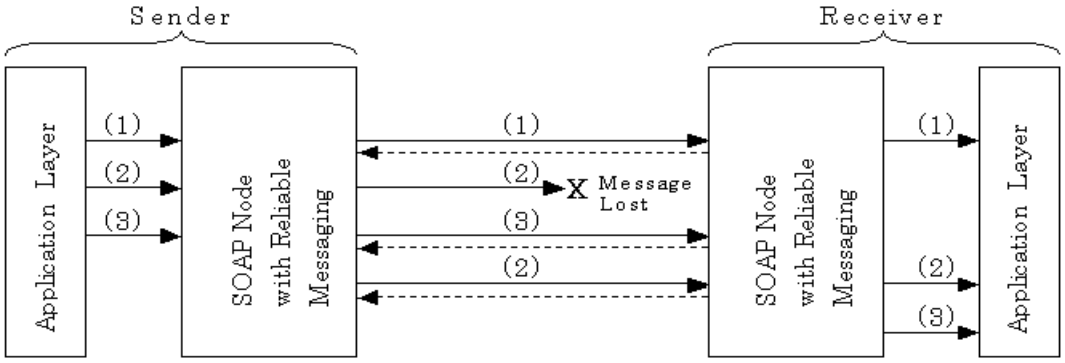
3.3 Guaranteed Message Ordering

When an ordered sequence of business payloads is submitted to a sender RMP, the OrderedDelivery agreement requires that when the receiver RMP delivers one of these business payloads, all previous payloads in the sequence have already been delivered.

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. Guaranteed Message Ordering provides this function. Figure 4 illustrates how Guaranteed Message Ordering works.

In the example illustrated by Figure 4, when the sender application submits three messages (1), (2), and (3) with Guaranteed Message Ordering, the receiver's RMP delivers these messages in the same order. The receiver RMP received message (1) and (3). The receiver RMP delivers the message (1), but it persists message (3) until message (2) is received. When message (2) is received, the RMP delivers message (2) and (3) in order.

Figure 4 Guaranteed Message Ordering



This behavior can be subject to variants and additional rules to deal with specific failure use cases, such as when a node cannot deliver the proper-sequence of messages due to a message being lost or expired.

Failure Case:

In case a message is missing in the sequence and if either one of the two following conditions is verified:

- A previously received and not yet delivered out-of-order message has expired.
- Restoring an ordered delivery would require too much effort from an implementation (e.g. The number of out-of-order received messages is too large for the available storage space).

Then the receiver RMP MUST abort the ordered delivery. i.e., It MUST NOT deliver any message for the group, beyond the last message delivered in order.

3.4 Sequence Number

A sequence number mechanism is used to track and enforce the order of a sequence of messages within the same group. Such a mechanism has been widely used in the past. In the Figure 4 above, messages (1), (2), and (3) will be respectively assigned sequence numbers 1, 2, and 3. If the message (2) was not properly received for any reason, the sender will resend the message. Sequence numbering allows the receiver RMP to easily detect a missing message in a sequence, that is (2), as soon as receiving (3). This condition is recognized by the receiver when the sequence numbers of the messages it receives are not contiguous (e.g., 1, 3, 2).

4 Message Format

4.1 Structure

Figure 5 shows the structure of WS-Reliability elements embedded in the SOAP Envelope.

Figure 5 Structure of WS-Reliability elements

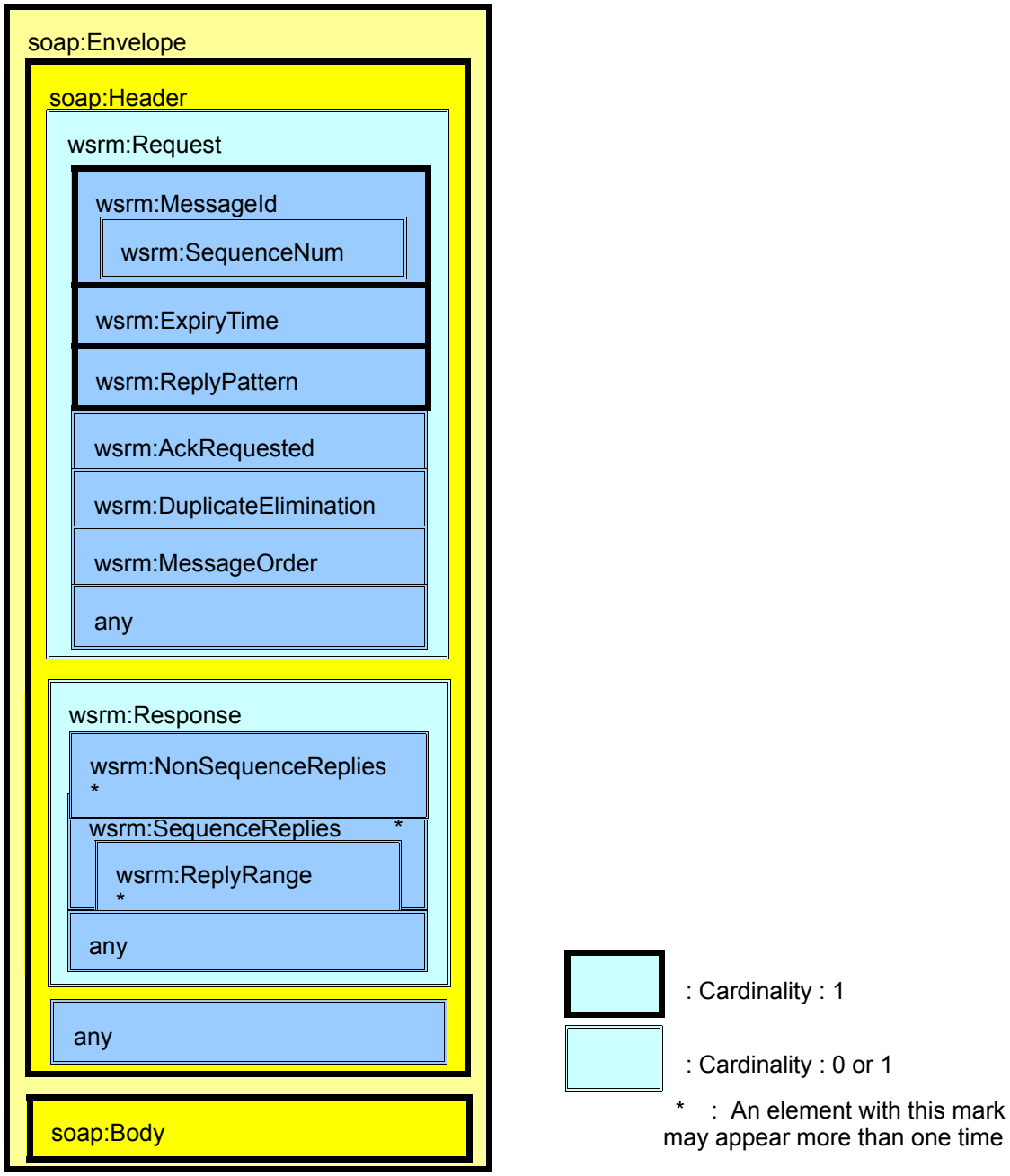
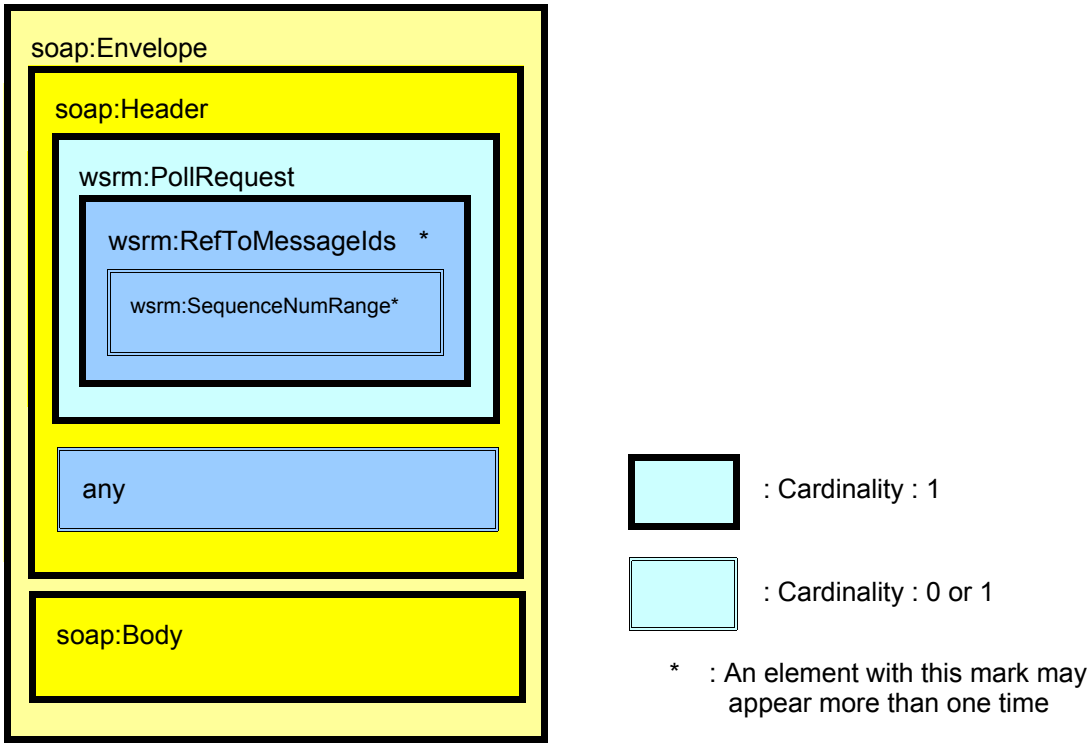


Figure 6 shows the structure of PollRequest message embedded in the SOAP Envelope.

Figure 6 Structure of PollRequest message elements



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

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

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

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

Any of the following three elements can be direct child element of the SOAP Header:

- **Request** element
- **PollRequest** element
- **Response** element

4.2 Request Element

A sending RMP MUST include a Request element in a Reliable Message. The Request element includes specific information to be used for a reliable message. All messages in a group MUST have the same values for the three Reliable Messaging Quality of Service parameters (AckRequested, DuplicateElimination and MessageOrder) in their Request element. This element includes the following attribute and child elements:

- SOAP **mustUnderstand** attribute with a value of “1”
- **MessageId** element
- **ExpiryTime** element
- **ReplyPattern** element
- **AckRequested** element
- **DuplicateElimination** element
- **MessageOrder** element

Table 1 Request Element

Cardinality	1
Value	None
Attributes	MustUnderstand (boolean)
Child elements	MessageId ExpiryTime ReplyPattern AckRequested DuplicateElimination MessageOrder

Example 5 shows an example of a Request element.

Example 5 Request Element

```
<Request
  xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soap:mustUnderstand="1">
 <MessageId groupId="mid://20040202.103832@oasis-open.org/">
          <SequenceNum number="0" status="Start"
            groupExpiryTime="2005-02-02T03:00:33-31:00" />
```

```
 </MessageId>
 <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
 <ReplyPattern>Response</ReplyPattern>
 <AckRequested/>
 <DuplicateElimination/>
 <MessageOrder/>
</Request>
```

4.2.1 MessageId Element

The sending RMP MUST include the MessageId element for a Reliable Message.

This element includes the following attribute:

- a **groupId** attribute

Table 2 MessageId Element

Cardinality	1
Value	None
Attributes	groupId (RFC2396 *See 3.1.1 for details)
Child elements	SequenceNum

(1) groupId attribute

The RMP MUST include this attribute in the MessageId element. This attribute is to identify a sequence of messages, where each sequence is of length 1 or more. The sending RMP MUST use a distinct globally unique groupId for any distinct group of messages. Any group of messages will have a common groupId value. The syntax of this identification is URI, as defined in [RFC2396]. It is RECOMMENDED to use the Message-ID schema, as defined in [RFC2392].

4.2.1.1 SequenceNum Element

The sender MUST include the SequenceNum element for a Group with more than one message.

When a message includes a MessageOrder element, the SequenceNum element is used for guaranteeing the message order within the group of messages specified by the same groupId value. When the MessageOrder element is present, the Message Ordering semantics as described in Section 3.3 applies.

When the sender requests Guaranteed Message Ordering, the sender MUST use Guaranteed Message Delivery and Duplicate Elimination for that message as well. In other words, an AckRequested element and a DuplicateElimination element MUST be present when the MessageOrder element is present.

This element includes the following attributes:

- a **groupExpiryTime** attribute

- a **groupMaxIdleDuration** attribute
- a **number** attribute
- a **status** attribute

In a request message, the sender MAY include either a groupExpiryTime attribute or a groupMaxIdleDuration attribute corresponding to the group termination parameters specified in Section 5.1.2:

If the MessageOrder element appears in the message sent, the receiver of the message MUST make messages available to the application layer only after all messages with the same groupId value and a lower number value have been made available to the application. Example 6 illustrates some message fragments with SequenceNum element:

Example 6 SequenceNum Element

1) First message

```
<MessageId groupId="mid://20040202.103832@oasis-open.org/">
  <SequenceNum number="0" status="Start"
   groupExpiryTime="2005-02-02T03:00:33-31:00" />
</MessageId>
```

2) Second message

```
<MessageId groupId="mid://20040202.103832@oasis-open.org/">
  <SequenceNum number="1" status="Continue"
   groupExpiryTime="2005-02-02T03:00:33-31:00" />
</MessageId>
```

3) Third message

```
<MessageId groupId="mid://20040202.103832@oasis-open.org/">
  <SequenceNum number="2" status="Continue"
   groupExpiryTime="2005-02-02T03:00:33-31:00" />
</MessageId>
```

Table 3 SequenceNum Element

Cardinality	0 or 1 *See 3.1.2 for details
Value	None
Attributes	groupExpiryTime (dateTime) groupMaxIdleDuration (duration) number (unSignedLong) status (string)
Child elements	None

(1) groupExpiryTime attribute

A sender MAY include this attribute when groupMaxIdleDuration attribute is not present. This attribute is used to specify the the date and time at which the sender wishes the sequence group to terminate. The groupExpiryTime MUST be expressed as UTC and MUST conform to a [XML Schema] dateTime.

(2) groupMaxIdleDuration attribute

A sender MAY include this attribute when groupExpiryTime attribute is not present. This attribute is used to specify the maximum idle time. On the receiver side, if the time interval since the last message was received exceeds the groupMaxIdleDuration, then the sequence group may be terminated. On the sender side, the same condition applies to the time since the last message was sent. The groupMaxIdleDuration MUST conform to a [XML Schema] duration.

(3) number attribute

The value of this attribute MUST be unique within the same groupId, and the combination of groupId and SequenceNum MUST be globally unique to be used for Message Identifier.

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

The value of number attribute MUST conform to [XMLSchema] unsignedLong. For the initial message with a specific groupId that is sent to the receiver, the number 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 number 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.

(4) status attribute

This attribute is used to specify status of the group of messages. The first message in a group MUST include this attribute, and the last message in a group MAY include this attribute. 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 message 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.”

When an application is receiving messages from an RMP, the actual order of delivered payloads may be affected by subsequent operations after the "deliver" operation has been invoked. For example, the actual order of delivery to the application may be affected by queuing taking place

between the RMP and the application - and by the way the application reads such a queue - which would be out of scope of this specification.

4.2.2 ExpiryTime Element

The ExpiryTime element is used to indicate the ultimate time after which the receiver RMP MUST NOT invoke the deliver operation for the received message. An RMP MUST include this element in a Request 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 manner by the Sending RMP, when resending the message: two 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.

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 MessageIds of past messages that have not expired yet at the time of the duplicate check.

Table 4 ExpiryTime Element

Cardinality	1
Value	dateTime
Attributes	None
Child elements	None

4.2.3 ReplyPattern Element

The ReplyPattern element is used for a sender to indicate what reply pattern is requested. A RMP MUST include the ReplyPattern element in a Request 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** : A RM-Reply MUST be sent back directly in the response to the Reliable Message. This pattern is not applicable for one-way application level MEP.
- **Callback**: A RM-Reply 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**: A RM-Reply MUST be sent as a response to a poll request. This pattern is not applicable for request-response application level MEP.

The ReplyPattern element contains the following attribute:

- a **replyTo** attribute

Table 5 ReplyPattern Element

Cardinality	1
Value	String : Response, Callback, or Poll
Attributes	replyTo (URI)
Child elements	None

804

805 (1) replyTo attribute

806 A sender MUST include this attribute for a message with “Callback” value for ReplyPattern
807 element. The sender MUST NOT include this attribute for a message with “Response” or “Poll”
808 value for ReplyPattern element. It is to specify the initial sender’s endpoint to receive a callback
809 Acknowledgment message or Fault message.

810 If present, the replyTo attribute MUST be URI as defined in [RFC 2396].

811

812 4.2.4 AckRequested Element

813 A sender MUST include the AckRequested element for Guaranteed Delivery and Guaranteed
814 Message Ordering. This element is used by a sender to request the receiver to send back an
815 Acknowledgment if the message sent was delivered, or else a Fault message. If a receiver
816 receives a message with AckRequested element, the receiver MUST send an Acknowledgment
817 message even when the message is a duplicate, and if it has already been previously delivered.
818 (Refer to “Section 3.1 Guaranteed Delivery” for details)

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

821 Table 6 AckRequested Element

Cardinality	0 or 1
Value	None
Attributes	None
Child elements	None

822

823 4.2.5 DuplicateElimination Element

824 The DuplicateElimination element is used to request the receiver RMP to identify duplicate
825 messages it has received and process them accordingly (Refer to “Section 3.2 Duplicate
826 Elimination” for details).

827 Table 7 DuplicateElimination Element

Cardinality	0 or 1
Value	None

Cardinality	0 or 1
Attributes	None
Child elements	None

4.2.6 MessageOrder Element

This element is used to request the receiver RMP to invoke delivery operation with the same order that the sender has submitted. When a sender submits multiple messages with Guaranteed Message Ordering, the sender MUST include the MessageOrder element in every message. All messages to be delivered in order MUST have the same groupId and MUST have sequence number as a value of SequenceNum element in order of the message to be delivered to receiver's application.

Table 8 MessageOrder Element

Cardinality	0 or 1
Value	None
Attributes	None
Child elements	None

4.3 PollRequest Element

A sender MUST include the PollRequest element only in the PollRequest message as shown in the Figure6. The PollRequest message contains the PollRequest element. The PollRequest message is used to query RM-Reply for specific message. Typically, the PollRequest message is to receive RM-Reply for a message sent with Polling RM-Reply Pattern. However PollRequest message also can be used to receive RM-Reply for a message that was originally sent with Response RM-Reply Pattern or Callback RM-Reply Pattern. The response to a PollRequest message includes RM-Reply information about prior messages. In addition to its use for receiving replies for requests using the poll RM-Reply pattern, a Sending RMP may use it as a general query to determine non-expired messages which have been delivered. If a Receving RMP does not support this general query, it MAY return a notSupportedFeature fault.

RM-Reply MUST be contained in the underlying response of the Poll request if the replyTo attribute doesn't exist and should be sent in an underlying request to the endpoint identified by this attribute if exists.

This element includes the following attributes and child element:

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

Table 9 PollRequest Element

Cardinality	0 or 1
Value	None
Attributes	MustUnderstand (boolean) replyTo (URI)
Child elements	RefToMessageIds

Example 7 PollRequest Element

```
<PollRequest
  xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soap:mustUnderstand="1">
    <RefToMessageIds groupId="mid://20040202.103832@oasis-open.org/">
        <SequenceNumRange from="0" to="5"/>
        <SequenceNumRange from="15" to="20"/>
    </RefToMessageIds>
    <RefToMessageIds groupId="mid://20040202.103811@oasis-open.org/" />
    <RefToMessageIds groupId="mid://20040202.103807@oasis-open.org/">
        <SequenceNumRange from="713" to="6150"/>
    </RefToMessageIds>
</PollRequest>
```

(1) replyTo attribute

This attribute, of type URI, MAY be included by the sending RMP. If present, then the receiver MUST send the RM-Reply in an underlying request to the value of the URI. If not present, the RM-Reply MUST be sent back in the underlying response of the Poll request itself.

4.3.1 RefToMessageIds Element

A sender MUST include the RefToMessageIds element for PollRequest message. This element is to be used to specify RM-Reply to be returned. This element MUST have one groupId attribute and MAY contain zero or more SequenceNumRange element as follows:

- a **groupId** attribute
- zero or more **SequenceNumRange** element

Table 10 RefToMessageIds Element

Cardinality	1 or more
Value	None
Attributes	groupId (URI)
Child elements	SequenceNumRange

When this RefToMessageIds element has a groupId attribute, but doesn't have SequenceNumRange element, the receiver MUST send back all RM-Replies for the messages received in the MessageId. When the RefToMessageIds element has a groupId attribute and SequenceNumRange element(s), the receiver MUST return RM-Reply for messages received that were specified by the combination of groupId of RefToMessageIds and SequenceNumRange element(s). When sender RMP requests multiple RM-Replies with different groupId value in one PollRequest Message, it MUST include RefToMessageIds element for each groupId.

(1) groupId attribute

The RefToMessageIds element MUST include one or more groupId attribute(s). The groupId attribute is to be used to specify the groupId for Acknowledgment message to be returned. The syntax of this attribute is URI, as defined in [RFC2396].

4.3.1.1 SequenceNumRange element

The sender MUST include the SequenceNumRange element when it specifies messages in a group to be acknowledged. If present, attributes of this element MUST contain the value of the SequenceNum of the message. This element MUST contain the following two attributes:

- a **from** attribute
- a **to** attribute

Table 11 SequenceNumRange Element

Cardinality	0 or more
Value	None
Attributes	from (unsignedLong) to (unsignedLong)
Child elements	None

(1) from attribute

A sender MUST include the from attribute in the SequenceNumRange element. This attribute is to be used to specify the smallest SequenceNum of the message range. The value of this attribute MUST be equal or smaller than the value of to attribute. It MUST be the same with the value of the to attribute to specify only one message. The value of this attribute is unsignedLong.

(2) to attribute

A sender MUST include the to attribute in the SequenceNumRange element. This attribute is to be used to specify the largest SequenceNum of the message range. The value of this attribute MUST be equal or larger than the value of from attribute. It MUST be the same with the value of the from attribute to specify only one message. The value of this attribute is unsignedLong.

4.4 Response Element

A receiver MUST include the Response element to indicate Acknowledgment Message for Reliable Messages and indications of Reliable Messaging Fault Messages. This element includes the following attributes:

- SOAP **mustUnderstand** attribute with a value of “1”
- a **ReplyPattern** attribute, which defaults to the value “Response”

Response element MUST include at least one of the following child elements:

- zero or more **NonSequenceReply** element
- zero or more **SequenceReplies** element

When the response is using the callback reply pattern, if the reply and the new request share a common destination URI, a Response element can coexist with a Request element, enabling the combination of an Acknowledgment message with the business response to the original message. This coexistence also enables a receiver sending another independent message to the sender with an Acknowledgment message (e.g., to reduce network traffic).

Table 12 Response Element

Cardinality	0 or 1
Value	None
Attributes	MustUnderstand (boolean) replyPattern (string)
Child elements	NonSequenceReply SequenceReplies

Example 8 shows an example of the Response element.

Example 8 Response Element

```
<Response
  xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soap:mustUnderstand="1" replyPattern=”Callback”>
    <NonSequenceReply groupId="mid://20040202.103832@oasis-open.org" />
    <NonSequenceReply groupId="mid://20040202.103811@oasis-open.org"
      fault=”wsrm:PermanentProcessingFailure”  />
    <SequenceReplies groupId="mid://20040202.103807@oasis-open.org/">
        <ReplyRange from="1" to="4" />
        <ReplyRange from="5" to="5" fault=”wsrm:InvalidRequest” />
```

```
        <ReplyRange from="6" to="42" />
    </SequenceReplies>
</Response>
```

(1) replyPattern attribute

If the response is being returned as a result of a Poll Message Reply Pattern, this attribute must have the value “Poll”.

If the response is being returned using the Callback Reply Pattern, this attribute must have the value “Callback”.

If the response is being returned using the Response Reply Pattern, this attribute indicate the “Response” value. In the case of a response returned using the Response Reply Pattern, the following restrictions apply:

- If the group does not use sequence numbers, the first element of the response must be a NonSequenceReply element containing the groupId which is the globally unique message identifier for the Reliable Messaging Request.
- If the group uses sequence numbering, the first element of the response must be a SequenceReplies element, with its groupId equal to that of the request, and with its first Range element having its from and to attributes both equal to the sequence number in the request.

4.4.1 NonSequenceReply Element

An acknowledgment or an Reliable Messaging Fault indication for a message which does not have a sequence number in its MessageId element MUST include a NonSequenceReply element.

This element MUST contain the value of the groupID attribute for the message the reply pertains to. If the reply is an acknowledgment of delivery, the Receiving RMP MUST NOT include the fault attribute. If the reply is an indication of an Reliable Messaging Fault, the Receiving RMP MUST include the fault attribute, and its value denotes the fault condition which was encountered.

Table 13 NonSequenceReply Element

Cardinality	0 or more
Value	RFC2396
Attributes	groupId (URI) fault (Cardinality 0 or 1)
Child elements	None

(1) groupId attribute

This groupId attribute is to be used to specify the groupId of message (which did not have a sequence number in its MessageId) to be acknowledged, or to have a fault indicated. The syntax of this attribute is URI, as defined in [RFC2396].

4.4.2 SequenceReplies Element

A receiver MUST include the SequenceReplies element to Acknowledgment message or to indicate Reliable Messaging Faults, for messages which include a SequenceNum element in their MessageId element. This element MUST contain the values of the original MessageIds of the messages delivered for a group, and for each Fault Code being reported, the MessageIds of messages which encountered the particular Fault Code.

Table 14 MessageReplies Element

Cardinality	0 or more
Value	RFC2396
Attributes	groupId (URI)
Child elements	ReplyRange

(1) groupId attribute

This groupId attribute is to be used to specify the group of message(s) to be acknowledged, or to have their faults indicated. The syntax of this attribute is URI, as defined in [RFC2396].

4.4.2.1 ReplyRange Element

A receiver MUST include the ReplyRange element in a SequenceReplies element to indicate sequence numbers which either are being acknowledged (in which case receiving RMP MUST NOT include the fault attribute) or have encountered a particular fault condition (in which case the receiving RMP MUST include the fault attribute with that particular RM fault code encountered).

Table 15 ReplyRange Element

Cardinality	None
Value	None
Attributes	from (unsigned Long) to (unsigned Long) fault (QName)
Child elements	None

(1) from attribute

A receiver MUST include the from attribute in the ReplyRange element. This attribute is to be used to specify the smallest SequenceNum of the message range. The value of this attribute MUST be equal or smaller than the value of to attribute. It MUST be the same with the value of the to attribute to specify only one message. The value of this attribute is unsignedLong.

(2) to attribute

A receiver MUST include the to attribute in the ReplyRange element. This attribute is to be used to specify the largest SequenceNum of the message range. The value of this attribute MUST be

1001 equal or larger than the value of from attribute. It MUST be the same with the value of the from
1002 attribute to specify only one message. The value of this attribute is unsignedLong.

1003 (3) fault attribute

1004 This attribute is used to indicate a Reliable Messaging Fault code which was encountered while
1005 processing all of the messages indicated by sequence numbers in the range. The receiving RMP
1006 MUST NOT include this attribute for a ReplyRange element used for Acknowledgments.

1007

1008

4.5 Fault Codes For Reliable Messaging Failures

This section describes the protocol specific fault codes that are needed to better describe the reason for WS-Reliability protocol processing failures.

We categorize the faults into 2 categories based on whether the fault was generated because Reliable Messaging Headers are malformed or invalid due to some runtime processing errors encountered by the RMP. The former category is called Invalid Message Format fault set and the latter is called Request Processing fault set. They are explained in detail in the following sections.

These protocol specific fault codes are returned by the receiving RMP within the response header element. The WS-Reliability protocol does not directly map our Reliable Messaging Faults to the SOAP Fault model.

The SOAP Fault model is used for reporting faults due to the request payload, which fits the SOAP fault model better. Thus a response may have a SOAP Fault message, but the reason for the SOAP fault would be due to problems associated with the WSDL operation message payload. (E.g., A problem with the soap:body of a request message or the inability of the receiving RMP to return the WSDL response in the soap:body of when using the Response RM-Reply pattern).

Example case 1:

For WSDL Request/Response operation types, a SOAP Fault can occur for a reliable request which was delivered, but then encountered an application level Fault due to something wrong in the payload (SOAP Body of request which is not under control of Sending RMP) or application processing space outside the realm of the receiving RMP.

That means a Acknowledgment can be delivered on a SOAP Fault.

Example case 2:

For the Response Reply Pattern, used with WSDL two way operation type, the return message could conceivably carry an indication of an RM Fault, which is not itself carried on a SOAP Fault. The exact behavior in such a case might be an implementation matter.

A message with an RM Fault indication MUST NOT be delivered by the receiving RMP. If the message cannot be delivered due, say an request fault, then there would be no meaningful data for the responder to put into the SOAP Body for the WSDL response.

When using the Response RM-Reply pattern, a WSDL operation reply will not always be available for the receiving RMP to return with the RM-Response. This will occur when there is a Reliable Messaging Fault for the message in the request, or when the message in the request is a duplicate of a prior delivered message with Duplicate Elimination in use.

When a receiving RMP cannot return the WSDL operation response for a request using the Response Reply Pattern, it MUST return the RM Response in a SOAP Fault message. If the RM Fault encountered was due to a problem with the request header element, a SOAP client fault MUST be returned. If the RM Fault encountered was due to a problem with processing by the receiving RMP (including the inability to return a response due to Duplicate Elimination), a soap:server fault must be returned.

The following Fault codes may be carried in a Response element associated with a MessageId.

4.5.1 Invalid Message Format Fault

These faults are thrown by the receiving RMP when the message format of the Reliable Messaging Headers are either invalid or wrong.

1052

1053 **Table 16 Invalid Message Format Fault Code Values**

Local part name	Description and Cause(s)
InvalidRequest	This fault is sent when the Request element is wrong or invalid. Examples are: 1.When any of the mandatory elements such as MessageId, ExpiryTime, ReplyPattern are missing 2.When AckRequested, DuplicateElimination or MessageOrder elements appear twice 3.soap:mustUnderstand attribute is missing
InvalidPollRequest	This fault is sent when the PollRequest element is wrong or invalid. Examples are: 1.When the required RefToMessageId element is missing 2.soap:mustUnderstand attribute is missing
InvalidMessageId	This fault is sent in any of the following cases: 1. If groupId attribute (for MessageId or RefToMessageIds) doesn't exist, or if exists, and the value is wrong or invalid. 2. If number attribute in SequenceNum element doesn't exist, or if exist, the value is invalid or wrong. 3. Attributes (from and to) of SequenceNumRange doesn't exist, or if exists, the values are invalid or wrong.
InvalidMessageParameters	This fault is sent for any of these cases: 1. groupExpiryTime is wrong or invalid 2. groupMaxIdleDuration is wrong or invalid 3. when both group parameters are present 4. when groupExpiryTime decreases for a subsequent messages. in an ordered group 5. If the status attribute of SequenceNum element exist and is not one of allowed {begin|continue| end} value.

InvalidReplyPattern	This fault is sent if the ReplyPattern format is wrong or invalid or when the replyTo attribute is missing for the Callback pattern.
InvalidExpiryTime	This fault is sent if the ExpiryTime format is wrong or invalid.

1054

1055 4.5.2 Message Processing Failure Faults

1056 These faults are thrown by the receiving RMP when there is an error processing a valid Reliable
1057 Messaging message.

1058 **Table 17 Messaging Processing Failure Fault Code Values**

Local part name	Description and Cause(s)
NonSupportedFeature	This fault is thrown by the receiving RMP when it receives a message with a RM feature that it doesn't support. An example is a RM message with MessageOrder element to a receiving RMP that doesn't support Guaranteed Message Ordering
PermanentProcessingFailure	This fault is sent for permanent/fatal processing failures such as: 1. Persistence Storage failures 2. Message Delivery failures A PermanentProcessingFailure fault indicates that the failure is fatal and subsequent retries of the same message will also fail.
MessageProcessingFailure	This fault is sent for transient failures such as: 1. Maximum number of buffered requests exceeded the limit. 2. Maximum number of threads reached the limit etc. A transient fault unlike a permanent fault is a temporary one and MAY succeed in subsequent retries.

1059

Example 9 Fault Message for Reliable Messaging

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Response
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1" replyPattern=”Callback”>
        <SequenceReplies groupId="mid://20040202.103832@oasis-open.org">
            <ReplyRange from="1" to="1" fault=”InvalidRequest” />
        </SequenceReplies>
      </Response>
   </soap:Header>
   <soap:Body />
</soap:Envelope>
```

5 Operational Aspects and Semantics

5.1 Message Group Life Cycle

5.1.1 Group Termination

Being able to know when a group may be terminated, is essential for efficient management of the persistent store of an RMP. As groups may last a long time and their state requires persistence, it is important to know when their persistent image can be reclaimed. The termination cases described in this section may seem multiple and complex. This plurality results from the flexibility given to users in specifying various ways a group can be terminated, which in turn depends on application needs. However, in spite of this plurality, the termination logic is straightforward to implement and shares the same basic mechanisms across termination cases.

Termination of a group in the sender RMP and in the receiver RMP are two distinct events not synchronized by any special message, but instead occurring as the result of rules applying separately to the sender and to the receiver. As a consequence, the termination of a group may occur at quite different times on the sender and receiver RMPs. However, this lack of synchronization allowed by these termination rules is not consequential.

More precisely, there are two distinct steps that an RMP must perform when terminating a group, and these may also occur at different times, especially on the receiver side:

- **Group Closing:** When a group is closed in the Sender RMP, no new message is expected to be sent by the RMP for this group. When a group closes in the receiver RMP, no new message is expected to be received for this group anymore. After a group is closed, all subsequent messages sent with same group ID would be handled as belonging to a new group, unless they are duplicates of previous messages in the group, in which case they are treated as duplicates within this group. If a message is received after the closing of a group, with same group ID as the closed group, it may be considered by the Receiver as belonging to a new group (the Receiver is not required to verify that a new group ID value has not already been used in a previous group, as this test is impractical).

- **Group State Removal:** The state of a group includes group-specific attributes such as group status (e.g. “active”, “closed”), group ID, current sequence number, as well as all received Message Identifiers for the group (e.g. sequence number intervals). The state of a group also includes the persistent image of messages of this group being currently processed, although the removal of the persistence of messages follows its own rules. E.g. The resending mechanism for guaranteed delivery will take care of removal of messages on the sender side, once they are acknowledged. State removal occurs at the time or after the group is closed. When the state of a group is removed, all group attributes are removed, including the past message Ids on receiver side. Therefore not duplicate check may be done over past messages of this group.

In all termination cases (t1, t2, t3, t4, t5) described in this section, it is not necessary to remember the ExpiryTime of all messages of a group, as only the max(ExpiryTime) of messages received for the group is needed. These termination rules apply to both ordered and unordered groups. However, these rules do NOT apply to singleton groups, which contain a single message with no sequence number.

Assuming the last message of a group is marked with “end” status value, a group is defined as being “complete” on the receiver RMP when all the messages from ‘start’ to ‘end’ are received.

5.1.2 Group Termination Parameters

There are two RM Agreement items - GroupExpiryTime and GroupMaxIdleDuration that can be used to determine when a group can be terminated. These two items can be considered as controlling the persistence of group data.
To each of these agreement items, correspond respectively the message header attributes: groupExpiryTime and groupMaxIdleDuration. The following requirements pertain to these header attributes:

a) the First message in a group (the one with status=start) MUST be used by the sender to indicate that time based group persistence control is in use for the group.

- If the first message in the sequence of a group has neither group persistence parameter present, the group will be terminated according to condition t4 or t5.
- If the first message has either one of the two group persistence parameter present (either groupExpiryTime, or groupMaxIdleDuration) then the group will be subject to termination rules t1, t2 or t3 described below.
- A fault MUST be returned if both group persistence parameters are present in any request message. An InvalidGroupParameter fault shall be sent in this case..
- If groupExpiryTime is in use, the sender must not send a message in that group with an ExpiryTime greater than the groupExpiryTime.

b) The group termination parameter which was sent on the first message in the group MUST be used on all subsequent messages in that group, and MUST be assigned a value.

c) The receiver MUST use the value from the message with the highest sequence number received for the group.

d) In any subsequent message the parameter which was sent in the first message can be changed by sending a new value. A new value for groupMaxIdleDuration can either be increased or decreased. The protocol allows change (up or down) of groupExpiryTime, as long as it is never less than max(ExpiryTime) of messages received so far for the group.

An InvalidMessageParameters Fault MUST be returned if the value of groupExpiryTime is decreased to be less than the max(ExpiryTime) of messages received for the group.

The receiving RMP MUST update its Group Termination Criteria using parameter values from a Reliable Message, even if that request encounters a Reliable Messaging Fault.

5.1.3 Termination Rules

(1) Termination (t1):

Context:

The group had groupExpiryTime specified.

Receiver side:

Triggering event: groupExpiryTime is over.

The RMP MUST NOT accept any new message for this group and MUST close the group. It is RECOMMENDED that its state be removed as soon as possible after this. No duplicate check needs to be done against that group ever. If a "late duplicate" arrives, it would never be delivered to the next layer, as its ExpiryTime, which is always earlier than groupExpiryTime, would have expired.

Sender side:

Triggering event: groupExpiryTime is over.

The group MUST be closed, and its state removed from the RMP.

(2) Termination (t2):

Context:

The group had groupMaxIdleDuration specified.

Receiver side:

Triggering event: groupMaxIdleDuration is over.

The group MUST be closed. But unlike (t1), some of its past messages may not have expired yet, and therefore their ID still be needed for duplicate checks. If we define max(ExpiryTime) as the max of all ExpiryTimes of messages received for a group, an RMP MUST persist the state of a group even after closing of the group, at least until max(ExpiryTime) is reached, in case Duplicate Elimination is required.

Sender side:

Triggering event: groupMaxIdleDuration is over.

The group MUST be closed, and its state removed from the RMP when the time elapsed since the last sent message (including retries) exceeds groupMaxIdleDuration.

(3) Termination (t3):

Subcase t3.1: The group is complete on receiver side.

Context:

The group had either groupExpiryTime or groupMaxIdleDuration specified. All received messages for the group have been delivered (no missing message).

Receiver side:

Triggering event: The RMP receives a status="end" message.

The group MUST be closed. However, its state is removed according to (t1) or (t2), depending which termination criterion was specified for the group.

Sender side:

Triggering event: The RMP sends a status="end" message.

All messages of the group have been sent. If Guaranteed Delivery was required, the group MUST be closed and state is removed once all sent messages have either been acknowledged, or their delivery failure notified. If no Guaranteed Delivery was required, the group MUST be closed and its state may be removed immediately.

Subcase t3.2: The group is not complete on receiver side.

Context:

The group had either groupExpiryTime or groupMaxIdleDuration specified. Not all received messages for the group have been delivered (some message is missing).

Receiver Side:

Triggering event: the RMP receives a status="end" message.

In this case, the group is not yet closed. Indeed, an "end" status only tells that "no greater sequence number will ever be received after", but late messages may still arrive for this group. Then the Receiver RMP MUST apply termination rules of (t1) or (t2), depending which one of the two group termination parameters (I.e. groupExpiryTime or groupMaxIdleDuration) was specified.

Sender Side:

Triggering event: the RMP sends a status="end" message.

As all messages for the group have been sent, same rules apply as in t3.1.

(4) Termination (t4):

Subcase t4.1: The group was complete on receiver side.

Context:

The group had neither groupExpiryTime nor groupMaxIdleDuration specified. All received messages for the group have been delivered (no missing message).

Receiver side:

Triggering event: the RMP receives a status="end" message.

The group MUST be closed. The time of removal of its state is determined by the max (ExpiryTime) received of the Group.

Sender side:

Triggering event: the RMP sends a status="end" message.

Same rule applies as in t3.1.

Subcase t4.2: The group was not complete on receiver side.

Context:

The group had neither groupExpiryTime nor groupMaxIdleDuration specified. Not all received messages for the group have been delivered (some message is missing).

Receiver side:

Triggering event: The RMP receives a status=”end” message.

In this subcase, the RMP should keep the group processing active: this event, by itself, does not cause the closing of the group.

Sender side:

Triggering event: the RMP sends a status="end" message.

Same rule applies as in t3.1.

(5) Termination (t5):

Context:

The group is under Guaranteed Message Ordering reliability requirement. Not all received messages for the group have been delivered (some message is missing).

Receiving side:

Triggering event: In an ordered group, a message expires before delivery in out-of-order sequence.

The group MUST be closed.

State is removed according to the following rule:

- If the group does not have termination parameter, then it will be removed when the max(ExpiryTime) received of the group passes.
- If the group uses groupExpiryTime, then removal criteria as defined in t1 will apply.
- If the group uses groupMaxIdleDuration, then removal criteria as defined in t2 will apply.

Sender Side:

Triggering event: In an ordered group, a non-acknowledged message expires.

State is removed according to the following rule:

- If the group does not have termination parameter, then it will be removed when the max(ExpiryTime) sent for the group passes.
- If the group uses groupExpiryTime, then removal criteria as defined in t1 will apply.
- If the group uses groupMaxIdleDuration, then removal criteria as defined in t2 will apply.

5.2 WSDL Operation Type

This specification supports Reliable Messaging capabilities for WSDL 1.1 [WSDL 1.1] One-way and Request-response operation types only. While a Request-reponse operation can use any of the three RM-Reply patterns to receive acknowledgments or faults, an One-way operation can only use either Callback or Poll RM-Reply pattern. See the table below for a complete support matrix:

Table 18 WSDL operation types

WSDL operation type	***Response RM-Reply pattern***	***Callback RM-Reply pattern***	***Poll RM-Reply pattern***
Request/Response WSDL operation type*	Supported	Supported	Supported
One-way WSDL operation type	Disallowed **	Supported	Supported

* The current version of the WS-Reliability protocol does not support reliability of WSDL response messages (the "output" messages in WSDL operations). It only supports reliability of the WSDL request ("input" messages).

** WS-I BP 1.0 disallows sending a SOAP envelope in HTTP response, so an RMP is not required to support this. However, this specification does not require an RMP to enforce this restriction (i.e. WS-I BP compliance). The receiver can do whatever the header asks for.

While the specification doesn't prohibit using Callback or Poll RM-Reply patterns to receive acknowledgments or faults for a Request-response operation, it is encouraged to use Response RM-Reply pattern for such operations as the acknowledgment or the fault can be sent on the same response itself thus saving extra round trips.

5.3 Poll Reply Pattern Semantics and Usage

Guaranteed Delivery will be most commonly used for a one-way message as the Sender know the status of the message delivery otherwise.

So the most common use case is to use AckRequested with Callback RM-Reply pattern so that the Sender can receive the acknowledgment or a fault on a listener at its end. However this pattern doesn't help when the Sender is within a firewall, as one cannot receive requests without opening a firewall, thus causing security lapses.

An alternate solution is for the Sender to ask the Receiver for the receiving status of the message it has sent earlier on a different channel. Such a pattern is called the poll RM-Reply pattern. The Sender sends a Poll request for a message it is to inquire and the Receiver sends a Poll response with the acknowledgment. The Sender can also batch multiple Poll requests for an efficient use. Receiver in such case will send acknowledgments for those messages it has received. If a Poll request is partially or completely invalid or wrong, then the Receiver sends either a InvalidPollRequest or InvalidMessageId Fault back.

Also, a RM Poll response MUST NOT be piggybacked on a different RM Poll request.

5.4 Attachments

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

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

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

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

4) The receiver RMP MUST deliver all MIME parts in a Reliable Message to the receiving application.

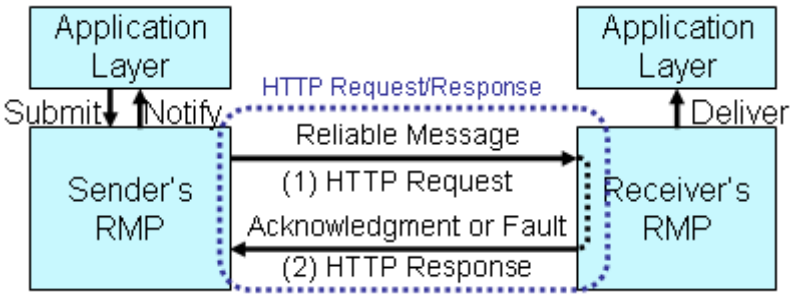
6 HTTP Binding

This section describes the three binding pattern - “Response”, “Callback”, and “Poll” binding pattern for HTTP. These binding pattern is identified by the value of ReplyPattern element (See Section4.2.3 for detail). This specification expects 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 message or a Fault message. This reply MUST be sent by specified binding pattern in the ReplyPattern element of the request message.

6.1 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 Figure 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 10 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.

Example 10 Request Message with Response binding pattern

```
POST /abc/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.100
SOAPAction: ""
Content-Length: 1214

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
    <Request
      xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
```

```
        soap:mustUnderstand="1">
          <MessageId groupId="mid://20040202.103832@oasis-open.org/">
            <SequenceNum number="0" status="Start"
             groupExpiryTime="2005-02-02T03:00:33-31:00" />
          </MessageId>
          <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
          <ReplyPattern replyTo="http://www.oasis-open.org/">Response</ReplyPattern>
          <AckRequested/>
          <DuplicateElimination/>
          <MessageOrder/>
      </Request>
  </soap:Header>
  <soap:Body>
      <Request xmlns=”http://wsr-example.org/”>Request Message</Request>
  </soap:Body>
</soap:Envelope>
```

Example 11 Acknowledgment Message with Response binding pattern

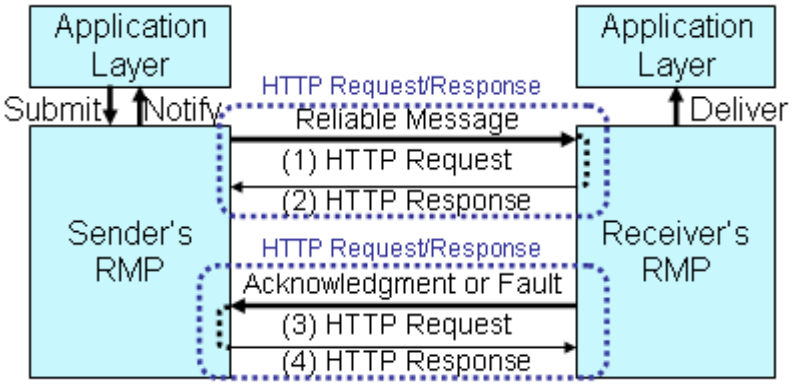
```
HTTP/1.0 200 OK
Server: WS-ReliabilityServer
Date: Mon, 02 Feb 2004 10:38:32 GMT
Content-Language: en
Content-Type: text/xml; charset=utf-8
Content-Length: 924

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
      <Response
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1" replyPattern=”Response”>
       <SequenceReplies groupId="mid://20040202.103832@oasis-open.org/">
            <ReplyRange from="0" to="0"/>
       </SequenceReplies>
      </Response>
  </soap:Header>
```

```
  <soap:Body />
</soap:Envelope>
```

6.2 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 12 is an example of this message.

(2) The HTTP response to the (1) has no content. Example 13 is an example of this HTTP response.

(3) The Acknowledgment Message is sent with another HTTP connection from the receiver to the sender. Example 14 is an example of this message.

(4) The HTTP response for (3) has no content. Example 13 is an example for this HTTP Response.

Example 12 Request Message with Callback binding pattern

```
POST /abc/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.100
SOAPAction: ""
Content-Length: 1214

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
```

```
      <Request
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1">
          <MessageId groupId="mid://20040202.103832@oasis-open.org/">
            <SequenceNum number="0" status="Start"
             groupExpiryTime="2005-02-02T03:00:33-31:00" />
          </MessageId>
          <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
          <ReplyPattern replyTo="http://www.oasis-open.org/">Callback</ReplyPattern>
          <AckRequested/>
          <DuplicateElimination/>
          <MessageOrder/>
      </Request>
   </soap:Header>
   <soap:Body>
      <Request xmlns=”http://wsr-example.org/”>Request Message</Request>
   </soap:Body>
</soap:Envelope>
```

Example 13 HTTP response with no content

```
HTTP/1.0 200 OK
Server: WS-ReliabilityServer
Date: Mon, 02 Feb 2004 10:38:32 GMT
Content-Language: en
Content-Type: text/xml; charset=utf-8
Content-Length: 184
```

Example 14 Acknowledgment Message with Callback binding pattern

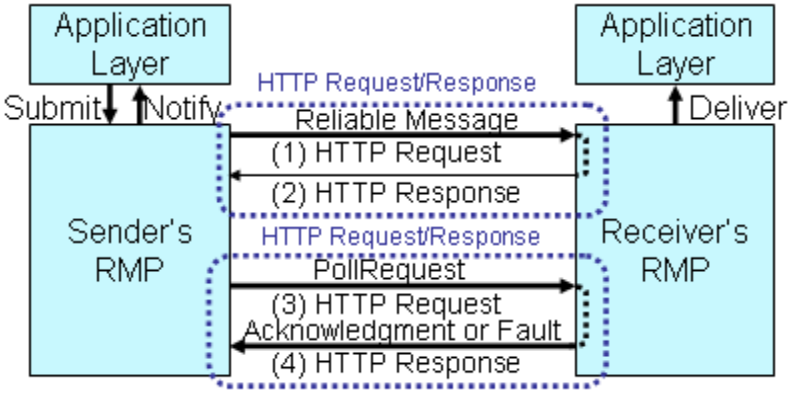
```
POST /xyz/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.200
SOAPAction: ""
Content-Length: 1024

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
```

```
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <Response
        xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
        soap:mustUnderstand="1" replyPattern=”Callback”>
       <SequenceReplies groupId="mid://20040202.103832@oasis-open.org/">
            <ReplyRange from="0" to="0"/>
       </SequenceReplies >
      </Response>
   </soap:Header>
   <soap:Body />
</soap:Envelope>
```

6.3 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 9.

Figure 9 Poll binding pattern



(1) The sender initiates a HTTP connection, and sends a Message using HTTP POST Request.

(2) The HTTP response to the (1) has no content. Example 13 is an example of this HTTP response.

(3) The sender initiates a HTTP connection, and sends a PollRequest message with HTTP POST Request. Example 15 is an example of this message.

(4) The HTTP response for (3) includes Acknowledgment message and/or Reliable Messaging Fault. Example 16 is an example for this message.

Example 15 PollRequest message with Poll binding pattern

```
POST /abc/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.100
SOAPAction: ""
Content-Length: 1021

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
    <PollRequest
      xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
      soap:mustUnderstand="1">
        <RefToMessageIds groupId="mid://20040202.103832@oasis-open.org/">
          <SequenceNumberRange from="0" to="20"/>
        </RefToMessageIds>
    </PollRequest>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

Example 16 Acknowledgment message with Poll binding pattern

```
HTTP/1.0 200 OK
Server: WS-ReliabilityServer
Date: Mon, 02 Feb 2004 10:38:32 GMT
Content-Language: en
Content-Type: text/xml; charset=utf-8
Content-Length: 924

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
    <Response
      xmlns="http://www.oasis-open.org/committees/wsrm/schema/1.1/SOAP1.1"
      soap:mustUnderstand="1" replyPattern=”Poll” >
```

```
        <SequenceReplies groupId="mid://20040202.103832@oasis-open.org/">
            <ReplyRange from="0" to="14"/>
            <ReplyRange from="16" to="20"/>
        </SequenceReplies>
      </Response>
   </soap:Header>
   <soap:Body />
</soap:Envelope>
```

7 Conformance

In order to be conform to this specification, an implementation must satisfy all the following conditions:

- It complies with the following interpretation of the keywords OPTIONAL and MAY: When these keywords apply to the behavior of the implementation, the implementation is free to support these behaviors or not, as stated in [RFC2119].
- If it has implemented optional features and/or behavior defined in this specification, it MUST be capable of interoperating with another implementation that has not implemented the optional syntax, features and/or behavior. It MUST be capable of processing the prescribed failure mechanism for those optional features it has chosen to implement.
- If it has chosen NOT to implement optional features, it is capable of interoperating with another implementation that has chosen to implement these. It MUST be capable of generating the prescribed failure mechanism for those optional features it has chosen to NOT implement.

8 References

8.1 Normative References

[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

[RFC2392] “Content-ID and Message-ID Uniform Resource Locators”, RFC2392, E. Levinson, IESG and IETF, August 1998. Available at

http://www.ietf.org/rfc/rfc2392.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 1.2] "SOAP 1.2 Part 1: Messaging Framework", Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, eds., W3C Recommendation, 24 June 2003. Available at

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

[XML] "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/

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

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

[WS-I BP1.0] “Basic Profile Version 1.0a”, Keith Ballinger, David Ehnebuske, Martin Gudgin, Mark Nottingham, Prasad Yendluri, eds., WS-I specification, 8 August 2003. Available at

http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html

8.2 Non-normative References

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

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

[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/

[WSDL1.1] “Web Services Description Language (WSDL) 1.1”, Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, eds., W3C Note, 15 March 2001. Available at

http://www.w3.org/TR/2001/NOTE-wsdl-20010315

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

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

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

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

[SOAP with Attachments] "SOAP Messages with Attachments”, John J. Barton, Satish Thatte, Henrik Frystyk Nielsen, W3C Note, 11 December 2000, Available at

1591 http://www.w3.org/TR/SOAP-attachments

1592

Appendix A. WS-Reliability Features, Properties and Compositor (Normative and Optional)

I. Introduction

Users of a Web Service will need to be aware of the reliability capabilities (RM capabilities)that are supported/required by the service. One practical location to advertise these capabilities is in the service description (WSDL document), which allows for publishing both abstract service definitions as well as concrete protocol details (bindings). This allows clients (or other Web services) to easily obtain information about specific capabilities such as guaranteed delivery, duplicate elimination, message ordering, and various reply patterns of a specific Web service, before calling the service. While bundling reliability capabilities with the service description may not be desirable in all cases, it is expected that this convenient approach will often be appropriate. The WSDL annotation mechanism described here is a flexible way to add such capability assertions.

WS-Reliability uses the WSDL 1.1 extensibility points to define an extensible framework consisting of features, properties and compositors to address the needs of a reliable Web service to advertise its capabilities, and composibility of those capabilities.

The following extensibility elements relevant to RM capabilities are used:

- feature - abstract RM capability or assertion associated with WSDL elements.
- property - an assertion or constraint on an atomic RM capability and its value(s) associated with WSDL elements.
- compositor - specify how features and properties are combined.

An annotation composed with the above extensibility elements will specify the reliability features and properties associated with specific WSDL constructs. Features and properties represent reliability capabilities and compositors specify how these capabilities are composed.

This would allow, for example, a Web service description to advertise the fact that clients invoking the service must use duplicate elimination or message ordering.

I.A Notational Convention

This specification uses the following namespace prefixes:

Prefix	*Namespace*
xs	http://www.w3.org/2001/XMLSchema
wsdl11	http://schemas.xmlsoap.org/wsdl/
fnp	http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/
wsrm	http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/

The choice of any namespace prefix is arbitrary and not semantically significant.

I.B Conformance

Implementations of WS-Reliability are expected, though not required, to understand the WSDL extensibility points defined in this section.

Understanding of these extensibility points promotes interoperability. When a WSDL document contains these extensibility points, it is through these extensibility points that a service advertises its supported and required features. Therefore it is RECOMMENDED that implementations recognize, understand and support these extensibility points.

It is also possible for services to advertise features through other channels (such as UDDI) in addition to these extensibility point.

II. WSDL Extensibility Elements

II.A Compositor

The compositor semantics describe how features and properties are composed for the enclosing component (or WSDL 1.1 element). The compositor's semantics determine whether the usage of composed elements by a client to the service, is required or optional. The RM capabilities represented by these elements must all be supported by the Service. A compositor element can occur as a child element of wsdl11:portType, wsdl11:operation (which may itself be a child of wsdl11:portType or wsdl11:binding), wsdl11:binding, wsdl11:service and wsdl11:port. The compositor element utilizes the extensibility defined by WSDL 1.1. A compositor element specifies the semantics for combining its children elements. These children elements can be additional compositor, features, properties, or extensibility element(s).

A compositor element is expressed by the following pseudo-syntax:

```
<fnp:compositor uri="..." name="NCName"?>
  [fnp:feature/> | <fnp:property/> | <fnp:compositor/> |
   <extensibility-element/>]+
</fnp:compositor>
```

The uri attribute of the compositor specifies its semantics. Four different compositors (URIs) and their capability-related semantics are described below. It is possible to provide additional compositors by using other URIs. The ability to define additional compositors and the existence of extensibility points (represented by "<extensibility-element>") make the framework extensible. The optional name attribute identifies the compositor. An element built with such compositors represents an RM capability.

- **all:** this compositor specifies that a service invocation MUST comply with all the children elements (representing RM capability assertions). This compositor is identified by using the URI:

  "http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositors/all"

- **choice:** this compositor specifies that a service invocation MUST comply with exactly one of the possibly many children elements (representing RM capability assertions). This compositor is identified by using the URI:

  "http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositors/choice"

- **one-or-more:** this compositor specifies that a service invocation MUST comply with at least one of the possibly many children elements (representing RM capability assertions). This compositor is identified by using the URI:

  "http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositors/one-or-more"

- **zero-or-more:** this compositor specifies that a service invocation MAY comply with one or more of the children elements (representing RM capability assertions). This compositor is identified by using the URI:

  "http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositors/zero-or-more"

Examples for each compositor are provided in Section VII below.

Compositors specified at different WSDL components are implicitly aggregated using the 'all' compositor at the dependent WSDL component. Consider the example below,

```
<wsdl11:definitions>
 ...
 <wsdl11:portType name="myPortType">
  <fnp:compositor uri="..." name="A">
    ...
  </fnp:compositor>
  ...
 </wsdl11:portType>

 <wsdl11:binding name="myBinding" type="myPortType">
  <fnp:compositor uri="..." name="B">
    ...
  </fnp:compositor>
  ...
 <wsdl11:binding>
 <wsdl11:service name="myService">
  <wsdl11:port name="myPort" binding="myBinding>
    ...
  </wsdl11:port>
 </wsdl11:service>
<wsdl11:definitions>
```

Compositor specified at the wsdl11:portType "myPortType" and compositor specified at wsdl11:binding "myBinding" are aggregated at the dependent wsdl11:port "myPort" using the 'all' compositor. I.e., the equivalent compositor at "myPort" is,

```
<fnp:compositor
uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
 <fnp:compositor uri="..." name="A">
```

```
  </fnp:compositor>
  <fnp:compositor uri="..." name="B">
    ...
  </fnp:compositor>
</fnp:compositor>
```

II.B Feature

A feature describes an abstract RM capability or assertion associated with a WSDL element. A feature can occur only as a child of a compositor.

Whether the usage of a feature is required or not is defined by the enclosing compositor(s). A feature is identified by a URI. Recognizing the URI of a feature is considered to be equivalent to understanding the feature identified by that URI.

A feature element is expressed by the following pseudo-syntax:

```
<fnp:feature uri="...">
  [<fnp:compositor/> | <extensibility-element/>]*
</fnp:feature>
```

II.C Property

A property is identified by a QName. A property is an assertion or constraint on a specific RM capability and its value(s) associated with WSDL elements.

Typically properties are associated with a feature (but are not required to) and are described in a feature specification. The QName identifier of a property uniquely identifies the property. Recognizing the property QName identifier is considered to be equivalent to understanding the semantics associated with that property. The property QName identifier typically points a global XML Schema element declaration. A property specification typically specifies the schema that contains this global element declaration. A constraint on the set of values that a property can have is specified by a QName that identifies a XML Schema type.

```
<fnp:property name="xs:QName">
  [<fnp:value>xs:anyType</fnp:value> |
          <fnp:constraint>xs:QName</fnp:constraint>]
  [<extensibility-element/>]*
</fnp:property>
```

III. WS-Reliability Feature

The WS-Reliability feature is identified by the URI

"http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"

This feature URI identifies the WS-Reliability specification. Understanding this URI implies understanding the WS-Reliability specification.

IV. WS-Reliability Properties

This section identifies properties for the WS-Reliability specification. Typically these properties would be scoped within the feature identified by the URI

"http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"

IV.A. Guaranteed Delivery Property

This property is identified by the QName "wsrm:GuaranteedDelivery" and corresponds to the semantics specified by the WS-Reliability guaranteed delivery semantics. The type of this property is "xs:boolean".

IV.B. Duplicate Elimination Property

This property is identified by the QName "wsrm:NoDuplicateDelivery" and corresponds to the semantics specified by the WS-Reliability duplicate elimination semantics. The type of this property is "xs:boolean".

IV.C. Message Ordering Property

This property is identified by the QName "wsrm:OrderedDelivery" and corresponds to the semantics specified by the WS-Reliability message ordering semantics. The type of this property is "xs:boolean".

IV.D. Reply Pattern Property

This property is identified by the QName "wsrm:ReplyPattern" and corresponds to the semantics specified by the WS-Reliability reply pattern options. The type of this property is "xs:String". (values: Response, Poll, Callback)

V. Other Reliability Properties

In addition to the properties defined in section III, there are WS-Reliability properties that are used on the Sender side (usually the client side and therefore do not occur in the WSDL document).

This section identifies such properties. These properties MUST NOT be specified in the WSDL document. How the properties are specified and/or represented does not affect interoperability as these properties are client-side only properties. They are defined here for convenience only.

V.A. Group Expiry Time

This property is identified by the QName "wsrm:GroupExpiryTime" and corresponds to the semantics specified by the WS-Reliability group expiration time. The type of this property is xs:duration.

Note: The expiry time is calculated at the time a message is sent, but adding this duration to the time the message is sent.

V.B. Group Maximum Idle Duration

This property is identified by the QName "wsrm:GroupMaxIdleDuration" and corresponds to the semantics specified by the WS-Reliability group maximum idle duration. The type of this property is xs:duration.

V.C. Message Expiration Time

This property is identified by the QName "wsrm:ExpiryTime" and corresponds to the semantics specified by the WS-Reliability message expiration time. The type of this property is xs:duration.

Note: The expiry time is calculated at the time a message is sent, but adding this duration to the time the message is sent.

V.D. Retry Maximum Time

This property is identified by the QName "wsrm:RetryMaxTimes" and corresponds to the semantics specified by the WS-Reliability maximum retry times. The type of this property is xs:int.

V.E. Retry Time Interval

This property is identified by the QName "wsrm:RetryTimeInterval" and corresponds to the semantics specified by the WS-Reliability retry time interval. The type of this property is xs:duration.

V.F. ReplyTo URI

This property is identified by the QName "wsrm:ReplyTo" and corresponds to the semantics specified by the WS-Reliability reply-to. The type of this property is xs:anyURI.

VI. Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:wsrm="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
   targetNamespace="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
   elementFormDefault="qualified" >

   <!-- properties to be used in WSDL -->
   <xs:element name="GuaranteedDelivery" type="xs:boolean"/>
   <xs:element name="NoDuplicateDelivery" type="xs:boolean"/>
   <xs:element name="OrderedDelivery" type="xs:boolean"/>
   <xs:element name="ReplyPattern" type="xs:string"/>

   <!-- properties to be used on the client side -->
   <xs:element name="GroupExpiryTime" type="xs:duration"/>
   <xs:element name="GroupMaxIdleDuration" type="xs:duration"/>
   <xs:element name="ExpiryTime" type="xs:duration"/>
   <xs:element name="RetryMaxTimes" type="xs:int"/>
   <xs:element name="RetryTimeInterval" type="xs:duration"/>
   <xs:element name="ReplyTo" type="xs:anyURI"/>
</xs:schema>
```

VII. Examples

VII.A Example for the "all" compositor

```
<wsdl11:portType name="Example-1">
  <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
   <fnp:feature uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
     <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
       <fnp:property name="wsrm:DuplicateElimination">
         <fnp:value>true</fnp:value>
       </fnp:property>
       <fnp:property name="wsrm:OrderedDelivery">
         <fnp:value>true</fnp:value>
       </fnp:property>
```

```
      <fnp:property name="wsrm:GuaranteedDelivery">
        <fnp:value>true</fnp:value>
      </fnp:property>
    </fnp:compositor>
  </fnp:feature>
 </fnp:compositor>
 ...
</wsdl11:portType>
```

In the example above, the reliability feature identified by URI "http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/" is required by the portType. This feature consists of three properties, all of which are required because of the semantics of the 'all' compositor that composes the three properties.

VII.B Example for the "choice" compositor:

```
<wsdl11:binding name="Example-2">
 <fnp:compositor uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
   <fnp:feature uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
     <fnp:compositor uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/fnp/compositors/choice">
       <fnp:property name="wsrm:ReplyPattern">
         <value>Response</value>
       </fnp:property>
       <fnp:property name="wsrm:ReplyPattern">
         <value>Callback</value>
       </fnp:property>
       <fnp:property name="wsrm:ReplyPattern">
         <value>Poll</value>
       </fnp:property>
     </fnp:compositor>
   </fnp:feature>
 </fnp:compositor>
 ...
</wsdl11:binding>
```

In the example above, the reliability feature identified by URI "http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/" is required by the portType. This feature consists of three properties, of which the client must choose one.

VII.C Example for the "one-or-more" compositor:

```
<wsdl11:portType name="Example-3">
  <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
    <fnp:feature uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
      <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/one-or-more">
        <fnp:property name="wsrm:DuplicateElimination">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrm:OrderedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrm:GuaranteedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
      </fnp:compositor>
    </fnp:feature>
  </fnp:compositor>
  ...
</wsdl11:portType>
```

VII.D Example for the "zero-or-more" compositor:

```
<wsdl11:portType name="Example-4">
  <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/all">
    <fnp:feature uri="http://www.oasis-open.org/committees/wsrm/schema/1.1/feature/rel/"
      <fnp:compositor uri="http://www.oasis-
open.org/committees/wsrm/schema/1.1/fnp/compositor/zero-or-more">
        <fnp:property name="wsrm:DuplicateElimination">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrm:OrderedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrm:GuaranteedDelivery">
          <fnp:value>true</fnp:value>
```

```
1918        </fnp:property>
1919      </fnp:compositor>
1920    </fnp:feature>
1921  </fnp:compositor>
1922  ...
1923 </wsdl11:portType>
```

Appendix B. 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 (Secretary), 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
- Mark Peel, Novell
- Sunil Kunisetty (Secretary), Oracle
- Anish Karmarkar, Oracle
- Jeff Mischkinsky, Oracle
- Marc Goodner (Secretary), SAP
- Pete Wenzel, SeeBeyond Technology Corporation
- Doug Bunting (Secretary), Sun Microsystems

- Tony Graham, Sun Microsystems
- Chi-Yuen Ng, University of Hong Kong
- Patrick Yee, University of Hong Kong
- Prasad Yendluri, webMethods, Inc.
- Scott Werden, WRQ, Inc.

And the following people made conditions to produce Ver1.0 of this specification:

- Colleen Evans, Sonic Software Corporation
- Dave Chappell, Sonic Software Corporation
- Doug Bunting, Sun Microsystems, Inc.
- George Tharakan, Sun Microsystems, Inc.
- Hisashi Shimamura, NEC Corporation
- Jacques Durand, Fujitsu Software Corporation
- Jeff Mischkinsky, Oracle Corporation
- Katsutoshi Nihei, NEC Corporation
- Kazunori Iwasa, Fujitsu Limited
- Martin Chapman, Oracle Corporation
- Masayoshi Shimamura, Fujitsu Limited
- Nicholas Kassem, Sun Microsystems, Inc.
- Nobuyuki Yamamoto, Hitachi Limited
- Sunil Kunisetty, Oracle Corporation
- Tetsuya Hashimoto, Hitachi Limited
- Tom Rutt, Fujitsu Software Corporation
- Yoshihide Nomura, Fujitsu Limited
- Akira Ochi, Fujitsu Limited
- Hirotaka Hara, Fujitsu Limited
- Hiroyuki Tomisawa, Hitachi Limited
- Katsuhisa Nakazato, Fujitsu Limited
- Masahiko Narita, Fujitsu Limited
- Nobuyuki Saji, NEC Corporation
- Shuichi Imabayashi, Fujitsu Limited

Appendix C. 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
WD-0.85	-2004-01-06	Kazunori Iwasa	Section2.4, Section2.5, and Section 3.1.1 are updated to incorporate resolutions for Rel-52, Rel-57, and Rel-82.

Rev	Date	By Whom	What
WD-0.86	-2004-01-14	Kazunori Iwasa	Updated for comments at : http://www.oasis-open.org/archives/wsrm/200401/msg00010.html except for: - More faults for Tables1 (Need to list up all faults) - Section2.4 Line#259 in Spec20040106 (Ver0.85): It should read "after the message has been processed and delivered to the "next processing layer". (Need to confirm with TC for this change, since the current text was approved one.) - Figure1,2,and3 “New processor Entity” (Want to confirm with TC member) -New terminologies for “Group Termination”, “Removal”, “Complete”, and others. (Needs definitions) -- And other editorial updates.

Rev	*Date*	*By Whom*	*What*
WD-0.87	-2004-01-15	Kazunori Iwasa	Updated for: Prelim Wed minutes on 1/14/2004 at: http://www.oasis-open.org/archives/wsrm/200401/msg00038.html. It includes: Rel33: New definitions in 1.8(deliver, submit, notify) Rel37: editorial change in 3.1.1 Rel40: editorial change in 3.1.3 Rel44: updates for 3.1.1 Rel51: change definition for Message Acknowledgment Rel52: Moved some of 3.1.1(line546-571) to 2.5.1 Rel98: removed informative notes in 2.4 Tables: Changed “Required” to “Cardinality” (Yes-1, No-0) The following resolutions are not updated yet: Rel 83-86 and 56: Change of element names and location (Eg. GroupId -> MessageId)

Rev	Date	By Whom	What
WD-0.88	-2004-01-16	Kazunori Iwasa	Updated for: 1) Prelim minutes for Thursday on 1/15/2004 at: http://www.oasis-open.org/apps/org/workgroup/wsrm/email/archives/200401/msg00053.html 2) Remaining items including: Rel36: Message ID -> Message Identifier Rel37: Reference for RFC2392 Changed element names and location (Eg. GroupId -> MessageId) And others. The following items are still in progress: Rel22: usage for MUST, MAY, Should, Optional

Rev	Date	By Whom	What
WD-0.90	-2004-01-26	Kazunori Iwasa	Updated for remaining action items at: http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/5089/Minutes-Jan04f2f.htm This includes : 1) 2.4: Message Identifier -> MessageId Sequence Number -> SequenceNum Included “Next processing layer” 2) 2.11: Chart is updated (Now2.9) 3) 3.1.1and 3.1.2: two group attributes were moved from MessageId to SequenceNum 4) 3.1.4 Response : Some sentences are added to restrict sending back previous Acknowledgment message in the other Response. 5) 3.3, 3.3.1 and 3.4.1: MessageId -> RefToMessageIds value -> groupId 6) Section4: Replaced with new text 7) Appendix A:Replaced with new schema Other changes includes: Cover page: location of the spec and errata were added. Section 1 and 2: Editorial review 1.9:New section was added (RM agreement) 1.6: Description for WS-I BP1.0 was included. 6.2: Added non-normative Reference for SOAP messages with Attachments

Rev	Date	By Whom	What
			Remaining Action items and editorial changes for 2004-01-26(0.90): 1. Consistency of word: e.g. sender RMP, sending RMP or sender's RMP 2.Removing MAY, Optional 3.NotSupportedFeaturesFault 4.Explanatin for cardinality and others 5.SOAP1.2 statement in the new section in 3 on rm:fault element 6.Update fault in section3 7.Examples
WD-0.91	-2004-02-02	Kazunori Iwasa	Updated for remaining action items at: http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/5089/Minutes-Jan04f2f.htm or http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/5090/Action%20Item%20List%20from%20Jan%20Face%20To%20Face%20Meeting.htm This includes : AI10.Done. Throughout the spec. AI16. Done. Section 1.5. AI20. Done. Section 3.5 is added. AI22. Done. Section1.5. AI24. Done. Section 3 AI28. (Still working) AI8, 9 and 25. Done. Section 2.9. AI4 & 19. Done. Section 2.10 is added. Schema was replaced with ws-reliability-2004-01-27.xsd. Table numbers were maintained sequentially. And other editorial updating.

Rev	*Date*	*By Whom*	*What*
WD-0.92	-2004-02-09	Kazunori Iwasa	This was updated for: AI10: Rel22: Remaining updates are done AI20: Section3.5: Added “SOAP1.2 can't use Fault element.” Section3.4.1.1: from and to attribute are included here. Section1.8: Definition of PollRequest was added. Section5: Examples are added. And editorial updates.
WD-0.93	-2004-02-10	Kazunori Iwasa	This was updated for: Section1: Editorial updates.
WD-0.94	-2004-02-12	Kazunori Iwasa	This was updated for: Section2.8: Section number correction Section3.1.1 and 3.1.2: MessageId text http://www.oasis-open.org/archives/wsrm/200402/msg00038.html http://www.oasis-open.org/archives/wsrm/200402/msg00068.html Rel108: Section1.7 and 3.3: Clarified that PollRequest can be used for any RM-Reply pattern, and a reply to PollRequest only includes successfully delivered messages. Section4: Example11 is added. Numbering of examples are corrected sequentially. Section2: Some editorial comments at: http://www.oasis-open.org/archives/wsrm/200402/msg00080.html Rel76: Section 3.3: Agreed text was added. Appendix B: Acknowledgments for Ver1.0 was added. And some other editorial updates.

Rev	*Date*	*By Whom*	*What*
WD-0.98	-2004-02-26	Kazunori Iwasa	This was updated with minutes at: http://www.oasis-open.org/apps/org/workgroup/wsrm/download.php/5630/MinutesWSRMTC021704.htm and http://www.oasis-open.org/archives/wsrm/200402/msg00223.html Rel102: Remove Section2.7 Rel108/115: Remove section 3.5 Updates on Section 3.4 Updates on Section 4 Rel113: Section 2.4 Section 2.8.1 New issue: removing MessageHeader throughout the spec Editorial reshuffle with: http://www.oasis-open.org/archives/wsrm/200402/msg00161.html Appendix A: Schema Appendix B: Acknowledgments And minor editorial updates
WD-0.99	-2004-03-03	Kazunori Iwasa	This was updated with minutes at: http://www.oasis-open.org/apps/org/workgroup/wsrm/email/archives/200403/msg00035.html except for Minutes 4.4 Rel119, which requires discussion. And also minor editorial updates were done.
WD-0.991	-2004-03-04	Kazunori Iwasa	This was updated with : Editorial updates : Bullet list consistency Appendix A: Two new members are added in the Acknowledgments.

Rev	Date	By Whom	What
WD-0.992	-2004-03-10	Kazunori Iwasa	This was updated with a minutes at: http://www.oasis-open.org/archives/wsrm/200403/msg00084.html And some editorial updates.

1988

1989

Appendix D. 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 E. 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.