Web Services Reliable Messaging TC
WS-Reliability 1.1

Working Draft 1.05, 7 July 2004

Document identifier:
wd-web services reliable messaging tc-ws-reliability 1.1-1.05

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 updated aperiodically on no particular schedule.

Committee members should send comments on this specification to the wsrm@lists.oasis-open.org list. Others should use the comment form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=wsrm.

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 .......................................................................................................................... 4
  1.1 Purpose of WS-Reliability ................................................................................................. 4
  1.2 Definition and Scope of Reliable Messaging ................................................................... 4
  1.3 Notational Conventions .................................................................................................. 5
  1.4 Relation to Other Specifications ...................................................................................... 6
  1.5 Terminology ...................................................................................................................... 7
2 Messaging Model ................................................................................................................... 10
  2.1 Messaging Context ........................................................................................................... 10
  2.2 RMP Operations and Their Invocation ........................................................................... 10
  2.3 Assumed SOAP Message Exchange Patterns ............................................................... 11
  2.4 Binding between WSDL Operation Types, RMP Invocations, and SOAP MEPs ........... 11
  2.5 Message Reply Patterns .................................................................................................. 12
  2.6 Message Identification and Grouping .............................................................................. 14
3 Reliability Agreement and Features ..................................................................................... 15
  3.1 RM Agreement ................................................................................................................ 15
  3.2 Main Reliability Features ................................................................................................ 17
4 Message Format .................................................................................................................... 20
  4.1 Structure .......................................................................................................................... 20
  4.2 Request Element .............................................................................................................. 21
  4.3 PollRequest Element ...................................................................................................... 29
  4.4 Response Element .......................................................................................................... 33
  4.5 Fault Codes For Reliable Messaging Failures .................................................................. 36
5 Operational Aspects and Semantics .................................................................................... 41
  5.1 Message Group Life Cycle ............................................................................................... 41
  5.2 Reliability of WSDL Operations ..................................................................................... 45
  5.3 Attachments .................................................................................................................... 46
6 HTTP Binding ....................................................................................................................... 47
  6.1 Reliable Messaging with Response RM-Reply Pattern .................................................. 47
  6.2 Reliable Messaging with Callback RM-Reply Pattern .................................................. 49
  6.3 Reliable Messaging with Poll RM-Reply Pattern ........................................................... 52
7 Conformance ........................................................................................................................ 57
8 References ............................................................................................................................. 58
Appendix A. Schema (Normative) ........................................................................................... 61
Appendix B. WS-Reliability Features, Properties and Compositor (Normative and Optional) ....... 62
<table>
<thead>
<tr>
<th>Appendix C. Acknowledgments</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix D. Notices</td>
<td>72</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose of WS-Reliability

WS-Reliability is a SOAP Module (as defined by [SOAP 1.2]), which fulfills reliable messaging requirements that are critical to some applications of Web Services. SOAP ([SOAP 1.1] and [SOAP 1.2]) over HTTP [RFC2616] are not sufficient when an application-level messaging protocol must also guarantee some level of reliability and security. This specification defines reliability in the context of current Web Services standards. This specification has been designed to be for used in combination with other complementary protocols (see Section 1.4), and has builds upon previous experiences (e.g., ebXML Message Service [ebMS].)

1.2 Definition and Scope of Reliable Messaging

Reliable Messaging (RM) is the execution of a transport-agnostic protocol based on SOAP for providing quality of service in the reliable delivery of messages. There are two aspects to Reliable Messaging, which both need to be equally addressed when specifying RM features:

1. The “wire” protocol aspect. Under this aspect, RM is a protocol – which includes specific message headers, and specific message choreographies – between a sending party and a receiving party.

2. The quality of service (QoS) aspect. Under this aspect, RM defines a quality of messaging service to the communicating parties, also defined as the messaging service user parties. This assumes a protocol between the provider of this service (i.e., the reliable messaging middleware) and the users of this service. This protocol is defined by a set of abstract operations: Submit, Deliver, Notify, Respond.

Reliable messaging requires the definition and enforcement of contracts between:

- The Sending and Receiving message processors (contracts about the wire protocol)
- The messaging service provider and the messaging service users (contracts about quality of service).

Each major RM feature will be defined as a composition of these two types of contract.

Example: Guaranteed message delivery will be defined as both (1) a messaging protocol involving Acknowledgment Indications and specific message headers, and (2) as a rule that guarantees that after “submit” has been invoked for a message on the sending side, then the “deliver” operation will be invoked on the receiving side for this message, or else “notify” (of failure) will be invoked on the sending side.
Figure 1 shows the two reliability contracts, between a producer of reliable messages, a consumer
of reliable messages, and the two Reliable Messaging Processors (RMPs) they use.

Note: This specification does not make any assumption about the implementation of a
messaging service user component (Producer or Consumer components in Figure 1); such a
component could be an application, a queuing or logging system, a database, a
SOAP node, or the next handler in the message processing chain. The QoS contracts only
concern the conditions of invocation of the “Deliver”, “Submit”, “Respond” and “Notify” operations.
The interpretation of these operations is a matter of implementation.

The current specification defines the following reliability features:

- 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 group of messages.

Some messaging features are out of scope for this specification. They are:

- Application level synchronous messaging. Synchronous messaging applications
  require immediate knowledge of the message status (e.g., error) and retain complete
  control on the corrective actions (e.g., message resending).
- 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 them.

Reliability is often associated with QoS quantitative measures in areas other than Web services
(e.g., networking). Thresholds such as rate of failures, minimal size of persistent store, average
latency, and more generally quantitative measures that may appear in service level agreements,
are out of scope for this version.

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 as follows:

<table>
<thead>
<tr>
<th><strong>Label</strong></th>
<th><strong>Meaning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>A constraint on the number of instances of an item type which may be present in an enclosing item. (e.g., &quot;0 or 1&quot; means the message may not include the element, or it may include the element only once.)</td>
</tr>
<tr>
<td>Value</td>
<td>A type or format for a value of the element.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Attribute names for the element. The type or format for its value is also included in parentheses.</td>
</tr>
<tr>
<td>Child elements</td>
<td>Child element for the element.</td>
</tr>
</tbody>
</table>

This specification uses the following namespace prefixes:

<table>
<thead>
<tr>
<th><strong>Prefix</strong></th>
<th><strong>Namespace</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>soap</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a></td>
</tr>
<tr>
<td>wsrn</td>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd</a></td>
</tr>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema/">http://www.w3.org/2001/XMLSchema/</a></td>
</tr>
<tr>
<td>wsd11</td>
<td><a href="http://schemas.xmlsoap.org/wsdl/">http://schemas.xmlsoap.org/wsdl/</a></td>
</tr>
<tr>
<td>fnp</td>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd</a></td>
</tr>
<tr>
<td>wsrnfp</td>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd</a></td>
</tr>
<tr>
<td>ref</td>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd</a></td>
</tr>
</tbody>
</table>

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

XPaths [Xpath1.0] are used in titles and other places in Section 4.

### 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 features, expressed as extension header blocks 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 2004: 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 2004 [WSS].

WS-I Basic Profile 1.0: This specification is compliant with WS-I Basic Profile 1.0a [WS-I BP1.0].

1.5 Terminology

Some of these definitions may forward-reference other definitions, either within or outside of the terminology section.

Reliable Messaging (RM):
The act of processing the set of transport-agnostic SOAP Features defined by WS-Reliability, resulting in a protocol which guarantees certain qualities of service. This includes the processing of Acknowledgment indications, re-sending of messages, duplicate message elimination, and message ordering.

Reliable Messaging Processor (RMP):
A SOAP Node (as defined by [SOAP 1.2]), or a subset or superset thereof, capable of performing Reliable Messaging as described by this specification. With regard to the transmission of a Reliable Message from one RMP to another, the former is referred to as the Sending RMP, and the latter as the Receiving RMP. An RMP may act in both roles.

Reliable Message:
A message for which some level of reliable delivery is required.

Payload:
Subset of message data that is intended for the consumer of the Reliable Message.

Producer (or Payload Producer)
An abstract component which produces the payload of a message to be sent out. An example of Producer is an application component able to invoke an RMP for sending the payload.

Consumer (or Payload Consumer)
An abstract component which consumes the payload of a received message after it has been processed by the Receiving RMP. Examples of Consumers are: an application component called back when a message is received, a queuing device where received payloads are stored.

Deliver:
An abstract operation supported by the RMP. When invoked on a Receiving RMP, the operation makes the payload of one Reliable Message available to the Consumer. For example, in one specific implementation choice, the payload is placed into a queue by the Receiving RMP to be consumed by an application component.

Submit:
An abstract operation supported by the RMP. When invoked, the operation transfers payload data from the Producer to the Sending RMP (e.g., a request to the Sending RMP to take responsibility for the Reliable Message).
Respond:
An abstract operation supported by the RMP. When invoked, the operation transfers payload data
from the Consumer to a Receiving RMP.

Notify:
An abstract operation supported by the RMP. When invoked, the operation makes available a
response payload received by the Sending RMP, to the Producer, or makes available to the
Producer the status of a Reliable Message (e.g., a notification that the Sending RMP failed to
send a Reliable Message).

Message Identifier:
A value or a combination of values in the message header that uniquely identifies a Reliable
Message. This identifier is only meaningful to the reliability features described here.

Duplicate Message:
A message is a duplicate of another message if it has same message identifier.

Message Delivery:
The action of invoking the “deliver” operation for a Reliable Message. This action marks the end of
the RMP processing for this message.

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

Reliable Messaging Fault Indication (RM Fault):
An indication which refers to a previous message that encountered a Reliable Messaging fault
condition at the Receiving RMP. It signals to the Sending RMP of the referred message that there
was a failure to invoke the deliver operation for the message.

Reliable Messaging Reply (RM-Reply):
An indication referring to a previous Reliable Message, that is, either an Acknowledgment
Indication or a Reliable Messaging Fault Indication. For the Callback and Poll RM-Reply Patterns,
RM-Repies for multiple Reliable Messages MAY be included in a single Reliable Messaging
response.

Response, Callback, and Poll RM-Reply Patterns:
See Section 2.5.

PollRequest Message:
A polling message for Acknowledgment Indication(s). A Sending RMP may send a PollRequest
Message for polling of Acknowledgment Indication(s) regardless of RM-Reply Pattern of the
original Reliable Message. For example, the Sending RMP may send PollRequest Message to
retrieve Acknowledgment Indication for a message originally sent using Callback RM-Reply
Pattern.

Intermediary:
A SOAP node between a Sending RMP and a Receiving RMP.
Publish:

An abstract operation making an RM-Reply available to its destination.

For the various RM-Reply Patterns this entails:

- Response RM-Reply Pattern: publishing the reply requires sending the RM-Reply as described in Section 2.5.1.
- Callback RM-Reply Pattern: publishing the reply requires sending a callback message including the RM-Reply information as described in Section 2.5.2.
- Poll RM-Reply Pattern: publishing the reply requires making the RM-Reply information available to be returned to the Sending RMP in response to a PollRequest, as described in Section 2.5.3.
2 Messaging Model

2.1 Messaging Context

The Reliable Messaging Model described in this document makes the following assumptions about SOAP messaging, and its relation to the RMP behavior:

- **Intermediary transparency.** SOAP Intermediaries do not play any active role in the reliability mechanisms. They can be abstracted from the communication between Sending RMP and Receiving RMP, which are the only parties involved in implementing the RM protocol, e.g., for handling RM- Replies. There is no other role for an RMP than Receiving RMP or Sending RMP. Figure 2 illustrates this model.

- **Message integrity.** For the reliability mechanisms described here to fulfill the reliability contract, it is strongly RECOMMENDED that message header integrity be guaranteed end-to-end by using adequate security options, such as those described in WS-Security.

2.2 RMP Operations and Their Invocation

The four operations (Submit, Deliver, Respond, and Notify) are used to model the reliability contracts between an RMP with its users (Producer, Consumer), as well as to relate these contracts to message exchange patterns.

An RMP acting in the role of a Sending RMP MUST implement Submit, and notification of failure (Notify). An RMP acting in the role of a Receiving RMP MUST implement Deliver.

When an RMP supports both Deliver and Respond, then it MUST be able to associate a payload obtained via Respond, with a payload previously delivered via Deliver, based on Consumer demand. An example of using such an association is the binding of these payloads to the same instance of a request-response message exchange pattern. Because there is not always an invocation of Respond for each invocation of Deliver, an RMP implementation MUST have knowledge of whether or not an invocation of Deliver will be matched or not by an invocation of Respond, so that no messaging resource (e.g., a transport connection) is held for too long.

An RMP which supports both Submit and Notify MUST be able to associate a failure notification (Notify) with the related submitted payload (Submit). In case the notification of payload is supported, the RMP MUST be able to associate a received payload (Notify) with a previously submitted payload (Submit).

![Figure 2 Messaging Model](image-url)
The basic exchange patterns described in the following section are derived from the above messaging assumptions. Reliability features defined in this specification will in turn rely on these patterns.

### 2.3 Assumed SOAP Message Exchange Patterns

Although SOAP was initially defined as a one-way messaging protocol, support for other message exchange patterns (MEPs) has been described in SOAP 1.2 Part 2. Described below are two MEPs – called here SOAP MEPs – of interest for the RM features specified here are described below.

- **SOAP One-way -MEP:**
  - From an RMP perspective, support for this MEP assumes the following:
    - The Sending RMP (as a SOAP node) is able to initiate the sending of a SOAP envelope over the underlying protocol (i.e., not as a result of a previous protocol action, such as an HTTP GET or even POST).
    - No response containing a SOAP envelope is sent back, unlike in the SOAP request-response MEP (although a non-SOAP response (e.g., an HTTP error code) may be sent back, and correlated for the purpose of communicating success or error status, e.g., an HTTP error code).

- **SOAP Request-response -MEP:**
  - This MEP is defined in SOAP 1.2 Part 2. From an RMP perspective, support for this MEP assumes the following:
    - The Sending RMP is able to initiate the sending of a SOAP envelope over the underlying protocol, and the Receiving RMP can send back a message with a SOAP envelope (called a response); after somehow associating somehow the response with the request.
    - The Sending RMP can correlate the received response with the request it has sent.

### 2.4 Binding between WSDL Operation Types, RMP Invocations, and SOAP MEPs

#### 2.4.1 Binding between WSDL Operation Types and RMP Invocations

This specification supports Reliable Messaging capabilities for WSDL 1.1 [WSDL 1.1] One-way and Request-Response operation types only. Assuming two RMPs (Sending RMP or S-RMP and Receiving RMP or R-RMP) respectively coupled with a Web service client and a Web service host, WSDL operation types MUST bind with the RMP operations in the following way:

- A successful WSDL operation of type One-way maps to a sequence of RMP invocations of the form: S-RMP.Submit(p) + R-RMP.Deliver(p), where (p) is the payload sent in the request (input message) of the operation described in WSDL.
- A successful WSDL operation of type Request-Response maps to a sequence of RMP invocations of the form: S-RMP.Submit(p) + R-RMP.Deliver(p) + R-RMP.Respond(p2) + S-RMP.Notify(p2), where (p) is the payload sent in the request, and (p2) is the payload returned in the response (output message) of the operation described in WSDL.
2.4.2 Binding between RMP Invocations and SOAP MEPS

A sequence of RMP invocations of the form:

\[ S-RMP.\text{Submit}(p) + R-RMP.\text{Deliver}(p), \]

(for example, as generated by the execution of a WSDL One-Way operation) may bind to:

- (b1) the message of a SOAP One-way MEP which contains the submitted (Submit) payload.
- (b2) a request in a SOAP Request-response MEP which contains the submitted (Submit) payload. In that case, the response message is NOT bound to any RMP operation.

A sequence of RMP invocations of the form:

\[ S-RMP.\text{Submit}(p) + R-RMP.\text{Deliver}(p) + R-RMP.\text{Respond}(p2) + S-RMP.\text{Notify}(p2), \]

where the Consumer requested the association of payloads (p) and (p2) (for example, as in the execution of a WSDL Request-response operation), may bind to:

- (b3) two instances of a SOAP One-way MEP, one message for the submitted (Submit) payload, one message for the response (Respond) payload.
- (b4) a full SOAP Request-response MEP, where the request carries the submitted (Submit) payload, and the response carries the payload passed by Respond.
- (b5) the requests of two separate instances of SOAP Request-response MEP initiated respectively by Sending and Receiving RMPs. In that case, the response messages are NOT bound to any RMP operation.

These binding options will restrict the use of message reply patterns described in 2.5. Also, they will be restricted based on additional compliance requirements. For example, conformance of message exchanges with WS-I BP 1.0 requires the use of bindings (b1) and (b4) above, given the binding between SOAP MEPS and HTTP described in Section 6.

2.5 Message Reply Patterns

1. There are three ways to publish an RM-Reply (Acknowledgment Indication or Fault Indication):

2.5.1 Response RM-Reply Pattern

When the Response RM-Reply Pattern is in use, the SOAP Request-response MEP MUST be supported. The following exchange pattern occurs:

Step 1: The Reliable Message is sent by the Sending RMP in a request of a SOAP MEP instance.

Step 2: The RM-Reply is sent by the Receiving RMP in the response message of the same SOAP MEP instance.

Figure 3 shows this reply pattern.
2.5.2 Callback RM-Reply Pattern

When the Callback RM-Reply Pattern is in use, either the SOAP Request-response or the SOAP One-way MEP MUST be supported. The following exchange pattern occurs:

Step 1: The Reliable Message is sent by the Sending RMP in a request of a SOAP Request-response MEP instance, or in the message of a SOAP One-way MEP.

Step 2: The RM-Reply is sent by the Receiving RMP either in a new request message of a different SOAP Request-response MEP instance, or in the message of a different SOAP One-way MEP instance.

Figure 4 shows this reply pattern.

2.5.3 Poll RM-Reply Pattern

When the Poll RM-Reply pattern is in use, either the SOAP Request-response or the SOAP One-way MEP MUST be supported. The following exchange pattern occurs:

Step 1: The Reliable Message is sent by the Sending RMP in a request of a SOAP Request-response MEP instance, or in the message of a SOAP One-way MEP.

Step 2: A message with a PollRequest element is issued by the Sending RMP in a new SOAP MEP, acting as a request for Acknowledgment. If the PollRequest is synchronous, the request of a SOAP Request-response MEP instance must be used. If the PollRequest is asynchronous, either a request of a SOAP Request-response MEP or the message of a One-way MEP may be used.

Step 3: The RM-Reply is sent back from the Receiving RMP either (if synchronous polling) in the response message of the same SOAP MEP instance that carried the PollRequest, or (if the polling is asynchronous) in any message from a different SOAP MEP instance (either Request-response or One-way) if the polling is asynchronous.
This reply pattern may be used in situations where it is inappropriate for the Sending RMP of Reliable Messages to receive underlying protocol requests, e.g., due to security restrictions. Figure 5 shows this reply pattern.

Figure 5 Poll RM-Reply Pattern

2.6 Message Identification and Grouping

Every Reliable Message MUST contain a globally unique Message Identifier. This Message Identifier relies on the notion of a group. A message always belongs to a group. A group of messages is sent from the Sending RMP to the Receiving 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 that is unique within a group. More precisely, a message is uniquely identified as follows:

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

2. When the message belongs to a group of several messages: the message is identified by the group ID and a unique sequence number.
3 Reliability Agreement and Features

3.1 RM Agreement

3.1.1 Definition
An agreement for messaging reliability or RM Agreement describes which reliability features a sending party and a receiving party have agreed to use when exchanging a set of messages. The RM Agreement can be seen as a contract at two levels: (1) quality of service (QoS) about the conditions and quality of message delivery to the consumer party, and (2) protocol features, including timing parameters and details about choreography between Sending RMP and Receiving RMP.

3.1.2 RM Agreement Items
An RM Agreement is a list of Agreement Items. An RMP implementation MUST be capable of:
1. taking knowledge (e.g., via either configuration, or via an API call, or via a message, or via the result of an algorithm) of a set of values that represent the RM Agreement Items described in this specification, and
2. processing them according to the semantics described in this specification.

Table 3 shows the Agreement Items that this specification uses. Each item is listed with its possible values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GuaranteedDelivery</td>
<td>enabled/disabled</td>
<td>For setting Guaranteed Delivery. (See Section 3.2 for details.)</td>
</tr>
<tr>
<td>NoDuplicateDelivery</td>
<td>enabled/disabled</td>
<td>For setting message delivery without duplicates, or Duplicate Elimination. (See Section 3.2 for details.)</td>
</tr>
<tr>
<td>OrderedDelivery</td>
<td>enabled/disabled</td>
<td>For setting Guaranteed Message Ordering. (See Section 3.2 for details.)</td>
</tr>
<tr>
<td>GroupMaxIdleDuration</td>
<td>number of seconds</td>
<td>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.</td>
</tr>
<tr>
<td>GroupExpiryTime</td>
<td>date/time</td>
<td>For setting the date and time after which the group can be terminated.</td>
</tr>
<tr>
<td>ExpiryTime</td>
<td>date/time</td>
<td>For setting the date and time after which a message must not be delivered to the Consumer.</td>
</tr>
<tr>
<td>ReplyPattern</td>
<td>&quot;Response&quot;, &quot;Callback&quot;, &quot;Poll&quot;</td>
<td>For setting the mode of response for Acknowledgments or Faults.</td>
</tr>
</tbody>
</table>
3.1.3 Messaging Scope

As messages may be associated with a group, the messaging scope of these Agreement items may vary, as messages may be associated with a group. There are two scopes to consider:

- **Group scope**: all messages sent within a group.
- **Message Scope**: a single message.

Agreement items relate to a particular scope, e.g., for example, ExpiryTime is affecting each message separately, while GroupExpiryTime is an Agreement item about groups.

The smallest scope of applicability for each RM Agreement item is:

**Message scope**:
- ExpiryTime
- ReplyPattern

**Group scope**:
- OrderedDelivery
- GuaranteedDelivery
- NoDuplicateDelivery
- GroupExpiryTime
- GroupMaxIdleDuration

An RMP MAY support message-scope RM Agreement items at group-scope level. For example, an RMP implementation may decide to provide a way to specify the same ExpiryTime value for all messages of a group, and not support setting different values for messages in a group.

An RMP MUST NOT support RM Agreement items at a scope that is lower than the smallest scope applicable. For example, an RMP implementation MUST NOT use different guaranteed delivery modes for different messages of a group. However, it is allowed to dynamically change the value of GroupExpiryTime or GroupMaxIdleDuration, pertaining to a group.

3.1.4 Rules

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 used for a messaging scope, then the item GroupMaxIdleTime MUST NOT be used; and vice versa.

3.1.5 Creation, Representation and Deployment of RM Agreements

The concrete representation of an RM Agreement is beyond the scope of this specification, as this may be part of a more general agreement that exceeds the reliability aspect. However, the RM Agreement determines the use of the reliability protocol and the behavior of RMPs. For these reasons, this specification describes the RM Agreement in an abstract way, simply as a list of (name, value) pairs called Agreement Items. This allows for describing the concrete effect of each Agreement Item on the message content and flow. Once there is a broad enough
consensus for using a particular representation for agreements, a future version of this specification will define a corresponding binding for RM Agreements.

The way an RM Agreement is established or communicated to each party is out of scope. However, one of the principles of this specification is that it should not be necessary to deploy an RM Agreement on both Sending RMP and Receiving RMP prior to executing business transactions. Only the Sending RMP needs to have knowledge of the RM Agreement initially. No prior communication of the agreement to the receiving party (RMP and its user) is required. The only input that the Receiving RMP will need in order to enforce the reliability requirements will be obtained from the header of received messages.

3.1.6 RM Capability

As a way to support the creation of RM Agreements, it may be useful for Web services providers to advertise somehow the reliability features (or RM Agreement Item values) that are supported by a deployed Web service. Such capabilities - called RM Capabilities, in contrast to agreements that involve both parties - may conveniently be associated with WSDL definitions. In support of this option, this specification proposes a concrete representation for these capabilities (see Appendix B).

3.2 Main Reliability Features

The main reliability features mentioned in Section 1 are formally described here in terms of requirements. This specification provides the means to enforce these requirements. A detailed description of protocol features implementing these means is given in Section 4 and beyond.

3.2.1 Guaranteed Delivery

Quality of Service requirements:

When the GuaranteedDelivery Agreement Item is enabled, one of the two following outcomes MUST occur for a payload submitted to the Sending RMP: either (1) the payload is successfully delivered by the Receiving RMP to the consumer party, or (2) the producer party is notified of a delivery failure.

Notes:

• This QoS feature only guarantees that when a payload is not delivered, the sender will always be notified. It is, however, impossible to guarantee this while at the same time guaranteeing that (1) and (2) will never occur together for the same message. A proper usage by an implementation of the protocol options described in this specification will, however, greatly reduce situations where both (1) and (2) occur.

• In the current specification, the GuaranteedDelivery agreement is defined for payloads passed to the Sending RMP via the Submit operation. An extension of this agreement to payloads passed via the Respond operation is out of scope for this specification.

Protocol requirements:

A Receiving RMP MUST publish the RM-Reply of any message that has been either delivered or faulted. In case of When the Sending RMP requires the Poll RM-Reply Pattern, the Sending RMP it MUST poll for all the messages it has sent.

A message resending technique combined with the acknowledgment and fault mechanism described here MUST be used in case of delivery failure. Parameters that control the resending policy (number of retries, frequency, etc.) are out of the scope of this specification. These...
parameters may be added to an RM Agreement, although the resending policy may need to be
dynamically adjusted depending on network conditions.

A Sending RMP that has not been able to receive an acknowledgment for a sent message, MUST
notify the Payload Producer of a delivery error.

A Receiving RMP MUST NOT publish a Reliable Messaging Fault for a delivered Message. The
RMP MUST NOT deliver a message for which a Reliable Messaging Fault has been published.

When resending a message, the Sending RMP MUST NOT modify the MessageId or any other
value in the reliability headers, including time-related values. It is RECOMMENDED to NOT
resend a message for which an RM-Reply with one of the following Fault types has been received:

- An Invalid Message Format fault code (Table 22)
- A NonSupportedFeature fault code
- A PermanentProcessingFailure fault code

3.2.2 Duplicate Elimination

Quality of Service requirements:

When the NoDuplicateDelivery Agreement item is enabled, a payload submitted only once to the
Sending RMP MUST NOT be delivered twice or more to the consumer party.

When NoDuplicateDelivery is enabled, an RMP MUST ensure that when delivering a payload
carried by a received message, no payload from a message received later with the same
Message Identifier as the message containing the first payload will ever be delivered to the
consumer party.

Note:

In the current specification, the NoDuplicateDelivery agreement is defined for payloads passed to
the Sending RMP via the Submit operation. An extension of this agreement to payloads passed
via the Respond operation is out of scope for this specification.

Protocol requirements:

An implementation of this specification must ensure the following invariants:

- Two message instances that carry different payloads MUST NOT share the same
  Message Identifier.
- Two message instances that share the same Message Identifier —(such as the
  resending mechanism generates—) MUST carry exactly the same payload(s) and the
  same reliability headers.

When the Response RM-Reply Pattern is requested with Duplicate Elimination for a Reliable
Message, and a resend of that message cannot be delivered to the Consumer by the Receiving
RMP because it is a duplicate of a previously delivered message, and when a Consumer
response payload is expected, the response of the SOAP MEP instance MUST contain one or the
other (but not both) of the following (but not both):

- a copy of the original response payload returned for that Message (in the SOAP Body)
in addition to the Acknowledgment Indication (in the SOAP Header), or;
- a SOAP server Fault (in the SOAP Body) in addition to the Acknowledgment Indication
  (in the SOAP Header).

The Sending RMP and Producer expect either a complete response or a SOAP Fault when using
the Response RM-Reply Pattern—and these two allowed behaviors satisfy that these expectations.
3.2.3 Guaranteed Message Ordering

Quality of Service requirements:

When the OrderedDelivery agreement is enabled, a sequence of payloads submitted to a sending RMP MUST be delivered in the same order by the receiving RMP to the consumer party. In addition, when the receiving RMP delivers one of these payloads, all previous payloads in the sequence MUST already have been delivered (no missing message allowed).

Note:

In the current specification, the OrderedDelivery agreement is defined for payloads passed to the sending RMP via the Submit operation. An extension of this agreement to payloads passed via the Respond operation is out of scope for this specification.

Protocol requirements:

Ordering is only supported over messages of the same group.

An implementation of this specification must ensure the following invariants, regarding the usage of sequence numbers (SequenceNum element):

- \[\text{The sequence number of messages sent by an RMP MUST reflect the order in which the payloads have been submitted by the producer party to the sending RMP.}\]
- \[\text{The messages received MUST be delivered according to the order expressed by their sequence numbers, which is the same as the submission order.}\]
- \[\text{From one sent message to the next in the same group, the sequence number MUST increase by one, starting with value 0.}\]
4 Message Format

4.1 Structure

Figure 6 shows the structure in the SOAP Envelope, as specified by the WS-Reliability protocol. On the left side of the figure, a Reliable Message is characterized by the presence of the wsrm:Request element. On the right side a response to a Reliable Message contains a wsrm:Response element. Both wsrm:Request and wsrm:Response elements may be found in the same message.

Figure 6 Structure of WS-Reliability elements
Figure 7 shows the structure of PollRequest message embedded in the SOAP Envelope.

**Figure 7  Structure of PollRequest message elements**

- soap:Envelope
  - soap:Header
    - wsrn:PollRequest
      - wsrn:RefToMessageIds *
      - wsrn:SequenceNumRange*
    - wsrn:ReplyTo
  - any
- soap:Body

The namespace [XML Namespaces] for reliable messaging defined in this specification is:

```
http://docs.oasis-open.org/wsrn/2004/06/wsrn-reliability-1.1.xsd
```

In a case where the text of the specification is shown to be in conflict with schema statements, the schema statement prevails. If a message contains additional elements not described in this specification, the Reliable Messaging Processor MUST ignore those elements.

Any of the following three elements can be a 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 elements. This element includes the following attribute and child elements:

- **SOAP mustUnderstand** attribute, as specified in Appendix A
- **MessageId** element
- **ExpiryTime** element
- **ReplyPattern** element
- **AckRequested** element
- **DuplicateElimination** element
- **MessageOrder** element

<table>
<thead>
<tr>
<th>Table 4 Request Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cardinality</strong></td>
</tr>
<tr>
<td><strong>Value</strong></td>
</tr>
<tr>
<td><strong>Attributes</strong></td>
</tr>
<tr>
<td><strong>Child elements</strong></td>
</tr>
</tbody>
</table>

Example 1 shows an example of a Request element.

**Example 1 Request Element**

```xml
<Request
  xmlns="http://docs.oasis-open.org/wsrmi/2004/06/wsrmi-1.1.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:mustUnderstand="1">
  <MessageId groupId="mid://20040202.103832@wsr-sender.org">
    <SequenceNum number="0" groupExpiryTime="2005-02-02T03:00:33-31:00" />
  </MessageId>
  <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
  <ReplyPattern>
    <Value>Response</Value>
  </ReplyPattern>
</Request>
```
4.2.1 Element: Request/MessageId

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

This element includes the following attribute:

- a `groupId` attribute

<table>
<thead>
<tr>
<th>Table 5 MessageId Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cardinality</strong></td>
</tr>
<tr>
<td><strong>Value</strong></td>
</tr>
<tr>
<td><strong>Attributes</strong></td>
</tr>
<tr>
<td><strong>Child elements</strong></td>
</tr>
</tbody>
</table>

4.2.1.1 Attribute: Request/MessageId/@groupId

The RMP MUST include this attribute in the MessageId element. This attribute identifies a sequence of messages, where each sequence is of length 1 or more. The Sending RMP MUST use a distinct globally unique `groupId` value for any each distinct group of messages. Within any such group, each of all messages will have the same value for 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].

4.2.1.2 Element: Request/MessageId/SequenceNum

The Sending RMP 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 to guarantee 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 3.2 apply.

This element includes the following attributes:

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

If the MessageOrder element appears in the message received, the Receiving RMP MUST NOT deliver the message until all messages with the same groupId value and a lower number value have been delivered. Example 2 illustrates the SequenceNum element with some message fragments with SequenceNum element:

Example 2 SequenceNum Element

1) First message

```xml
<MessageId groupId="mid://20040202.103832@wsr-sender.org">
  <SequenceNum number="0"
    groupExpiryTime="2005-02-02T03:00:33-31:00" />
</MessageId>
```

2) Second message

```xml
<MessageId groupId="mid://20040202.103832@wsr-sender.org">
  <SequenceNum number="1"
    groupExpiryTime="2005-02-02T03:00:33-31:00" />
</MessageId>
```

3) The last message for the group

```xml
<MessageId groupId="mid://20040202.103832@wsr-sender.org">
  <SequenceNum number="2"
    groupExpiryTime="2005-02-02T03:00:33-31:00" last="true" />
</MessageId>
```

<table>
<thead>
<tr>
<th>Table 6 SequenceNum Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

4.2.1.2.1 Attribute: Request/MessageId/SequenceNum/@groupExpiryTime

A Sending RMP MAY include this attribute when @groupMaxIdleDuration 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. Constraints on allowed values for this attribute are specified in section 5.

4.2.1.2.2 Attribute:
Request/MessageId/SequenceNum/@groupMaxIdleDuration

A Sending RMP MAY include this attribute when @groupExpiryTime is not present. This attribute is used to specify the maximum idle time. On the Receiving RMP, if the time interval since the last message was received exceeds @groupMaxIdleDuration, then the sequence group may be terminated. On the Sending RMP, the same condition applies to the time since the last message was sent. The @groupMaxIdleDuration MUST conform to a [XML Schema] duration. Constraints on allowed values for this attribute are specified in section 5.

4.2.1.2.3 Attribute: Request/MessageId/SequenceNum/@number

Two messages with the same groupId, MUST NOT use the same sequence number value.

The @number MUST have a value between 0 and 18446744073709551615 (maximum value for [XMLschema] unsignedlong) and MUST conform to [XMLSchema] unsignedLong. The first message of a group MUST have value 0. The value is incremented by 1 for each message submitted to the Sending RMP for this group. Once the value reaches the maximum the group is terminated (See Section 5).

4.2.1.2.4 Attribute: Request/MessageId/SequenceNum/@last

This attribute is used to mark the end of a group, when its last message is known from the Sending RMP before the message is sent. When this attribute is present, its boolean value has the following meaning:

• false: Indicating the message is not the last message of the group, or is not known to be the last message of the group.

• true: Indicating the message is known to be the last message sent within a group of messages.

When this attribute is not present, its value defaults to false.

4.2.2 Element: Request/ExpiryTime

The ExpiryTime element is used to indicate the ultimate date and time after which the Receiving 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 Sending RMP before being successfully sent, a Sending RMP MUST NOT send it or resend it, and MUST communicate a delivery failure to the Producer. The time MUST be expressed as UTC and MUST conform to a [XML Schema] dateTime. The message is considered expired if the current time, expressed in UTC, is greater than the value of the ExpiryTime element.

Note: Given the above definition of ExpiryTime, when a received message requiring Duplicate Elimination is processed, it is sufficient to only check only for its duplicates among MessageIds of past messages that have not yet expired yet at the time of the duplicate check.

Table 7 ExpiryTime Element
4.2.3 Element: Request/ReplyPattern

An RMP MUST include the ReplyPattern element in a Request element. The ReplyPattern element includes the following child elements:

- a Value element
- a ReplyTo element

<table>
<thead>
<tr>
<th>Table 8 ReplyPattern Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

4.2.3.1 Element: Request/ReplyPattern/Value

The Value element is used by a Sending RMP to indicate which reply pattern it is requesting. An RMP MUST include the Value element in a ReplyPattern element. This element is used to specify whether the Acknowledgment Indication (or RM Fault Indication) should be sent back directly in the response 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
- Callback
- Poll

These values respectively indicate that the RM-Reply Patterns - Response, Callback, and Poll - are in use, as described in Section 2.5.

| Table 9 Value Element |
### 4.2.3.2 Element: Request/ReplyPattern/ReplyTo

A Sending RMP MUST include this element for a message with a “Callback” value for the Value element. The Sending RMP MUST NOT include this element for a message with a “Response” or “Poll” value for the Value element. This element is used to specify the endpoint for where the initial Sending RMP expects to receive a callback Acknowledgment Indication or RM Fault Indication.

If present, the format of ReplyTo element MUST be specified by the reference-schema attribute. If the attribute is omitted, the default format of ReplyTo element is URI as defined in [RFC 2396].

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String: Response, Callback, or Poll</td>
</tr>
<tr>
<td>Attributes</td>
<td>None</td>
</tr>
<tr>
<td>Child elements</td>
<td>None</td>
</tr>
</tbody>
</table>

### 4.2.3.2.1 Attribute: Request/ReplyPattern/ReplyTo/@reference-schema

This attribute is used to specify the format or schema of the value of the ReplyTo element. The Sending RMP MAY omit this attribute when the value of the ReplyTo element is expressed with a value of type URI.

### 4.2.4 Element: Request/AckRequested

A Sending RMP MUST include the AckRequested element when the GuaranteedDelivery or OrderedDelivery Agreement items are enabled. This element is used by a Sending RMP to request the Receiving RMP to publish an Acknowledgment after the message is delivered to the consumer party or else to publish an RM Fault Indication. This publishing MUST be done even for received messages that are duplicates of previously delivered messages. For example, if the RM-Reply Pattern is Callback, an Acknowledgment Indication MUST be sent back.

The Receiving RMP MAY publish an RM Fault Indication for a Reliable Message, even if the AckRequested element is not present in the Request element for that message.

The pattern used to send the Acknowledgment or RM Fault Indication is based on the value of the ReplyPattern element.

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
</tr>
<tr>
<td>Attributes</td>
<td>reference-schema</td>
</tr>
<tr>
<td>Child elements</td>
<td>None</td>
</tr>
</tbody>
</table>
4.2.5 Element: Request/DuplicateElimination

A Sending RMP MUST include the DuplicateElimination element when the NoDuplicateDelivery Agreement item is enabled. (Refer to Section 3.2 for details.)

<table>
<thead>
<tr>
<th>Table 12 DuplicateElimination Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

4.2.6 Element: Request/MessageOrder

A Sending RMP MUST include the MessageOrder element when the OrderedDelivery Agreement item is enabled. When this element is present, the AckRequested element and DuplicateElimination element MUST also be present.

<table>
<thead>
<tr>
<th>Table 13 MessageOrder Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

Example

The HTTP message below uses the Request reliability element, which specifies (among other things) that all three reliability features should be used: GuaranteedDelivery ("AckRequested" element), NoDuplicateDelivery ("DuplicateElimination" element), and Guaranteed Message Ordering ("MessageOrder" element). The reply pattern is “Poll”, meaning that no Acknowledgment or Fault will be sent back unless explicitly requested by another message containing a PollRequest header.

Example 3 Reliable Message with Request header
4.3 PollRequest Element

A Sending RMP MUST include a PollRequest element when the ReplyPattern agreement item has the value "Poll". However, PollRequest messages can also be used to obtain delivery status for messages that were originally sent with “Response” or “Callback” ReplyPattern elements. If a Receiving RMP does not support the use of PollRequest as a general status query mechanism, it MAY return a NonSupportedFeature fault. The response to a PollRequest message MUST contain within a Response header element RM-Reply information relevant to non-expired messages, within a Response header element MUST be contained in the response of the PollRequest message.
This element includes the following attribute and child elements:

- SOAP `mustUnderstand` attribute, as specified in Appendix A
- a `ReplyTo` element
- a `RefToMessageIds` element

### Table 14 PollRequest Element

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 or 1</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Child elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>soap:mustUnderstand (boolean)</td>
<td>ReplyTo, RefToMessageIds</td>
</tr>
</tbody>
</table>

### Example 4 PollRequest Element

```xml
<PollRequest
   xmlns="http://docs.oasis-open.org/wsrmi/2004/06/wsrmi-1.1.xsd"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   soap:mustUnderstand="1">
   <RefToMessageIds groupId="mid://20040202.103832@wsr-sender.org">
       <SequenceNumRange from="0" to="5"/>
       <SequenceNumRange from="15" to="20"/>
   </RefToMessageIds>
   <RefToMessageIds groupId="mid://20040202.103811@wsr-sender.org"/>
   <RefToMessageIds groupId="mid://20040202.103807@wsr-sender.org">
       <SequenceNumRange from="713" to="6150"/>
   </RefToMessageIds>
</PollRequest>
```

### 4.3.1 Element: PollRequest/ReplyTo

A Sending RMP MAY include this element. If present, the Receiving RMP MUST send the RM-Reply information in a new request to the endpoint specified by this element. If not present, the Receiving RMP may send back the RM-Reply on the response of to the Poll request itself. The format or schema of the value of this element is specified by the `reference-schema` attribute. If the attribute is omitted, the default format of ReplyTo element is URI as defined in [RFC 2396].

### Table 15 ReplyTo Element
4.3.1.1 Attribute: PollRequest/ReplyTo/@reference-schema

This attribute is used to specify the format or schema of the value of the ReplyTo element. The Sending RMP MAY omit this attribute, when in which case, the value of the ReplyTo element is expressed with a value of type URI.

4.3.2 Element: PollRequest/RefToMessageIds

A Sending RMP MUST include the RefToMessageIds element for a PollRequest message. This element contains the identifiers of groups and messages queried for whose status the Sending RMP is requesting. This element MUST have one @groupId and MAY contain zero or more SequenceNumRange element as follows:

- a groupId attribute
- zero or more SequenceNumRange elements

<table>
<thead>
<tr>
<th>Table 16 RefToMessageIds Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

When this RefToMessageIds element has a @groupId but doesn't have SequenceNumRange element, the Receiving RMP MUST send back RM- Replies for non-expired messages that were either delivered or faulted in that group.

When the RefToMessageIds element has a @groupId and SequenceNumRange element(s), the Receiving RMP MUST return RM- Replies for the non-expired messages, specified by the combination of groupId of RefToMessageIds and SequenceNumRange element(s), for the non-expired messages that were either delivered or faulted.

When the Sending RMP requests multiple RM- Replies with different groupId values in one PollRequest Message, it MUST include a RefToMessageIds element for each groupId value.

4.3.2.1 attribute: PollRequest/RefToMessageIds/@groupId

The RefToMessageIds element MUST include a @groupId. The @groupId specifies the group of messages to be queried for whose status the Sending RMP is requesting. The syntax of this attribute is URI, as defined in [RFC2396].
4.3.2.2 element: PollRequest/RefToMessageIds/SequenceNumRange

The Sending RMP MUST include the SequenceNumRange element when it specifies which messages in a group are queried for status. Attributes @from and @to of this element express a range for SequenceNum values. This element MUST contain the following two attributes:

- a from attribute
- a to attribute

<table>
<thead>
<tr>
<th>Table 17 SequenceNumRange Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

4.3.2.2.1 attribute: PollRequest/RefToMessageIds/SequenceNumRange/@from

This attribute specifies the lowest SequenceNum/@number value of the message range. The value of @from is of type unsignedLong, and MUST be equal to or smaller than the value of @to.

4.3.2.2.2 attribute: PollRequest/RefToMessageIds/SequenceNumRange/@to

This attribute specifies the highest SequenceNum/@number value of the message range. The value of @to is of type unsignedLong, and MUST be equal to or larger than the value of @from.

When the range is limited to a single message, @from and @to MUST have same value.

Example

The HTTP message below uses the PollRequest reliability element, asking the Receiving RMP 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 5 PollRequest Message embedded in HTTP Request

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

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <PollRequest
      xmlns="http://docs.oasis-open.org/wsrmm/2004/06/ws-reliability-1.1.xsd"
```
4.4 Response Element

Indicating Acknowledgments and Faults for Reliable Messages MUST be done by using the Response element. This element includes the following attributes:

- SOAP mustUnderstand attribute, as specified in Appendix A
- a replyPattern attribute

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 uses the Callback RM-Reply Pattern, and the reply and the new request share a common destination URI, a Response element can be bundled with a Request element, enabling the combination of an Acknowledgment Indication with the business response to the original message. This also allows a Receiving RMP to bundle an Acknowledgment Indication with another unrelated message to the Sending RMP (e.g., to reduce network traffic).

**Table 18 Response Element**

<table>
<thead>
<tr>
<th>Cardinality</th>
<th>0 or 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>None</td>
</tr>
</tbody>
</table>
| Attributes  | soap:mustUnderstand (boolean)  
|             | replyPattern (string) |
| Child elements | NonSequenceReply  
|              | SequenceReplies |

Example 6 shows an instance of the Response element.

**Example 6 Response Element**

```xml
<Response
  xmlns="http://docs.oasis-open.org/wsrml/2004/06/wsrml-1.1.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
>
4.4.1 attribute: Response/@replyPattern

The Response element MUST include the @replyPattern. If the response is being returned as a result of a message with the "Poll" RM-Reply Pattern, this attribute must have the value "Poll".

If the response is being returned as a result of a "Callback" RM-Reply Pattern, this attribute must have the value "Callback".

If the response is being returned as a result of a "Response" RM-Reply Pattern, this attribute must have the value "Response". In this case, the following restrictions apply:

- If the group is made of a single message without a sequence number, the first element of the response must be a NonSequenceReply element containing its @groupId which is the globally unique message identifier for the Reliable Messaging equal to that of the 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 both equal to the sequence number in the request.

4.4.2 Element: Response/NonSequenceReply

An Acknowledgment or a Reliable Messaging Fault indication for a message which does not have a sequence number MUST include a NonSequenceReply element.

This element MUST contain the @groupId for the message referred to. If the reply is an acknowledgment of delivery, the element MUST NOT include @fault. If the reply is an indication of a Reliable Messaging Fault, the element MUST include @fault.

<table>
<thead>
<tr>
<th>Table 19 NonSequenceReply Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>
4.4.2.1 attribute: Response/NonSequenceReply/@groupId

This attribute specifies the groupId of a message which did not have a sequence number. The value is of type URI, as defined in [RFC2396]. A NonSequenceReply element MUST include the message’s @groupId.

4.4.2.2 attribute: Response/NonSequenceReply/@fault

This attribute indicates the code of a Reliable Messaging Fault which was encountered while processing the message. The Cardinality of this attribute is 0 or 1.

4.4.3 Element: Response/SequenceReplies

A Receiving RMP MUST include the SequenceReplies element to respond with the status of messages which had a SequenceNum element. This element MUST contain a @groupId attribute, and 0 or more ReplyRange elements.

<table>
<thead>
<tr>
<th>Table 20 SequenceReplies Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Child elements</td>
</tr>
</tbody>
</table>

4.4.3.1 attribute: Response/SequenceReplies/@groupId

This @groupId specifies the group of message(s) for which its SequenceReplies element carries the status. The value is of type URI, as defined in [RFC2396].

4.4.3.2 Element: Response/SequenceReplies/ReplyRange

The ReplyRange element indicates a range of sequence numbers. The messages referred to either are acknowledged (in which case @fault MUST NOT be present) or have encountered a particular, common fault condition (in which case @fault MUST be present).

<table>
<thead>
<tr>
<th>Table 21 ReplyRange Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
</tr>
<tr>
<td>Value</td>
</tr>
</tbody>
</table>
| Attributes | from (unsigned Long)  
to (unsigned Long)  
fault (QName) |
| Child elements | None |
4.4.3.2.1 attribute: Response/SequenceReplies/ReplyRange/@from
This attribute has same type and semantics as in the PollRequest element.

4.4.3.2.2 attribute: Response/SequenceReplies/ReplyRange/@to
This attribute has same type and semantics as in the PollRequest element.

4.4.3.2.3 attribute: Response/SequenceReplies/ReplyRange/@fault
This attribute indicates the code of a Reliable Messaging Fault code which was encountered while processing all of the messages indicated by sequence numbers in the @from - @to range. The Receiving RMP MUST NOT include this attribute for a ReplyRange element used for Acknowledgments. The Cardinality of this attribute is 0 or 1.

Example
The message below uses the Response reliability element, which in this case is carrying the response of a previous PollRequest element. The response acknowledges a message specified by the groupId "mid://20040202.103811@wsr-sender.org", and messages for a group specified by the groupId "mid://20040202.103832@wsr-sender.org" within the ranges of sequence numbers 0 to 14 and 16 to 20. Also, the response is reporting an RM Fault for a message with sequence number 15 for the group.

Example 7 RM-Reply 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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Response
      xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1" replyPattern="Poll">
      <NonSequenceReply groupId="mid://20040202.103811@wsr-sender.org"/>
      <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">
        <ReplyRange from="0" to="14"/>
        <ReplyRange from="15" to="15" fault="InvalidRequest"/>
        <ReplyRange from="16" to="20"/>
      </SequenceReplies>
    </Response>
  </soap:Header>
</soap:Envelope>
The protocol defines two fault categories:

- The Message Format fault set, which includes all faults generated because of a malformed Reliable Message header.
- The Message Processing fault set, which includes all faults generated while processing the message.

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. Reliable Message Faults are carried in the SOAP Header, and do not rely exclusively on the SOAP Fault model for the following reasons:

- The SOAP Fault model does not allow batching several faults in the same message.
- RM Faults may be carried by business messages that are not concerned with these faults, and for this reason they should not affect the SOAP body of these messages.

The rules for processing faults are:

- A message for which an RM Fault is published MUST NOT be delivered by the Receiving RMP; and therefore MUST NOT be acknowledged.
- When the Response RM-Reply Pattern is in use and the message cannot be delivered to the Consumer, the response of the SOAP MEP instance MUST contain a SOAP Fault (in the SOAP Body) in addition to the appropriate RM Fault (in the SOAP Header). If the RM Fault encountered was due to a problem with the request header element, and a Consumer response payload is expected, a SOAP client fault MUST be returned. If the RM Fault encountered was due to a problem with processing by the Receiving RMP, and a Consumer response payload is expected, a SOAP server fault MUST be returned. The Sending RMP and Producer expect either a complete response or a SOAP Fault when using the Response RM-Reply Pattern; and this requirement satisfies those expectations. More details are given in the HTTP Binding section.
- When a Callback or Poll RM-Reply Pattern is in use and the message cannot be delivered to the Consumer due to a failure in processing the RM headers, then no SOAP Fault shall be returned. The HTTP binding section gives more details on the recommended behavior in such case.

### 4.5.1 Message Format Faults

The following Fault codes may be carried in a Response element as the value of @fault. These faults are sent by the Receiving RMP when the message format of the Reliable Messaging Headers are invalid or wrong.

<table>
<thead>
<tr>
<th>Local part name</th>
<th>Description and Cause(s)</th>
</tr>
</thead>
</table>

Table 22: Invalid Message Format Fault Code Values
| 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. The soap:mustUnderstand attribute is missing |
| InvalidPollRequest | This fault is sent when the PollRequest element is wrong or invalid. Examples are: |
|                | 1. The soap:mustUnderstand attribute is missing |
| InvalidMessageId | This fault is sent in any of the following cases: |
|                | 1. If @groupId (for MessageId or RefToMessageIds) doesn’t exist; or if it does exist, and the value is wrong or invalid. |
|                | 2. If number attribute in SequenceNum element doesn’t exist; or if it does exist, the value is invalid or wrong. |
|                | 3. The aAttributes (from and to) of SequenceNumRange doesn’t exist; or if they exists, the values are invalid or wrong. |
| InvalidMessageParameters | This fault is sent for any of these cases: |
|                | 1. The groupExpiryTime is wrong or invalid. |
|                | 2. The 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 @last attribute of the SequenceNum element exists and is not one of the allowed {False|True} values. |
| InvalidReplyPattern | This fault is sent if the ReplyPattern format is wrong or invalid, or when the ReplyTo element is missing for the Callback pattern. |
| InvalidExpiryTime | This fault is sent if the ExpiryTime format is wrong or invalid. |
4.5.2 Message Processing Faults

These faults are sent by the Receiving RMP when there is an error processing a valid Reliable Messaging message.

### Table 23 Messaging Processing Failure Fault Code Values

<table>
<thead>
<tr>
<th>Local part name</th>
<th>Description and Cause(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NonSupportedFeature</td>
<td>This fault is sent by the Receiving RMP when it receives a message with an RM feature that it doesn’t support. An example is an RM message with a MessageOrder element to a Receiving RMP that doesn’t support Guaranteed Message Ordering.</td>
</tr>
<tr>
<td>PermanentProcessingFailure</td>
<td>This fault is sent for permanent/fatal processing failures such as:</td>
</tr>
<tr>
<td></td>
<td>1. Persistence Storage failures</td>
</tr>
<tr>
<td></td>
<td>2. Message Delivery failures</td>
</tr>
<tr>
<td></td>
<td>A PermanentProcessingFailure fault indicates that the failure is fatal, and subsequent retries of the same message will also will fail.</td>
</tr>
<tr>
<td>MessageProcessingFailure</td>
<td>This fault is sent for transient failures such as:</td>
</tr>
<tr>
<td></td>
<td>1. Maximum-The number of buffered requests exceeded the maximum limit.</td>
</tr>
<tr>
<td></td>
<td>2. Maximum-The number of threads reached the maximum limit, etc.</td>
</tr>
<tr>
<td></td>
<td>A transient fault, unlike a permanent fault, is a temporary, one and the message MAY succeed after a in subsequent retries.</td>
</tr>
<tr>
<td>GroupAborted</td>
<td>All processing for the groupID value associated with the reliable message request has been aborted by the Receiving RMP. No subsequent messages within that group will be delivered by the Receiving RMP.</td>
</tr>
</tbody>
</table>
Example 8  RM Fault Indication for Reliable Messaging

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Response
      xmlns="http://docs.oasis-open.org/wsrmi/2004/06/wsrmi-1.1.xsd"
      soap:mustUnderstand="1" replyPattern="Callback">
      <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">
        <ReplyRange from="1" to="1" fault="InvalidRequest" />
      </SequenceReplies>
    </Response>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

If the PollRequest element in Example 4 were missing the soap:mustUnderstand attribute, the InvalidPollRequest fault may be sent as follows.

Example 9  RM Fault Indication for PollRequest message

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Response
      xmlns="http://docs.oasis-open.org/wsrmi/2004/06/wsrmi-1.1.xsd"
      soap:mustUnderstand="1" replyPattern="Poll">
      <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">
        <ReplyRange from="0" to="5" fault="InvalidPollRequest" />
        <ReplyRange from="15" to="20" fault="InvalidPollRequest" />
      </SequenceReplies>
      <NonSequenceReply groupId="mid://20040202.103811@wsr-sender.org"
        fault="InvalidPollRequest" />
      <SequenceReplies groupId="mid://20040202.103811@wsr-sender.org"/>
      <NonSequenceReply groupId="mid://20040202.103807@wsr-sender.org"
        fault="InvalidPollRequest" />
      <SequenceReplies groupId="mid://20040202.103807@wsr-sender.org"/>
      <NonSequenceReply groupId="mid://20040202.103807@wsr-sender.org"
        fault="InvalidPollRequest" />
    </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, and its persistent resources reclaimed, is essential for keeping the resource footprint of reliability low. However, this section is not just about efficient management of resources; it describes normative behavioral rules for RMPs when handling group termination.

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

The states of a group that are part of the termination process on the Sending RMP and the Receiving RMP are defined as follows:

Group complete:

- A group is considered complete in the Sending RMP, when all of its messages have been sent and the last sent message has an ending marker (SequenceNum/@last="true", or it has a sequence number with maximum value).
  Note that completeness occurs even if not all of the group's messages have been either acknowledged or faulted (in case GuaranteedDelivery is enabled.)

- A group is considered complete in the Receiving RMP, when a last message with an ending marker has been received, and all previous messages for this group also have been received, (no number missing in the sequence), although not necessarily delivered yet.

Group closed:

- When a group is closed in the Sending RMP, no new message is expected to be sent by the RMP for this group. However, messages MAY still be resent in GuaranteedDelivery is enabled. If a new message is submitted for a closed group, the Sending RMP MUST notify the Producer that the group is closed and MUST NOT send the message.

- When a group is closed in the Receiving RMP, no new message is expected to be received for this group anymore. After a group is closed, and before the group is "removed" (see definition), a Receiving RMP MUST NOT deliver messages received with this group ID, whether or not they are duplicates of previous messages, and regardless of whether they result from a resending of previously failed messages initiated before closing on the Sending RMP (in GuaranteedDelivery is enabled).

Note: Due to timeout, a group may be closed without being complete, due to timeout. Once complete, a group will close (see termination rules).

Group Removed:
Group removal occurs at the time the group is closed, or after. Intuitively, a group is removed when a Receiving RMP does not need to remember anything about this group, i.e., when there is no need to check for duplicates of its messages in the future. This is the case when all its messages have expired.

- When a group is removed in the Sending RMP, the RMP is not required to verify that future submitted messages that are submitted are not associated with the removed group, and MAY treat these as belonging to a new group. However, when in case the Sending RMP is in charge of generating group IDs, it MUST NOT reuse the group ID of a removed group when initiating a new group.

- When a group is removed in the Receiving RMP, the RMP is no longer supposed to remember anything about this group. In particular, the group ID is discarded from the RMP state. When receiving a message with same group ID as a removed group, a Receiving RMP is not required to verify if this group ID value has already been used. Such a message MAY be treated as belonging to a new group.

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 Group Termination parameters that control the persistence of the group data. The corresponding message header attributes are groupExpiryTime and groupMaxIdleDuration respectively. The following requirements pertain to these header attributes:

a) The first message in a group (the one with Request/MessageId/SequenceNum/@number=0) MUST be used by the Sending RMP to indicate which timeout parameters are in use for the group.

- If the first message in the sequence of a group has neither group timeout parameter present, the group will be terminated according to condition t3, t4 or t5.

- If the first message has either one of the two group timeout parameters present (either @groupExpiryTime, or @groupMaxIdleDuration), then the group will be subject to termination rules t1 or t2 described below.

- A fault MUST be returned if both group persistence parameters are present in any request message. An InvalidMessageParameters fault shall be sent in this case.

- If @groupExpiryTime is in use, the Sending RMP MUST NOT send a message in that group with an ExpiryTime value greater than @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 Sending RMP MAY modify the value by sending a subsequent message with a new value. When applying termination rules, the Sending RMP MUST use the value in the message with the highest sequence number sent for the group. The Receiving RMP MUST use the value from the message with the highest sequence number received for the group.

d) 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 the max (ExpriyTime) of the 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 the messages received for the group.
5.1.3 Termination Rules

Termination is the process by which an RMP discontinues the use of a group, allowing the RMP to reclaim resources used by the group. Termination typically involves two steps that may occur at different times: closing and removal. Removal of a group may happen some time later after it is closed, so that it will be possible to filter out potential duplicate messages. The general rule is that a group is removed once all of its messages have expired. If we define \( \text{max(Exp}_\text{iryTime)} \) as the maximum date of all \( \text{Exp}_\text{iryTime} \) values of the messages sent for a group (on the Sender side) or received for a group (on the Receiver side), then a group will not be removed before \( \text{max(Exp}_\text{iryTime)} \) occurs.

As a summary, there are two general indicators an RMP will use to terminate a group:

(a) Message marker: Information within a message (either ending marker, \text{RequestId}/\text{MessageId}/\text{SequenceNum}/@last="true", or the maximum sequence number) that indicates the last message for the group. This is used by termination rules T3, T4.

(b) Timing: Either the group lifespan expired, or its idle time exceeds a timeout. This is used by termination rules T1, T2. Or, due to message expiration, a group with the ordering requirement cannot be delivered. This is used by termination rule T5.

These termination rules apply to both ordered and unordered groups. However, these rules do NOT apply to groups which contain a single message with no sequence number.

5.1.3.1 Termination by expiration (T1):

Context:

The group had specified \textit{@groupExp}_\textit{iryTime}-specified.

Receiver side:

Triggering event: \textit{@groupExp}_\textit{iryTime} is over.

The RMP MUST close and remove the group.

Sender side:

Triggering event: \textit{@groupExp}_\textit{iryTime} is over. (Note that in that case, \text{max(Exp}_\text{iryTime)} is also \text{is over}.)

The RMP MUST close and remove the group.

5.1.3.2 Termination by idle timeout (T2):

Context:

The group specified had \textit{@groupMax}_\textit{IdleDuration}-specified.

Receiver side:

Triggering event: \textit{The time since the last received message for the group is over} \textit{@groupMax}_\textit{IdleDuration}.

The group MUST be closed. But unlike (T1), some of its past messages may not have expired yet. \textit{In case Duplicate Elimination is required}, in order to make sure all potential duplicates for the group will not be delivered, the group MUST NOT be removed until \text{max(Exp}_\text{iryTime)} is reached.

Sender side:
The time since the last sent message for the group is over \( @\text{groupMaxIdleDuration} \). The group MUST be closed. If Guaranteed Delivery was required, the group MUST be removed once all sent messages either have either been acknowledged, or their delivery failure notified. If no Guaranteed Delivery was required, the group MUST be removed immediately.

### 5.1.3.3 Termination by completeness (T3):

#### Context:
No specific context.

#### Receiver side:
Triggers event: The RMP receives a message marked last (Request/MessageId/SequenceNum/@last="true"), which closes the group, assuming that all previous messages for the group have been received. Or, assuming in case that the message with ending marker has already been received, the RMP receives the last missing message in the group.

The group MUST be closed. However, its removal is done according to (T1) or (T2), depending upon which timeout parameter was specified for the group. If no timeout parameter was specified, the group is removed once all its messages have expired, i.e., the date max(ExpiryTime) has passed.

#### Sender side:
Triggers event: The RMP sends a message marked last.

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

#### Note:
In the case in which a message is received with an ending marker, but not before all previous messages have been received, the group remains active. No termination process is initiated yet.

### 5.1.3.4 Termination by sequence exhaustion (T4):

#### Context:
No specific context.

#### Receiver side:
Triggers event: The RMP receives a message with a sequence number of maximum value and assuming that all previous messages for the group have been received. Or, assuming that the message with maximum sequence number has already been received, the RMP receives the last missing message in the group.

The group closing and removal follows the rules in T3, the message with maximum sequence number acting as a message with the ending mark.

#### Sender side:
Triggers event: The RMP sends a message with a sequence number with maximum value.

The group closing and removal follows the rules in T3, the message with maximum sequence number acting as a message with the ending mark.
Note: In case a message is received with the maximum sequence number, but not before all previous messages have been received, then the group remains active. No termination process is initiated yet.

5.1.3.5 Termination by ordering failure (T5):

Context:

The group requires under the Guaranteed Message Ordering reliability feature requirement.

Receiving side:

Triggering event: in an ordered group, a received message expires before delivery.

The group MUST be closed. The group is removed according to rule T3.

Sender Side:

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

The group MUST be closed. The group is removed according to rule T3.

5.1.3.6 Summary of Group Termination Rules

Table 24  Conditions for terminating a group – Receiving RMP:

<table>
<thead>
<tr>
<th>Group Closing</th>
<th>Group Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>When ( t_{\text{GroupExpiryTime}} ) has expired.</td>
<td>(after closing) When ( t_{\text{GroupExpiryTime}} ) has expired.</td>
</tr>
<tr>
<td>When ( t_{\text{GroupMaxIdleDuration}} ) timeout has expired.</td>
<td>(after closing) When ( t_{\text{ExpiryTime}} ) has expired.</td>
</tr>
<tr>
<td>When a gGroup is complete.</td>
<td>(after closing) When ( t_{\text{ExpiryTime}} ) has expired.</td>
</tr>
<tr>
<td>When a gGroup is ordered AND a non-delivered message has expired.</td>
<td>(after closing) When ( t_{\text{ExpiryTime}} ) has expired.</td>
</tr>
</tbody>
</table>

Conditions for terminating a group in a Sending RMP:

Table 25  Conditions for terminating a group – Sending RMP:

<table>
<thead>
<tr>
<th>Group Closing</th>
<th>Group Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>When ( t_{\text{GroupExpiryTime}} ) has expired.</td>
<td>(after closing) When ( t_{\text{GroupExpiryTime}} ) has expired.</td>
</tr>
<tr>
<td>When ( t_{\text{GroupMaxIdleDuration}} ) timeout has expired.</td>
<td>(after closing) In case GuaranteedDelivery is not required, remove the group immediately. Otherwise, remove it if all messages have been either acknowledged or faulted.</td>
</tr>
</tbody>
</table>
### 5.2 Reliability of WSDL Operations

This specification supports Reliable Messaging capabilities for WSDL 1.1 [WSDL 1.1] One-way and Request-response operation types only. In addition, this specification does not address the reliability of the output message of WSDL Request-response operations. This is because the binding between WSDL operation types and RMP operation invocations (Section 2) requires the output message of a WSDL operation to map to a Respond RMP invocation, while the scope of reliability agreements is restricted to payloads passed to an RMP with Submit (Section 3).

Some restriction in the use of RM-Reply patterns apply, depending on WSDL operations.

While the input message of an operation of type Request-response can use any of the three RM-Reply Patterns to receive Acknowledgment or Fault Indications, a One-way operation SHOULD (for WS-I BP 1.0 conformance) only use either Callback or Poll RM-Reply Pattern. See Section 2 for the detail of binding constraints with SOAP MEPs. Table 26 indicates recommended usage of RM-Reply Patterns, for the WSDL operation types. At a minimum, an RMP MUST, at least, support the recommended combinations in Table 26, for the RM-Reply Patterns it supports.

<table>
<thead>
<tr>
<th>WSDL operation type</th>
<th>Response RM-Reply Pattern</th>
<th>Callback RM-Reply Pattern</th>
<th>Poll RM-Reply Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request-response</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>One-way WSDL operation type*</td>
<td>No **</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* As mentioned above, the current version of the WS-Reliability protocol does not address the reliability of WSDL response messages (the "output" messages in WSDL operations). This table entry only concerns RM-Reply patterns for WSDL requests ("input" messages).

** WS-I BP 1.0 disallows sending a SOAP envelope in an 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 Receiving RMP may determine its behavior entirely based on message header content, regardless of the WSDL definition.

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 the 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 Attachments

When this specification is used with the W3C Note "SOAP Messages with Attachments" specification [SOAP with Attachments], the following rules MUST be met:

1) The first MIME part MUST include the 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 Receiving RMP MUST deliver all MIME parts in a Reliable Message to the Consumer.
6 HTTP Binding

This section specifies two normative bindings of WS-Reliability header elements to SOAP header blocks carried using HTTP as a transport protocol:

- SOAP 1.1 over HTTP POST binding: An implementation of WS-Reliability MAY support mapping the WS-Reliability header elements as SOAP header blocks in accordance with the SOAP 1.1 HTTP Binding, as specified in Section 6 of SOAP 1.1. In that case, the Request-response SOAP MEP defined in this specification will map to an HTTP request-response. The One-way SOAP MEP, as defined in Section 2, maps to the request of an HTTP request-response.

- SOAP 1.2 over HTTP POST binding: An implementation of WS-Reliability MAY support mapping the WS-Reliability header elements as SOAP header blocks in accordance with the SOAP 1.2 HTTP binding for the Request/Response MEP, as specified in Section 7, “SOAP HTTP Binding”, of SOAP 1.2 Part 2.

If a Reliable Message request is invoked using SOAP 1.1, all subsequent message exchanges pertaining to that Message Identifier MUST use the SOAP 1.1 protocol.

If a ReplyTo element present in a Request element or Poll Request header element, sent using the SOAP 1.1 protocol, contains only a URL and uses the 'http:' URL scheme, then the WS-Reliability response MUST be sent using the HTTP binding specified in section 6 of SOAP 1.1.

If a Reliable Message request is invoked using SOAP 1.2, all subsequent message exchanges pertaining to that Message Identifier MUST use the SOAP 1.2 protocol.

If a ReplyTo element present in a Request element or Poll Request header element, sent using the SOAP 1.2 protocol, contains only a URL and uses the 'http:' URL scheme, then the WS-Reliability response MUST be sent using the HTTP binding for Request/Response MEP specified in SOAP 1.2.

The following subsections specify the mapping of WS-Reliability header elements to HTTP request and response messages, for the three RM-Reply patterns. The Poll RM-Reply Pattern has two variations, (synchronous and asynchronous).

The specific reply pattern in use is identified by the value of ReplyPattern element (See Section 4.2.3 for detail).

This specification expects that the transport layer will not deliver a corrupted message to the reliability layer. When a request message contains the AckRequested element, upon receipt of a ReliableMessage, the Receiving RMP MUST send an RM-Reply for that request. This RM-Reply MUST be either an Acknowledgment Indication or an RM Fault Indication. For the Callback and Poll reply patterns, a WS-Reliability Response element can contain multiple Acknowledgment and/or RM Fault indications.

For simplicity, the detailed examples only show the use of SOAP 1.1. However, the figures showing the mapping of WS-Reliability elements to HTTP POST request messages and HTTP response messages apply to both the SOAP 1.1 over HTTP POST binding, and the SOAP 1.2 over HTTP POST binding.

6.1 Reliable Messaging with Response RM-Reply Pattern

The Reliable Messaging Acknowledgment or RM Fault Indication MUST be sent back on the same HTTP connection with the HTTP Request that the Sending RMP initiated to send the Message. This is illustrated in Figure 8.
In case the message cannot be delivered to the Consumer due to a failure in processing the RM headers, then it is RECOMMENDED that the response be conforming to the WS-I Basic Profile 1.0. To achieve this, the SOAP Fault element MUST be returned in an HTTP response with the “500 Internal Server Error” HTTP status code (see R1126 in [WS-I BP1.0]).

Figure 8  Response RM-Reply Pattern

1) The Sending RMP initiates an HTTP connection, and sends a Message using the HTTP POST Request. Example 10 is an example of such a message.

2) The Receiving RMP sends back an Acknowledgment Indication to the Sending RMP on the same connection.

Example 10  Request Message with Response RM-Reply Pattern

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
    <Request
      xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1">
      <MessageId groupld="mid://20040202.103832@wsr-sender.org">
        <SequenceNum number="0"
          groupExpiryTime="2005-02-02T03:00:33-31:00" />
      </MessageId>
      <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
    </Request>
    <ReplyPattern>
      <Value>Response</Value>
    </ReplyPattern>
    <AckRequested/>
    <DuplicateElimination/>
    <MessageOrder/>
  </soap:Header>
```

wd - web services reliable messaging tc-ws-reliability 1.1-1.05
Copyright © OASIS Open 2003-2004. All Rights Reserved.
6.2 Reliable Messaging with Callback RM-Reply Pattern

The Reliable Messaging Acknowledgment or RM Fault Indication MUST be sent back on a different HTTP connection from the HTTP connection that the Sending RMP initiated to send the message. The direction of the HTTP connection is from the Receiving RMP to the Sending RMP. This is illustrated in Figure 9.

In case the message cannot be delivered to the Consumer due to a failure in processing the RM headers, then it is RECOMMENDED that the HTTP response be-conforming to the WS-I Basic Profile 1.0. To achieve this, a SOAP Fault MUST NOT be returned, and the HTTP response entity-body MUST be empty, with a "400 Bad Request" HTTP status code. If the RM Fault is a Message Format fault, the HTTP status code SHOULD be with a "400 Bad Request". (See...
otherwise, the RM fault is a Message Processing fault and the status code SHOULD be "500 Internal Server Error" otherwise, in case of a Message Processing fault.

Figure 9 Callback RM-Reply Pattern

1. The Sending RMP 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 HTTP message body. Example 13 is an example of this HTTP response.
3. The Acknowledgment Indication is sent with another HTTP connection from the Receiving RMP to the Sending RMP. An HTTP POST MUST be used for this operation. Example 14 is an example of this message.
4. The HTTP response to the (3) has no HTTP message body. Example 13 is an example of this HTTP Response.

Example 12 Request Message with Callback RM-Reply Pattern

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Header>
    <Request
      xmlns="http://docs.oasis-open.org/wsrmi/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1">
      <MessageId groupid="mid://20040202.103832@wsr-sender.org">
        <SequenceNum number="0" groupExpiryTime="2005-02-02T03:00:33-31:00" />
      </MessageId>
      <ExpiryTime>2004-09-07T03:01:03-03:50</ExpiryTime>
    </Request>
    <ReplyPattern>
```

Copyright © OASIS Open 2003-2004. All Rights Reserved.
<Value>Callback</Value>
<ReplyTo>http://wsr-sender.org/abc/wsrListnener</ReplyTo>
</ReplyPattern>
<AckRequested/>
<DuplicateElimination/>
<MessageOrder/>
</Request>
</soap:Header>
<soap:Body>
<Request xmlns="http://example.org/wsr">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 Indication with Callback RM-Reply Pattern
POST /abc/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.200
SOAPAction: ""
Content-Length: 1024

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Response
      xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1" replyPattern="Callback">
      <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">
        <ReplyRange from="0" to="0"/>
      </SequenceReplies>
    </Response>
  </soap:Header>
</soap:Envelope>
6.3 Reliable Messaging with Poll RM-Reply Pattern

When a PollRequest is issued, the Reliable Message Acknowledgment or RM Fault Indication MAY also be sent back on a different HTTP connection from the HTTP connection used to send the acknowledged/faulted message (synchronous) being acknowledged when the PollRequest is issued, or it MAY be sent back on a different HTTP connection (asynchronous). Whether the RM-Reply corresponding to the PollRequest MAY either be synchronous or asynchronous depending upon the presence of a ReplyTo element in the PollRequest element.

In case the message cannot be delivered to the Consumer due to a failure in processing the RM headers, then it is RECOMMENDED that the HTTP response be conforming to the WS-I Basic Profile 1.0. To achieve this, a SOAP Fault MUST NOT be returned; and the HTTP response entity-body MUST be empty with a "400 Bad Request" HTTP status code. If the RM Fault is a Message Format fault, the HTTP status code SHOULD be "400 Bad Request". See R1113 in [WS-I BP1.0]; otherwise, the RM Fault is a Message Processing Fault and the status code SHOULD be "500 Internal Server Error" otherwise, in case of a Message Processing fault.

6.3.1 Synchronous Poll RM-Reply Pattern

When the PollRequest doesn’t include the ReplyTo element, then the RM-Reply is sent back as a HTTP Response on the same HTTP connection used to send the PollRequest. This is illustrated in Figure 10.

- (1) The Sending RMP initiates a HTTP connection, and sends a Message using HTTP POST Request.
- (2) The HTTP response to the (1) has no HTTP message body. Example 13 is an example of this HTTP response.
- (3) The Sending RMP initiates a different HTTP connection, and sends a PollRequest message with HTTP POST Request. Example 15 is an example of this message. Note that the PollRequest element doesn’t have a ReplyTo element.
(4) The HTTP response to (3) includes an Acknowledgment Indication and/or a Reliable Messaging Fault. Example 16 is an example of this message.

**Example 15 PollRequest message with Synchronous Poll RM-Reply Pattern**

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <PollRequest
      xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1">
      <RefToMessageIds groupId="mid://20040202.103832@wsr-sender.org">
        <SequenceNumberRange from="0" to="20"/>
      </RefToMessageIds>
    </PollRequest>
  </soap:Header>
  <soap:Body />
</soap:Envelope>
```

**Example 16 Synchronous Acknowledgment Indication**

```xml
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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Response
      xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1" replyPattern="Poll">
      <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">
        <ReplyRange from="0" to="14"/>
      </SequenceReplies>
    </Response>
  </soap:Header>
</soap:Envelope>
```
6.3.2 Asynchronous Poll RM-Reply Pattern

When the Poll request includes the ReplyTo element, then the RM-Reply is sent back to the listener identified by the ReplyTo element as a HTTP request on a different HTTP connection to the listener identified by the ReplyTo element. This is illustrated in Figure 11.

Figure 11  Asynchronous Poll RM-Reply Pattern

1. The Sending RMP initiates an HTTP connection, and sends a Message using HTTP POST request.
2. The HTTP response to the (1) has no HTTP Message Body. Example 13 is an example of this HTTP response.
3. The Sending RMP initiates a different HTTP connection, and sends a PollRequest message with HTTP POST request. Example 17 is an example of this message. Note that the PollRequest element has a ReplyTo element.
4. The HTTP response to the (3) has no HTTP Message Body. Example 13 is an example of this HTTP response.
5. The HTTP request corresponding to the Poll Request in (3) includes an Acknowledgment Indication and/or a Reliable Messaging Fault. An HTTP POST MUST be used for this operation. Example 18 is an example of this message. This request is sent to the listener identified the ReplyTo element in the PollRequest element.
6. The HTTP response to the (5) has no HTTP Message Body. Example 13 is an example of this HTTP response.
Example 17 PollRequest message with Asynchronous Poll RM-Reply Pattern

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsrm="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
   soap:mustUnderstand="1">  
   <PollRequest xmlns="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
      soap:mustUnderstand="1">  
     <RefToMessageIds groupId="mid://20040202.103832@wsr-sender.org">  
        <SequenceNumberRange from="0" to="20"/>  
     </RefToMessageIds>  
     <ReplyTo>http://wsr-sender.org/xyz/servlet/wsrmListener</ReplyTo>  
   </PollRequest>  
</soap:Header>  
<soap:Body />
</soap:Envelope>

Example 18 Asynchronous Acknowledgment Indication

POST /xyz/servlet/wsrListener HTTP/1.0
Content-Type: text/xml; charset=utf-8
Host: 192.168.183.200
SOAPAction: ""
Content-Length: 924

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsrm="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd"
   soap:mustUnderstand="1" replyPattern="Poll">  
   <SequenceReplies groupId="mid://20040202.103832@wsr-sender.org">  
      <ReplyRange from="0" to="14"/>  
      <ReplyRange from="16" to="20"/>  
   </SequenceReplies>  
</soap:Envelope>
<soap:Response>
  <soap:Body />
</soap:Envelope>
7 Conformance

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

• It has implemented all required syntax, features and behaviors.

• 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 NOT to implement.
8 References


[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
Appendix A. Schema (Normative)

The schemas for this specification have the following URLs and are located using the filenames shown in the table:

<table>
<thead>
<tr>
<th>Schema Namespace URL</th>
<th>File name</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/ws-reliability-1.1.xsd</a></td>
<td>ws-reliability-1.1.xsd</td>
<td>wsrm</td>
</tr>
<tr>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/reference-1.1.xsd</a></td>
<td>reference-1.1.xsd</td>
<td>ref</td>
</tr>
<tr>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd</a></td>
<td>fnp-1.1.xsd</td>
<td>fnp</td>
</tr>
<tr>
<td><a href="http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd">http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd</a></td>
<td>wsrmfp-1.1.xsd</td>
<td>wsrmfp</td>
</tr>
</tbody>
</table>

The SOAP mustUnderstand attribute ([defined in](#) the same namespace as used for the soap:envelope element), MUST be present in all Reliable Messaging specified header blocks, with the following restrictions:

- For SOAP 1.1 the mustUnderstand attribute value is restricted to "1".
- For SOAP 1.2, the mustUnderstand attribute value is restricted to "1" or "true".
Appendix B. WS-Reliability Features, Properties and Compositor (Normative and Optional)

B.1 Introduction

Users of a Web Service will need to be aware of the reliability capabilities (RM capabilities) that are supported or required by the service. One practical location to advertise these capabilities is in the service description (WSDL document), which allows publishing both abstract service definitions as well as concrete protocol details (bindings). This allows clients (or including 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 the composability 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.

B.2 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 points.
B.3 WSDL Extensibility Elements

B.3.1 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 all must all be supported by the Service. A compositor element can occur as a child element of wsdl11:portType, wsdl11:operation (which itself 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 compositors, 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 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 of the children elements (representing RM capability assertions). This compositor is identified by using the URI:

  "http://docs.oasis-open.org/wsrmi/2004/06/fnp-1.1.xsd/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://docs.oasis-open.org/wsrmi/2004/06/fnp-1.1.xsd/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://docs.oasis-open.org/wsrmi/2004/06/fnp-1.1.xsd/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://docs.oasis-open.org/wsrmi/2004/06/fnp-1.1.xsd/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:

```xml
<wsdl11:definitions>
```
The compositor specified at the wsdl11:portType "myPortType" and the 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:

```xml
<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>
```

### B.3.2 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:

```xml
<fnp:feature uri="..."/>
```
B.3.3 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 (but are not required to be) 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.

B.4 WS-Reliability Feature

The WS-Reliability feature is identified by the URI "http://docs.oasis-open.org/wsrm/2004/06/wsrmp-1.1.xsd". This feature URI identifies the WS-Reliability specification. Understanding this URI implies understanding the WS-Reliability specification.

B.5 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://docs.oasis-open.org/wsrm/2004/06/wsrmp-1.1.xsd".

B.5.1 Guaranteed Delivery Property

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

B.5.2 Duplicate Elimination Property

This property is identified by the QName "wsrmfp:NoDuplicateDelivery" and corresponds to the semantics specified by the WS-Reliability duplicate elimination semantics. The type of this property is "xs:boolean".
B.5.3 Message Ordering Property

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

B.5.4 Reply Pattern Property

This property is identified by the QName "wsrmfp: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)

B.6 Other Reliability Properties

In addition to the properties defined in section III, there are WS-Reliability properties that are used on the Sender side only – which is (usually the client side, and therefore they 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.

B.6.1 Group Expiry Time

This property is identified by the QName "wsrmfp: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.

B.6.2 Group Maximum Idle Duration

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

B.6.3 Message Expiration Time

This property is identified by the QName "wsrmfp: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.

B.7 Examples

B.7.1 Example for the "all" compositor

```
<wsl11:portType name="Example-1">
  <fn:compositor uri="http://docs.oasis-open.org/wsr/2004/06/fnp-1.1.xsd/compositor/all">
    <fn:feature uri="http://docs.oasis-open.org/wsr/2004/06/wsr-fp-1.1.xsd"/>
  </fn:compositor>
</wsl11:portType>
```
In the example above, the reliability feature identified by URI "http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd" 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.

B.7.2 Example for the "choice" compositor:

...
In the example above, the reliability feature identified by URI "http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd" is required by the portType. This feature consists of three properties, from which the client must choose one.

**B.7.3 Example for the "one-or-more" compositor:**

```xml
<wSDL11:portType name="Example-3">
  <fnp:compositor uri="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd/compositor/all">
    <fnp:feature uri="http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd">
      <fnp:compositor uri="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd/compositor/one-or-more">
        <fnp:property name="wsrmfp:NoDuplicateDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrmfp:OrderedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrmfp:GuaranteedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
      </fnp:compositor>
    </fnp:feature>
  </fnp:compositor>
</wSDL11:portType>
```

**B.7.4 Example for the "zero-or-more" compositor:**

```xml
<wSDL11:portType name="Example-4">
  <fnp:compositor uri="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd/compositor/all">
    <fnp:feature uri="http://docs.oasis-open.org/wsrm/2004/06/wsrmfp-1.1.xsd">
      <fnp:compositor uri="http://docs.oasis-open.org/wsrm/2004/06/fnp-1.1.xsd/compositor/zero-or-more">
        <fnp:property name="wsrmfp:NoDuplicateDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
        <fnp:property name="wsrmfp:OrderedDelivery">
          <fnp:value>true</fnp:value>
        </fnp:property>
      </fnp:compositor>
    </fnp:feature>
  </fnp:compositor>
</wSDL11:portType>
```
<fnp:value>true</fnp:value>
</fnp:property>
</fnp:compositor>
</fnp:feature>
</fnp:compositor>
...
Appendix C. 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 Nishiyma, 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
And the following people made contributions to produce Ver 1.0 of this specification:

Appendix D. Notices

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

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

Copyright © OASIS Open 2003-2004. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.