Web Services Reliability (WS-Reliability) Ver1.0

January 8, 2003

Authors:

Colleen Evans	Sonic Software Corporation
Dave Chappell	Sonic Software Corporation
Doug Bunting	Sun Microsystems, Inc.
George Tharakan	Sun Microsystems, Inc.
Hisashi Shimamura	NEC Corporation
Jacques Durand	Fujitsu Software Corporation
Jeff Mischkinsky	Oracle Corporation
Katsutoshi Nihei	NEC Corporation
Kazunori Iwasa	Fujitsu Limited
Martin Chapman	Oracle Corporation
Masayoshi Shimamura	Fujitsu Limited
Nicholas Kassem	Sun Microsystems, Inc.
Nobuyuki Yamamoto	Hitachi Limited
Sunil Kunisetty	Oracle Corporation
Tetsuya Hashimoto	Hitachi Limited
Tom Rutt	Fujitsu Software Corporation
Yoshihide Nomura	Fujitsu Limited

such implementations, including the right to authorize others to do the same, provided however, that you only distribute the Specification subject to a license agreement that protects the Owners' interests by including the proprietary legend and terms set forth in this Copyright Notice.

Disclaimer of Warranties

THIS SPECIFICATION IS PROVIDED "AS IS" AND IS EXPERIMENTAL AND MAY CONTAIN DEFECTS OR DEFICIENCIES WHICH CANNOT OR WILL NOT BE CORRECTED BY THE OWNERS). THE OWNERS MAKE NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY OR COPYRIGHT OWNER PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS.

This document does not represent any commitment to release or implement any portion of the Specification in any product.

THIS SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS, CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. THE OWNERS MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

LIMITATION OF LIABILITY

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL THE OWNERS OR THEIR LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED

AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF THE OWNERS AND/OR LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend the Owners and their licensors from any claims based on your use of the Specification for any purposes other than those of internal evaluation, and from any claims that later versions or releases of any Specifications furnished to you are incompatible with the Specification provided to you under this license.

Restricted Rights Legend

If this Specification is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Specification and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for the non-DoD acquisitions).

Report

You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your evaluation of the Specification ("Feedback"). To the extent that you provide the Owners with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant the Owners a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the Specification and future versions, implementations, and test suites thereof.

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.

This model enables a sender (i.e., a SOAP node with reliable messaging functions for sending) to send a message to a receiver (i.e., a SOAP node with reliable messaging functions for receiving) that can accept an incoming connection. Functions to accommodate a receiver that cannot accept an incoming connection (e.g., because of a firewall) are intended for further study, and are not included in this version of the specification.

Status of this document

This specification is a draft document and may be updated, extended or replaced by other documents if necessary. It is for review and evaluation only. The authors of this specification provide this document as is and provide no warranty about the use of this document in any case. The authors welcome feedback and contributions to be considered for updates to this document in the near future.

Table of contents

1. Introduction

1.1. Purpose of WS-Reliability

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

1.1.1. Scope and Definition of Reliable Messaging

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

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

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

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

Out of the scope of this specification are:

- Application level synchronous messaging. Applications which intentionally use synchronous messaging at the application level, require knowledge of the error status immediately, rather than waiting for the messaging layer to resend the

message when an error occurs at the receiver side, and are out of scope of this specification.

- Routing. Other techniques can be used in conjunction with an implementation of this specification.
- Security. Other mechanisms can be used in conjunction with an implementation of this specification.

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

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

1.1.2. The Goal of this specification

The goal of this specification is to define:

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

1.2. Notational Conventions

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

1.3. Relation to Other Specifications

(1) W3C SOAP1.1/1.2:

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

specification could be updated and defined as a Reliable Messaging Feature to be compliant with SOAP 1.2 [SOAP1.2] when it becomes a W3C Recommendation.

(2) OASIS ebXML Message Service Specification 2.0:

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

(3) OASIS WS-Security:

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

1.4. Examples of Messages compliant with WS-Reliability

Example 1 shows WS-Reliability message elements embedded in an HTTP Request.

Example 1 WS-Reliability compliant message elements in an HTTP Request

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version=”1.1”?>
<SOAP:Envelope
 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP:Header>
    <rm:MessageHeader xmlns:rm=”http://schemas.fujitsu.com/rm”
     SOAP:mustUnderstand=”1”>
        < r m:From>requestor@anyuri.com</rm:From>
        < r m:To>responder@someuri.com</rm:To>
        < r m:Service>urn:services:ItemQuoteService</rm:Service>
        < r m:MessageId>20020907-12-34@anyuri.com</rm:MessageId>
        < r m:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
    </rm:MessageHeader>
```

```
    <rm:ReliableMessage xmlns:rm=”http://schemas.fujitsu.com/rm”
        SOAP:mustUnderstand=”1”>
          <rm:MessageType>Message</rm:MessageType>
          <rm:ReplyTo>http://server1.anyuri.com/service/</rm:ReplyTo>
          <rm:TimeToLive>2002-09-14T10:19:00</rm:TimeToLive>
          <rm:AckRequested SOAP:mustUnderstand=”1” synchronous="false" />
          <rm:DuplicateElimination/>
    </rm:ReliableMessage>
    <rm:MessageOrder xmlns:rm:=”http://schemas.fujitsu.com/rm”
          SOAP:mustUnderstand=”1”>
        <rm:GroupId  status=”Continue”>020907-45261-0450@a.com</rm:GroupId>
        <rm:SequenceNumber>12</rm:SequenceNumber>
    </rm:MessageOrder>
  </SOAP:Header>
  <SOAP:Body>
        <gip:GetItemPrice xmlns:gip="Some-URI">
            <gip:itemnumber>product12345</gip:itemnumber>
        </gip:GetItemPrice>
  </SOAP:Body>
</SOAP:Envelope>
```

Example 2 shows an Acknowledgment Message embedded in an HTTP Request to the sender.

Example 2 Acknowledgment Message embedded in HTTP Request

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version=”1.0”?>
<SOAP:Envelope
 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <SOAP:Header>
    <rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
     SOAP:mustUnderstand="1">
          <rm:From>responder@someuri.com</rm:From>
          <rm:To>requester@anyuri.com</rm:To>
          <rm:Service>urn:services:ItemFilingService</rm:Service>
          <rm:MessageId>20020907-045261-0450@someuri.com</rm:MessageId>
          <rm:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
    </rm:MessageHeader>
    </rm:RMResponse xmlns:rm=http://schemas.fujitsu.com/rm
         SOAP:mustUnderstand="1">
       <rm:MessageType>Acknowledgment</rm:MessageType>
          <rm:RefToMessageId>20020907-12-34@anyuri.com</rm:RefToMessageId>
 </rm:RMResponse>
  </SOAP:Header>
  <SOAP:Body>
 </SOAP:Body>
</SOAP:Envelope>
```

Example 3 Fault Message embedded in HTTP Request

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version="1.0"?>
<SOAP:Envelope
 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP:Header>
    <rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
     SOAP:mustUnderstand="1">
          <rm:MessageId>20020907-045261-0450@anyuri.com</rm:MessageId>
```

```
        < r m:Timestamp>2002-09-07T10:10:07</rm:Timestamp>
    </rm:MessageHeader>
    <rm:RMResponse xmlns:rm="http://schemas.fujitsu.com/rm"
        SOAP:mustUnderstand="1">
       <rm:MessageType>Fault</rm:MessageType>
          <rm:RefToMessageId>20020907-12-34@anyuri.com</rm:RefToMessageId>
    </rm:RMResponse>
  </SOAP:Header>
  <SOAP:Body>
    <SOAP:Fault>
   <faultcode>SOAP:Client</faultcode>
      <faultstring>Error in the Message Header sent from Server</faultstring>
      <detail>
          < rm:rmFault xmlns:rm="http://schemas.fujitsu.com/rm">
              <rm:faultcode>rm:InvalidMessageHeader</rm:faultcode>
          </rm:rmFault>
    </detail>
    </SOAP:Fault>
 </SOAP:Body>
</SOAP:Envelope>
```

1.5. Terminology

NOTE: The following terminology has not yet been aligned with the terminology used across other Web Services specifications. It is included here for purposes of establishing an understanding of how these terms are used in this specification.

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

Reliable Messaging Processor (RMP):
A module capable of processing and enforcing Reliable Messaging as described in this specification.

Reliable Message:

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

Acknowledgment Message:
A signal message sent by a SOAP node, to notify the initial sender of delivery of the message.

Non-Reliable Message:
A message for which the sender doesn't require any level of reliability, for example no Acknowledgment message.

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

Normal Message:
A message which is not an Acknowledgment message, and which is not a Fault message.

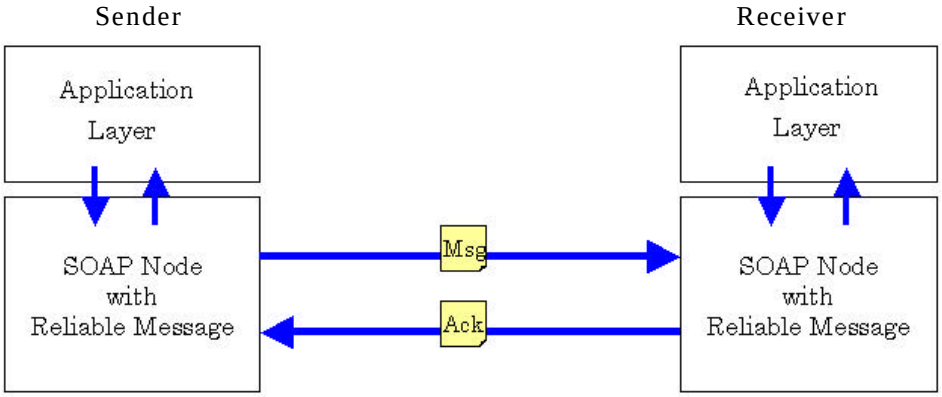
2. Messaging Model

The following section provides an overview of the WS-Reliability Messaging Model.

2.1. Overview of Messaging Model

In the Reliable Messaging Model described in this document, the sender node sends a message to the receiver node directly (i.e., intermediaries are assumed to be transparent in this specification). The receiver node sends back an Acknowledgment message to the sender node. Figure 1 shows this model.

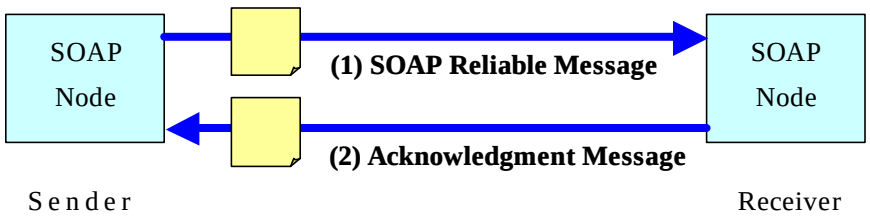
Figure 1 Messaging Model



2.2. Overview of the Acknowledgment Message

When supporting reliable messaging, upon receipt of a reliable message, the server MUST send a reply. This reply MUST be either an Acknowledge message or a Fault message. A SOAP Reliable Message is used as described in Figure 2 to guarantee message delivery. The Acknowledgment is correlated with a normal message by reference to its message ID.

Figure 2 Guaranteeing Message delivery



When the Sender sends a SOAP Reliable Message to the Receiver, the Receiver MUST send back an Acknowledgment messageor Fault message to the Sender.

2.2.1. Message ID

Every Reliable Message MUST contain a globally unique Message ID. The Acknowledgment message MUST contain a reference to the Message ID of the Acknowledged message, confirming that the receiver SOAP node has received the message.

2.2.2. Retry

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

2.2.3. Persistent Storage

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

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

The receiver is also REQUIRED to keep the received message in persistent storage to pass the message to the application layer reliably in the event a system failure or server down time occurs. Both sender and receiver MUST behave as if there was no system failure or system down after recovery. For this reason, both sender and receiver MUST use a persistent storage mechanism, e.g, HDD or equivalent nonvolatile storage.

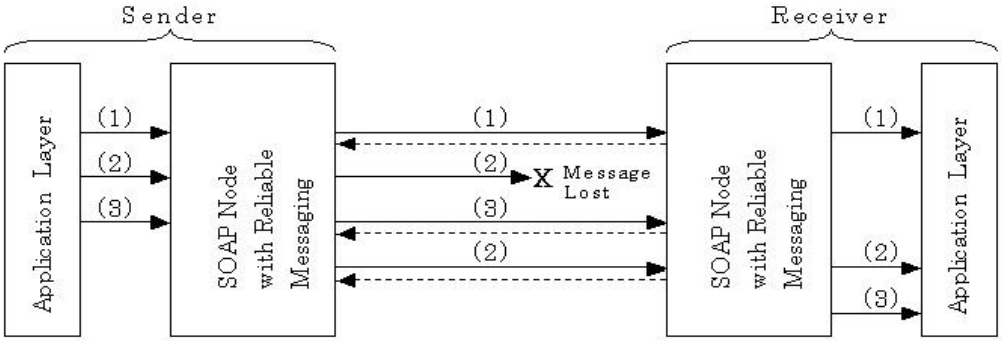
2.3. Duplicate Elimination

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

2.4. Guaranteeing Message Order

Some applications will expect to receive a sequence of messages from the same sender in the same order these messages were sent. Although there are often means to enforce this at the application layer, this is not always possible or practical. In such cases, the messaging layer is required to guarantee the Message Order. This specification defines a model described in Figure 3 to meet this requirement. (See 3.3 for more information)

Figure 3 Ordering Model



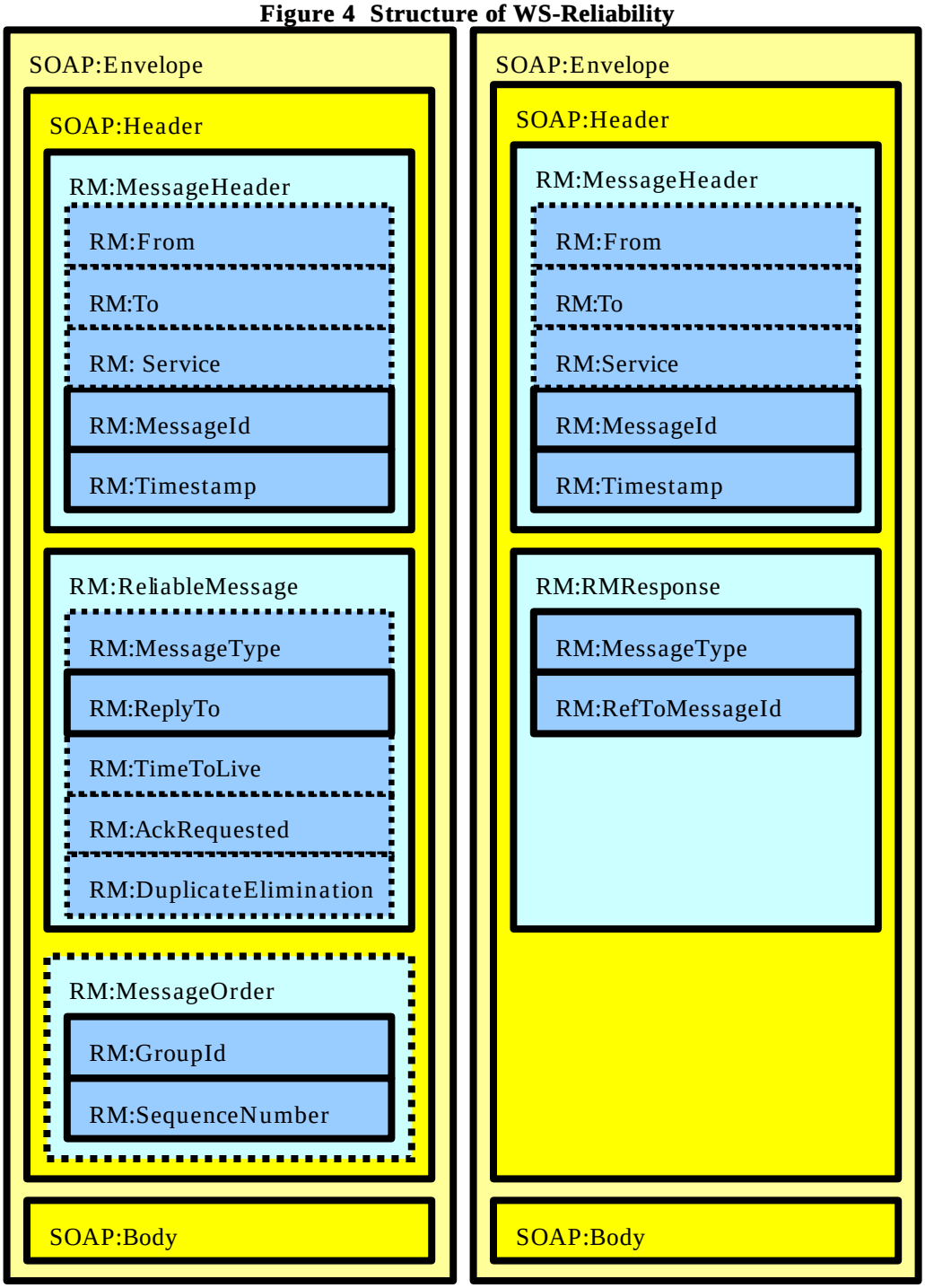
2.4.1. Sequence Number

A sequence number mechanism is used to track and enforce the order of a sequence of messages having a common grouping identifier value. Such a mechanism has been widely used in the past. For example, assume the sender application layer generates three messages in order of (1), (2), and (3). The sender SOAP node, with the message ordering function enabled, sends those messages in order of (1), (2), and (3), sequentially and asynchronously, with respective sequence numbers 1, 2, and 3. If the message (2) was not properly received for any reason, the sender will resend the (2) message after a timeout has occurred. The receiver's SOAP node will finally receive these messages as a sequence: (1), (3), and (2). The receiver SOAP node, with the message ordering function enabled, may provide the application layer with message (1), but not (3). Sequence numbering allows the receiver node to easily detect a missing message in a sequence, that is (2), as soon as receiving (3). This condition is recognized by the receiver when the sequence numbers of the messages it receives are not contiguous (e.g., 1, 3, 2). The receiver SOAP node will wait for a message with sequence number 2, and then provide message (2) and then message (3) to the application layer, in order. This behavior can be subject to variants and additional rules to deal with specific failure use cases, such as when a node cannot deliver the proper-sequence of messages due to a message being lost. (See 3.3.2 for more information)

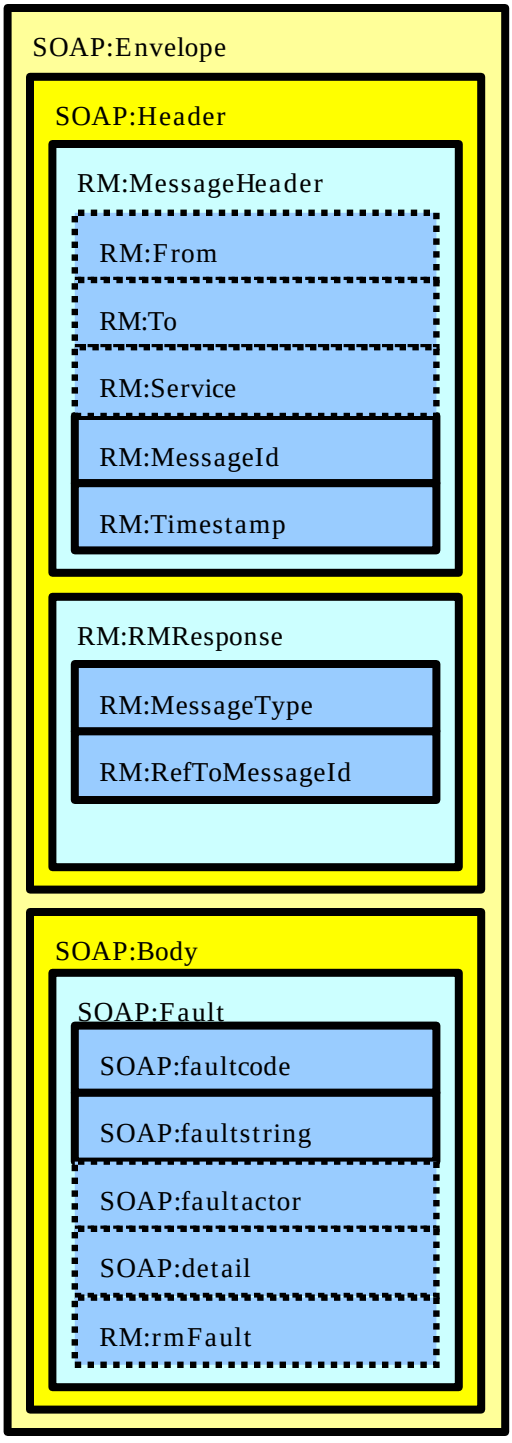
3. Message Format

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

Figure 4 Structure of WS-Reliability



(1) Reliable Message **(2) Acknowledgment Message**



(3) SOAP Fault Message

: REQUIRED element : OPTIONAL element

The namespace [XML namespaces] for Reliable Messaging defined in this specification is:

http://schemas.fujitsu.com/rm

NOTE: this namespace is not registered to any standard organization yet.

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

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

- **MessageHeader** element
- **ReliableMessage** element
- **MessageOrder** element
- **RMResponse** element

NOTE: This Reliability specification defines elements and attributes that may also be required by functions other than reliable messaging (e.g. routing, security, choreography, etc.). When using a messaging mode combining several of these functions, such parameters SHOULD NOT be duplicated across multiple SOAP headers.

This specification groups these elements and attributes into a header block, called a "MessageHeader", and under the namespace "RM". This is for the sake of completeness of this draft specification as a stand alone module. Work for future versions should consider reuse of corresponding headers and header elements, across multiple web service specifications.

This could be achieved in a number of ways. This specification could be modularized to make these elements and attributes available for any applicable use case. Alternatively, external specifications could be defined and leveraged within future versions of this specification. In either case, in order to standardize these parameters in a reusable way future versions of this specification should be coordinated with the work of other parties involved in specifying SOAP extensions relying on these parameters,.

3.1. MessageHeader Element

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

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

- **From** element
- **To** element
- **Service** element
- **MessageId** element
- **Timestamp** element

Example 4 shows an example of a MessageHeader element.

Example 4 Example of MessageHeader Element

```
<rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
 SOAP:mustUnderstand="1">
   <rm:From>requestor@anyuri.com</rm:From>
    <rm:To>responder@someuri.com</rm:To>
   <rm:Service>urn:services:ItemQuoteService</rm:Service>
   <rm:MessageId>20020907-045261-0450@anyuri.com</rm:MessageId>
   <rm:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
</rm:MessageHeader>
```

3.1.1. From Element

The OPTIONAL From element is used to specify the original sender node of the message. The value of this element MAY be a URI [RFC2396].

3.1.2. To Element

The OPTIONAL To element is used to specify the final receiver node of the message. The value of this element MAY be a URI.

3.1.3. Service Element

The OPTIONAL Service element is used to specify the service that receives and processes the message at the receiver side. The value of the Service element MAY be the name of the specific service in a business process, defined in advance. The Service element MAY contain a type attribute. If a type attribute doesn't exist, the content of the Service element MUST be a URI conforming to [RFC2396].

(1) type attribute

The type attribute is an OPTIONAL attribute. This attribute is to assist in the interpretation of the content of Service element. An example follows:

```
<rm:Service type="RetailBusinessProcess102">PurchaseOrder</rm:Service>
```

3.1.4. MessageId Element

The MessageId element is a REQUIRED element. This element MUST have a globally unique identifier as its value. The format of this identification is REQUIRED to conform to MessageId, as defined in [RFC2822].

3.1.5. Timestamp Element

The Timestamp element is a REQUIRED element. This element has a date value set to the time at which the message header was generated. The format of this value MUST conform to a [XML Schema] dateTime and MUST be expressed as UTC.

3.2. ReliableMessage Element

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

- a SOAP **mustUnderstand** attribute with a value of "1"
- **MessageType** element
- **ReplyTo** element
- **TimeToLive** element
- **AckRequested** element
- **DuplicateElimination** element

Example 5 shows an example of ReliableMessage element.

Example 5 Example of ReliableMessage element

```
<rm:ReliableMessage xmlns:rm="http://schemas.fujitsu.com/rm"
 SOAP:mustUnderstand="1">
   <rm:MessageType>Message</rm:MessageType>
    <rm:ReplyTo>http://server1.companyA.com/AckCapture</rm:ReplyTo>
   <rm:TimeToLive>2002-09-14T10:19:00</rm:TimeToLive>
      <rm:AckRequested/>
   <rm:DuplicateElimination/>
</rm:ReliableMessage>
```

3.2.1. MessageType Element

This is an OPTIONAL element. If present, the only valid value is:

- **Message** : A SOAP Request Message

3.2.2. ReplyTo Element

This is a REQUIRED element, used to specify the initial sender's endpoint to receive an asynchronous Acknowledgment message or Fault Message. The value of this element is REQUIRED to be URL as defined in [RFC 1738].

3.2.3. TimeToLive Element

This element is used to define the expiration time of the message. This is an OPTIONAL element. If the TimeToLive element is present, it MUST be used to indicate the time window that a message should be made available by a receiver node to its application layer. The time MUST be expressed as UTC and MUST conform to a [XML Schema] dateTime. The message is considered expired if the current time, in UTC, is greater than the value of the TimeToLive element. If a receiver receives an expired message, it MUST send the sender a Fault message with Error code of "InvalidTimeToLive".

3.2.4. AckRequested Element

The AckRequested element is an OPTIONAL element. It is REQUIRED for guaranteeing message delivery and message order. However this element MUST NOT appear in a non-Reliable Message. This element is to be used for a sender to request the receiver to send back an Acknowledgment message for the message sent. The AckRequested element contains the following attribute:

- a **synchronous** attribute

(1) synchronous attribute

The synchronous attribute is an OPTIONAL attribute. This attribute is used to specify whether the Acknowledgment Message should be sent back synchronously or asynchronously. This attribute, when used, MUST have one of the following two values. The default value of this attribute is "false", when omitted.

- **true** : An Acknowledgment Message MUST be sent back synchronously.
- **false** : An Acknowledgment Message MUST be sent back asynchronously.

3.2.5. DuplicateElimination Element

The DuplicateElimination element is used to require the receiver node to identify duplicate messages it has received and process them accordingly (see section 2.3). A duplicate message is a message with the same MessageID as another message. This

element is OPTIONAL. It is REQUIRED when duplicate elimination is mandated. If the MessageOrder element is present, the DuplicateElimination element MUST also be present.

3.3. MessageOrder Element

The MessageOrder element includes information to be used for Message Ordering. It is REQUIRED when message ordering is requested. When this element is used, then both the AckRequested element and the DuplicateElimination element MUST also be used. This element includes the following attribute and child elements:

- a SOAP **mustUnderstand** attribute with a value of "1"
- **GroupId** element
- **SequenceNumber** element

3.3.1. GroupId Element

This element is used to specify the group of messages for which message ordering is guaranteed. This element is REQUIRED. This element MUST have a globally unique identifier as its value. The format of this identification is REQUIRED to be conform to MessageId, as defined in [RFC2822]. This element contains the following attributes:

- a **removeAfter** attribute
- a **status** attribute

(1) removeAfter attribute

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

- The sender sends a Message with the value of "End" in the status attribute.
- The time specified in the removeAfter attribute has passed.

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

(2) status attribute

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

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

- **End**: Indicating the message is the last message for a series of messages.

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

3.3.2. SequenceNumber Element

The SequenceNumber element is a REQUIRED element, when the MessageOrder element is present. This element is used for guaranteeing the message order within the group of messages categorized by the same GroupId value. (See 3.3.1 for detail). In other words, the sequence of numbered messages that the receiver node presents to the application MUST be in the same order as the sequence of numbered messages that the sender application has produced, within the group of messages having the same GroupId value.

If the MessageOrder element appears in the message sent, the receiver of the message is REQUIRED to make this message available to the application layer only after all messages with lower sequence number with the same GroupId have been made available to the application. In other words, an implementation of the receiver node MUST enforce the order in which messages are made available to the application, according to the sequence number order for messages with the same GroupId value. Example 6 illustrates this:

Example 6 Example of SequenceNumber element

1) First message:

```
< rm:MessageOrder SOAP:mustUnderstand="1">
    <rm:GroupId status="Start">020907-45261-0450@a.com</rm:GroupId>
    <rm:SequenceNumber>0</rm:SequenceNumber>
</rm:MessageOrder>
```

2) Second message:

```
< rm:MessageOrder SOAP:mustUnderstand="1">
    <rm:GroupId status="Continue">020907-45261-0450@a.com</rm:GroupId>
    <rm:SequenceNumber>1</rm:SequenceNumber>
</rm:MessageOrder>
```

3) Third message:

```
< rm:MessageOrder SOAP:mustUnderstand=”1”>
    < rm:GroupId status="End">020907-45261-0450@a.com</rm:GroupId>
    <rm:SequenceNumber>2</rm:SequenceNumber>
</rm:MessageOrder>
```

When a sender node communicates with a receiver node across several GroupId values, the sender MUST maintain a distinct counter of the value of SequenceNumber for each GroupId independently. When sending a message containing a MessageOrder element with a new GroupId, the sender is REQUIRED to generate a new value for the SequenceNumber element in the GroupId.

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

NOTE: Because delivery between the reliable messaging provider and the sequence is not specified, this is not a complete guarantee of ordering to the application.

3.4. RMResponse Element

The RMResponse element includes response information to be used for a reliable message. It is REQUIRED if the message is an Acknowledgment message or a Fault message. This element includes the following attribute and sub-elements:

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

Example 7 shows an example of RMResponse element.

Example 7 Example of RMResponse element

```
<rm:RMResponse xmlns:rm="http://schemas.fujitsu.com/rm"
 SOAP:mustUnderstand="1">
   <rm:MessageType>Acknowledgment</rm:MessageType>
    <rm:RefToMessageId>20020907-045261-0450@anyuri.com</rm:RefToMessageId>
</rm:RMResponse>
```

3.4.1. MessageType Ele ment

This is a REQUIRED element and is used to specify the type of the Response Message. It could have either of the following values:

- **Acknowledgment** : To indicate an Acknowledgment message
- **Fault** : To indicate a Fault message

3.4.2. RefToMessageId Element

The RefToMessageId element is a REQUIRED element. This element MUST contain the value of the original MessageId of the message received successfully when used in the Acknowledgment message, or for the message in error, when used in the Fault Message.

4. SOAP Fault

This section describes extensions to the fault codes defined in the SOAP 1.1 specification. Intended to carry error or status information for the SOAP layer, these fault code extensions MUST comply with SOAP Fault as defined in SOAP1.1. The SOAP Fault is used in this model for notification of only SOAP level errors and Reliable Messaging errors. Errors specific to Reliable Messaging are described in the following sections.

4.1. SOAP Fault extension for Reliable Messaging

To describe the details of the Reliable Messaging error, an additional rmFault element is defined as a sub-element of the detail element in a SOAP Fault element.

4.1.1. rmFault element

This element is OPTIONAL and if present MUST appear within a SOAP detail element. It contains only one sub-element:

- **faultcode**: To specify Reliable Messaging specific fault value

4.1.2. faultcode sub-element

This sub-element is REQUIRED and SHOULD have a value specified in Chart 1. The value should be namespace qualified. These fault codes are explained in detail in section 4.2.

Chart 1 RM faultcode values

Value of faultcode	Description
InvalidMessageHeader	Content or format of the Message Header element is invalid, or it was impossible to process the MessageHeader element for some reason.
InvalidMessageId	Content or format of the MessageId element is invalid, or it was impossible to process the MessageId element for some reason.
InvalidRefToMessageId	Content or format of the RefToMessageId element is invalid, or it was impossible to process the RefToMessageId element for some reason.
InvalidTimestamp	Content or format of the Timestamp element is invalid, or it was impossible to process the Timestamp element for some reason.
InvalidTimeToLive	Content or format of the TimeToLive element is invalid, or it was impossible to process the TimeToLive element for some reason.
InvalidReliableMessage	Content or format of the ReliableMessage element is invalid, or it was impossible to process the ReliableMessage element for some reason.
InvalidAckRequested	Content or format of the AckRequested element is invalid, it was impossible to process the AckRequested element for some reason, or the receiver couldn't send back Acknowledgment Message as it was specified in the synchronous attribute.
InvalidMessageOrder	Content or format of the MessageOrder element is invalid, or it was impossible to process the MessageOrder element for some reason.

Example 8 Fault Message for Reliable Messaging

```
<?xml version="1.0"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
     SOAP:mustUnderstand="1">
        <rm:From>requestor@anyuri.com</rm:From>
        <rm:To>responder@someuri.com</rm:To>
        <rm:Service>urn:services:ItemQuoteService</rm:Service>
        <rm:MessageId>20020907-12-34@anyuri.com</rm:MessageId>
        <rm:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
    </rm:MessageHeader>
    <rm:RMResponse>
         <rm:MessageType>Fault</rm:MessageType>
        <rm:RefToMessageId>20020907-03-30@someuri.com </rm:RefToMessageId>
    </rm:RMResponse>
 </SOAP:Header>
 <SOAP:Body>
  <SOAP:Fault>
   <faultcode>SOAP:Client</faultcode>
   <faultstring>fault from server</faultstring>
   <detail>
    <rm:rmFault xmlns:rm="http://schemas.fujitsu.com/rm">
          <rm:faultcode>rm:InvalidMessageHeader</rm:faultcode>
      </rm:rmFault>
   </detail>
  </SOAP:Fault>
 </SOAP:Body>
</SOAP:Envelope>
```

4.2. Fault code description

The following sections describe, in more detail, use of the error codes in Chart 5-1 .

4.2.1. InvalidMessageHeader

This is an error message to be used when the content or format of the MessageHeader is invalid. This error message will be used also when the type attribute specified in the

Service Element was not found. This error message also will be used to report receipt of invalid information in the From element or the To element.

4.2.2. InvalidMessageId

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

4.2.3. InvalidRefToMessageId

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

4.2.4. InvalidTimestamp

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

4.2.5. InvalidTimeToLive

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

4.2.6. InvalidReliableMessage

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

4.2.7. InvalidAckRequested

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

4.2.8. InvalidMessageOrder

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

5. HTTP Binding

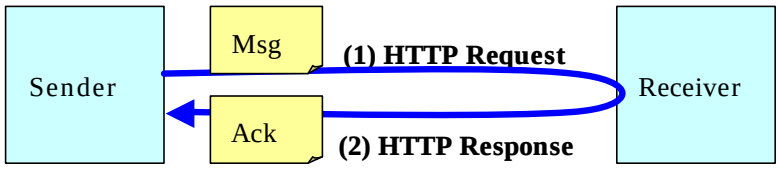
This section describes the HTTP binding for SOAP, when the original message is sent asynchronously at the application level. When supporting reliable messaging, upon receipt of a reliable message, the server MUST send a reply. This reply MUST be either an Acknowledgment or a Fault message. This reply MUST be sent either synchronously or asynchronously.

5.1. Reliable Messaging with Synchronous Acknowledgment or Fault Message

The Reliable Messaging Acknowledgment or Fault message MAY be sent back on the same HTTP connection as the HTTP Request that included the message being acknowledged or faulted. This is illustrated in Figures 5 and 6. The SOAP Fault MAY also be sent back asynchronously on a different HTTP connection, as illustrated in Figure 7.

(1) Synchronous Acknowledgment Message Sequence

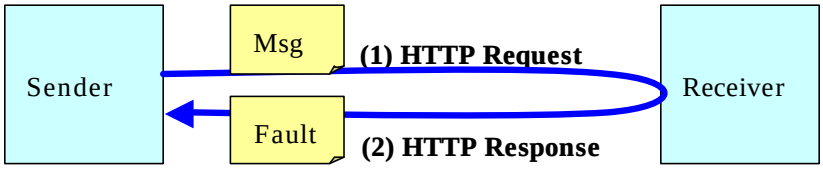
Figure 5 Synchronous Acknowledgment Message



1) The sender initiates an HTTP connection, and sends a Message using the HTTP POST Request. The Example 9 is an example of such a message.
2) The receiver sends back an Acknowledgment message to the sender on the same connection, as the HTTP response. There are two options – synchronous and asynchronous – for sending back the SOAP Fault to the sender. The following sections describe it in detail.

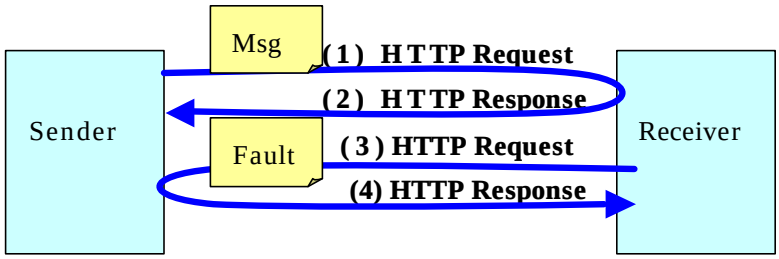
(2) Synchronous SOAP Fault Message Sequence

Figure 6 Synchronous SOAP Fault Message



(3) Asynchronous SOAP Fault Message Sequence

Figure 7 Asynchronous SOAP Fault Message

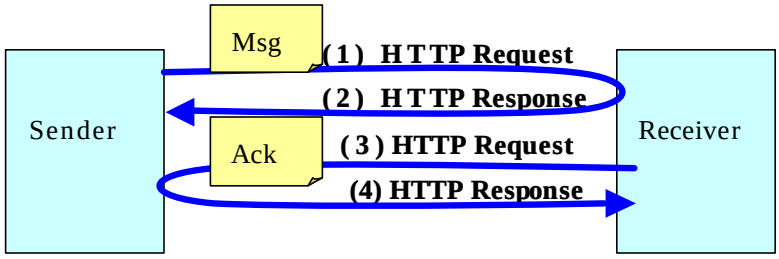


5.2. Reliable Messaging with Asynchronous Acknowledgment Message

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

(1) Asynchronous Acknowledgment Message Sequence

Figure 8 Asynchronous Acknowledgment Message

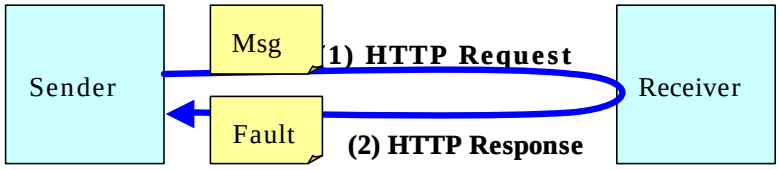


(1) The sender initiates a HTTP connection, and sends a Message using HTTP POST Request. Example 9 is an example of this message.
(2) The HTTP response to the (1) has no content. Example 10 is an example of this HTTP response.
(3) The Acknowledgment Message is sent with the other HTTP connection. Example 11 is an example of this message.
(4) The HTTP response for (3) has no content. Example 10 is an example for this HTTP Response.

There are two options – synchronous and asynchronous – for sending back a SOAP Fault to the sender when an error is detected by the receiver. These are illustrated in figure 9 and 10 respectively.

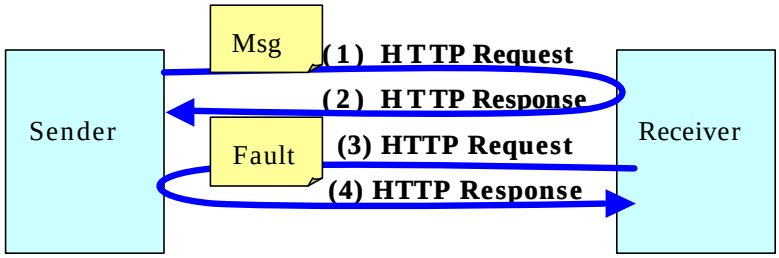
(2) Synchronous Fault Message Sequence

Figure 9 Synchronous SOAP Fault Message



(3) Asynchronous Fault Message Sequence

Figure 10 Asynchronous SOAP Fault Message



Example 9 Example of Reliable Message within HTTP POST Request

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version=”1.0”?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP:Header>
```

```
    <rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
     SOAP:mustUnderstand="1">
        < r m:From>requester@anyuri.com</rm:From>
        < r m:To>responder@someuri.com</rm:To>
        < r m:Service>urn:services:ItemQuoteService</rm:Service>
        < r m:MessageId >20020907-045261-0450@anyuri.com</rm:MessageId>
        < r m:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
    </rm:MessageHeader>
    <rm:ReliableMessage xmlns:rm="http://schemas.fujitsu.com/rm"
     SOAP:mustUnderstand="1">
         <rm:MessageType>Message</rm:MessageType>
         <rm:ReplyTo>requester@anyuri.com</rm:ReplyTo>
       <rm:AckRequested/>
        < r m:DuplicateElimination/>
    </rm:ReliableMessage>
  </SOAP:Header>
  <SOAP:Body>
       <gip:GetItemPrice xmlns:gip="Some-URI">
           <gip:itemnumber>product12345</gip:itemnumber>
     </gip:GetItemPrice>
 </SOAP:Body>
</SOAP:Envelope>
```

Example 10 Example of HTTP Response to the Example 6-1

```
HTTP/1.1 200 OK
Content-Length: 0
```

Example 11 Example of SOAP Fault message

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version="1.0"?>
<SOAP:Envelope
```

```
 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP:Header>
    <rm:MessageHeader xmlns:rm=”http://schemas.fujitsu.com/rm”
     SOAP:mustUnderstand=”1”>
         < r m:From>responder@someuri.com</rm:From>
         < r m:To>requester@anyuri.com</rm:To>
         < r m:Service>urn:services:ItemFilingService</rm:Service>
         < r m:MessageId>20020907-029762-0118@someuri.com</rm:MessageId>
         < r m:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
   </rm:MessageHeader>
    <rm:RMResponse xmlns:rm=”http://schemas.fujitsu.com/rm”
     SOAP:mustUnderstand=”1”>
         < r m:MessageType>Fault</rm:MessageType>
         < r m:RefToMessageId>20020907-045261-0450@anyuri.com
           </rm:RefToMessageId>
    </rm:RMResponse>
  </SOAP:Header>
  <SOAP:Body>
    <SOAP:Fault>
          <faultcode>SOAP:Client</faultcode>
          <faultstring>MessageId is invalid</faultstring>
    </SOAP:Fault>
 </SOAP:Body>
</SOAP:Envelope>
```

Example 12 Example of HTTP POST Request with Acknowledgment Message

```
POST /ItemQuote HTTP/1.1
Host: www.PartsShopServer.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: ""

<?xml version=”1.0”?>
<SOAP:Envelope
 xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
```

```
SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP:Header>
   <rm:MessageHeader xmlns:rm="http://schemas.fujitsu.com/rm"
    SOAP:mustUnderstand="1">
       < r m:From>responder@someuri.com</rm:From>
       < r m:To>requester@anyuri.com</rm:To>
       < r m:Service>urn:services:ItemFilingService</rm:Service>
       < r m:MessageId>20020907-047184-0297@someuri.com</rm:MessageId>
       < r m:Timestamp>2002-09-07T10:19:07</rm:Timestamp>
   </rm:MessageHeader>
   <rm:RMResponse xmlns:rm="http://schemas.fujitsu.com/rm"
    SOAP:mustUnderstand="1">
       < r m:MessageType>Acknowledgment</rm:MessageType>
       < r m:RefToMessageId>20020907-045261-0450@anyuri.com
         </rm:RefToMessageId>
   </rm:RMResponse>
 </SOAP:Header>
 <SOAP:Body>
</SOAP:Body>
</SOAP:Envelope>
```

6. Acknowledgements

Akira Ochi	Fujitsu Limited
Hirotaka Hara	Fujitsu Limited
Hiroyuki Tomisawa	Hitachi Limited
Katsuhisa Nakazato	Fujitsu Limited
Masahiko Narita	Fujitsu Limited
Nobuyuki Saji	NEC Corporation
Shuichi Imabayashi	Fujitsu Limited

7. References

7.1. Normative References

[RFC1738] "Uniform Resource Locators (URL)", T. Berners-Lee et al, RFC 1738, IESG and IETF, December 1994. Available at
http://www.ietf.org/rfc/rfc1738.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, Bradner, S., IESG and IETF, March 1997. Available at
http://www.ietf.org/rfc/rfc2119.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, Tim Berners-Lee et al, IESG and IETF, August 1998. Available at:
http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, R. Fielding et al, IESG and IETF, June 1999. Available at
http://www.ietf.org/rfc/rfc2616.txt

[RFC2822] "Internet Message Format", RFC 2822, P. Resnick, Editor, IESG and IETF, April 2001. Available at
http://www.ietf.org/rfc/rfc2822.txt

[SOAP 1.1] "Simple Object Access Protocol (SOAP) 1.1", Don Box et al, W3C Note, 8 May, 2000. Available at
http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

"Extensible Markup Language (XML) 1.0, Second Edition", Tim Bray et al, eds., W3C Recommendation, 6 October 2000. Available at
http://www.w3.org/TR/2000/REC-xml-20001006/

[XML Namespaces] "Namespaces in XML", Tim Bray et al., eds., W3C Recommendation, 14 January 1999. Available at
http://www.w3.org/TR/1999/REC-xml-names-19990114/

[XML Schema] "XML Schema Part 2: Datatypes", Paul V. Biron and Ashok Malhotra, eds. W3C Recommendation, 2 May 2001. Available at http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

7.2. Non-normative References

[ebMS] "Message Service Specification Version 2.0", OASIS ebXML Messaging Services Technical Committee, OASIS Standard, 1 April 2002. Available at http://www.ebxml.org/specs/ebMS2.pdf

[SOAP 1.2] "SOAP 1.2 Part 1: Messaging Framework", Martin Gudgin, Marc Hadley, Jean-Jacques Moreau, Henrik Frystyk Nielsen, eds., W3C Candidate Recommendation (work in progress), 20 December 2002. Available at http://www.w3.org/TR/2002/CR-soap12-part1-20021219/

[WSS] "OASIS Web Services Security TC", documentation in progress, OASIS Technical Committee. Committee home page at http://www.oasis-open.org/committees/wss/

"XML Schema Part 1: Structures", Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, eds., W3C Recommendation, 2 May 2001. Available at http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

Appendix 1 – Schema for WS-Reliability

```
<?xml version="1.0" encoding="utf-8"?>
<!--WS-Reliability Schema Definitions -->
<!-- Copyright (c) 2003 Fujitsu Limited, Sun Microsystems, Oracle Corp.,
      Sonic Software Corp., Hitachi Ltd., and NEC Corp. -->
<xsd:schema targetNamespace="http://schemas.fujitsu.com/rm"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:ns1="http://schemas.fujitsu.com/rm"
          xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          elementFormDefault="qualified" attributeFormDefault="unqualified">

          <xsd:import namespace="http://schemas.xmlsoap.org/soap/envelope/" />
          <xsd:simpleType name="RFC2822MessageIdType">
              <xsd:restriction base="xsd:string" />
          </xsd:simpleType>

          <xsd:element name="MessageHeader">
              <xsd:complexType>
                  <xsd:sequence>
                      <xsd:element ref="ns1:From" minOccurs="0"/>
                      <xsd:element ref="ns1:To" minOccurs="0"/>
                      <xsd:element ref="ns1:Service" minOccurs="0"/>
                      <xsd:element ref="ns1:MessageId"/>
                      <xsd:element ref="ns1:Timestamp"/>
                  </xsd:sequence>
                  <xsd:attribute ref="soap:mustUnderstand" use="required"/>
              </xsd:complexType>
          </xsd:element>

          <xsd:element name="From" type="xsd:anyURI"/>

          <xsd:element name="To" type="xsd:anyURI"/>

          <xsd:element name="Service" >
              <xsd:complexType>
```

```
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" type="xsd:string" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="MessageId" type="ns1:RFC2822MessageIdType"/>

<xsd:element name="Timestamp" type="xsd:dateTime"/>

<xsd:element name="ReliableMessage">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="MessageType" minOccurs="0">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Message"/>
                 </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>

            <xsd:element ref="ns1:ReplyTo"/>
            <xsd:element ref="ns1:TimeToLive" minOccurs="0"/>
            <xsd:element ref="ns1:AckRequested" minOccurs="0"/>
            <xsd:element ref="ns1:DuplicateElimination" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute ref="soap:mustUnderstand" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="ReplyTo" type="xsd:anyURI"/>

<xsd:element name="TimeToLive" type="xsd:dateTime"/>
```

```
<xsd:element name="AckRequested">
    <xsd:complexType>
        <xsd:attribute name="synchronous" type="xsd:boolean" use="optional"
          default="false"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="DuplicateElimination">
     <xsd:complexType/>
</xsd:element>

<xsd:element name="RMResponse">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="MessageType">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Acknowledgment" />
                        <xsd:enumeration value="Fault" />
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>

            <xsd:element ref="ns1:RefToMessageId"/>
        </xsd:sequence>
        <xsd:attribute ref="soap:mustUnderstand" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="RefToMessageId" type="ns1:RFC2822MessageIdType"/>

<xsd:element name="MessageOrder">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="ns1:GroupId"/>
            <xsd:element ref="ns1:SequenceNumber"/>
```

```
            </xsd:sequence>
            <xsd:attribute ref="soap:mustUnderstand" use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="GroupId" >
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="ns1:RFC2822MessageIdType">
                    <xsd:attribute name="removeAfter" type="xsd:dateTime"
                      use="optional"/>
                    <xsd:attribute name="status"
                      type="ns1:SequenceNumberStatusType" use="optional"/>
                </xsd:extension >
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="SequenceNumberStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Start"/>
            <xsd:enumeration value="Continue"/>
            <xsd:enumeration value="End"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:element name="SequenceNumber" type="xsd:unsignedLong"/>

    <xsd:element name="rmFault">
         <xsd:complexType>
              <xsd:sequence>
                   <xsd:element name="faultcode" type="xsd:QName"/>
              </xsd:sequence>
         </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Appendix 2 – Futures List

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

	Category	Details
1	Application of defined elements and attributes beyond RM	The existing specification contains a number of elements that may apply to many use cases beyond reliable messaging. For example, From / To may be useful for (unreliable) routing through an intermediary. Alignment on a set of common components meeting these general requirements across use cases should be sought.
2	WSDL	Define WSDL extensions profiling the use of RM SOAP extensions (e.g., align Service element with WSDL-SOAP binding)
3	Faults	Fault handling is from receiver back to the sender. There is no notion of a central Fault location equivalent to a dead message queue or dead letter queue. This would be useful in the case where a missing message in message ordering and sequencing is “never” received.
4	Order and sequencing	The behavioral semantics of senders and receivers need to be further defined with regard to the tracking of sequence numbers for the purpose of detecting duplicate or out of order messages. For example: How long should a receiver hold on to out-of-sequence messages in anticipation of a missing message?"
5	Persistence requirements	The specified persistence requirements are high. These requirements could be reduced and full support made configurable.
6	Combined message types	The current specification explicitly prevents bundling an acknowledgment for an earlier message with a request for an acknowledgment (and the associated payload). This restriction reduces the efficiency of long running message exchanges and should be lifted.
7	Sync/Async	Clarify whether sync HTTP binding is required for servers implementing the HTTP binding for WS-Reliability.
8	Negotiation	Add a negotiation mechanism (e.g., message exchange, notification message for flow control and restart of messaging after failure, etc.).
9	From and To	Clarify the use of the From and To Elements
10	Optionality	The use of the term OPTIONAL needs to be revisited particularly in a specification of this nature where interoperability is an explicit goal and RFC 2119 has been referenced.

11 Security	A reply (ACK or Fault) is required for reliable messaging, either synchronously or asynchronously. Possible denial of service issues should be considered.