



1 **Web Services Reliable Messaging** 2 **(WS-Reliable Messaging)**

3 **Working Draft 05, September 26th 2005**

Document identifier:

4 WS-ReliableMessaging-1.1-draft-03.doc

5 **Location:**

6 TBD

7 **Editors:**

8 Doug Davis, IBM <dug@us.ibm.com>

9 Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>

10 TBD

11 **Abstract:**

12 This specification (WS-ReliableMessaging) describes a protocol that allows messages
13 to be delivered reliably between distributed applications in the presence of software
14 component, system, or network failures. The protocol is described in this
15 specification in a transport-independent manner allowing it to be implemented using
16 different network technologies. To support interoperable Web services, a SOAP
17 binding is defined within this specification.

18 The protocol defined in this specification depends upon other Web services
19 specifications for the identification of service endpoint addresses and policies. How
20 these are identified and retrieved are detailed within those specifications and are out
21 of scope for this document.

22 **Composable Architecture:**

23 By using the SOAP [[SOAP](#)] and WSDL [[WSDL](#)] extensibility model, SOAP-based and
24 WSDL-based specifications are designed to be composed with each other to define a
25 rich Web services environment. As such, WS-ReliableMessaging by itself does not
26 define all the features required for a complete messaging solution. WS-
27 ReliableMessaging is a building block that is used in conjunction with other
28 specifications and application-specific protocols to accommodate a wide variety of
29 protocols related to the operation of distributed Web services.

30 **Status:**

31 TBD

32 Table of Contents

33	1INTRODUCTION.....	6
34	1.1GOALS AND REQUIREMENTS.....	6
35	1.1.1Requirements.....	6
36	1.2NOTATIONAL CONVENTIONS.....	6
37	1.3NAMESPACE.....	7
38	1.4COMPLIANCE.....	8
39	2RELIABLE MESSAGING MODEL.....	9
40	2.1GLOSSARY.....	10
41	2.2PROTOCOL PRECONDITIONS.....	11
42	2.3PROTOCOL INVARIANTS.....	11
43	2.4EXAMPLE MESSAGE EXCHANGE.....	11
44	3RM PROTOCOL ELEMENTS.....	14
45	3.1SEQUENCES.....	14
46	3.2SEQUENCE ACKNOWLEDGEMENT.....	16
47	3.3REQUEST ACKNOWLEDGEMENT.....	18
48	3.4SEQUENCE CREATION.....	19
49	3.5SEQUENCE TERMINATION.....	23
50	3.6CLOSING A SEQUENCE.....	24
51	4FAULTS.....	27
52	4.1SEQUENCEFAULT ELEMENT.....	29
53	4.2SEQUENCE TERMINATED.....	30
54	4.3UNKNOWN SEQUENCE.....	30
55	4.4INVALID ACKNOWLEDGEMENT.....	30
56	4.5MESSAGE NUMBER ROLLOVER.....	31
57	4.6LAST MESSAGE NUMBER EXCEEDED.....	31
58	4.7CREATE SEQUENCE REFUSED.....	32
59	4.8SEQUENCE CLOSED.....	32
60	5SECURITY CONSIDERATIONS.....	33
61	6REFERENCES.....	35
62	6.1NORMATIVE.....	35
63	6.2NON-NORMATIVE.....	36

64	APPENDIX A.SCHEMA	37
65	APPENDIX B.MESSAGE EXAMPLES.....	42
66	B.1.CREATE SEQUENCE.....	43
67	B.2. INITIAL TRANSMISSION.....	45
68	B.3.FIRST ACKNOWLEDGEMENT.....	47
69	B.4.RETRANSMISSION.....	48
70	B.5.TERMINATION.....	49
71	APPENDIX C.WSDL.....	51
72	APPENDIX D.ACKNOWLEDGMENTS.....	53
73	APPENDIX E.REVISION HISTORY.....	54
74	APPENDIX F.NOTICES.....	55

1 Introduction

It is often a requirement for two Web services that wish to communicate to do so reliably in the presence of software component, system, or network failures. The primary goal of this specification is to create a modular mechanism for reliable message delivery. It defines a messaging protocol to identify, track, and manage the reliable delivery of messages ~~between exactly two parties, a source and a destination~~. It also defines a SOAP binding that is required for interoperability. Additional bindings may be defined.

This mechanism is extensible allowing additional functionality, such as security, to be tightly integrated. This specification integrates with and complements the WS-Security, WS-Policy, and other Web services specifications. Combined, these allow for a broad range of reliable, secure messaging options.

1.1 Goals and Requirements

1.1.1 Requirements

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [[KEYWORDS](#)].

This specification uses the following syntax to define normative outlines for messages:

- The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- Characters are appended to elements and attributes to indicate cardinality:
 - "?" (0 or 1)
 - "*" (0 or more)
 - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

- An ellipsis (i.e. "...") indicates a point of extensibility that allows other child, or attribute, content. Additional children elements and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If an extension is not recognized it SHOULD be ignored.
- XML namespace prefixes (See Section [Namespace](#)) are used to indicate the namespace of the element being defined.

1.3 Namespace

The XML namespace [[XML-ns](#)] URI that MUST be used by implementations of this specification is:

<http://docs.oasis-open.org/wsrn/200510/>

Table 1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

The following namespaces are used in this document:

Table 1

Prefix	Namespace
S	http://www.w3.org/2003/05/soap-envelope
S11	http://schemas.xmlsoap.org/soap/envelope/
wsrn	http://docs.oasis-open.org/wsrn/200510/
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

The normative schema for WS-Reliable Messaging can be found at:

<http://docs.oasis-open.org/wsrn/200510/wsrn.xsd>

All sections explicitly noted as examples are informational and are not to be considered normative.

If an action URI is used, and one is not already defined per the rules of the WS-Addressing specification [WS-Addressing], then the action URI MUST consist of the reliable messaging namespace URI concatenated with the "/" character and the element name. For example:

126 <http://docs.oasis-open.org/wsrn/200510/SequenceAcknowledgement>

127 **1.4 Compliance**

128 An implementation is not compliant with this specification if it fails to satisfy one or
129 more of the MUST or REQUIRED level requirements defined herein. A SOAP Node
130 MUST NOT use the XML namespace identifier for this specification (listed in
131 Section [Namespace](#)) within SOAP Envelopes unless it is compliant with this
132 specification.

133 Normative text within this specification takes precedence over normative outlines,
134 which in turn take precedence over the XML Schema [[XML Schema Part 1](#), [Part 2](#)]
135 descriptions.

2 Reliable Messaging Model

Many errors may interrupt a conversation. Messages may be lost, duplicated or reordered. Further the host systems may experience failures and lose volatile state.

The WS-ReliableMessaging specification defines an interoperable protocol that requires a Reliable Messaging (RM) Source and Reliable Messaging (RM) Destination to ensure that each message transmitted by the RM Source is successfully received by an RM Destination, or barring successful receipt, that an RM Source can, except in the most extreme circumstances, accurately determine the disposition of each message transmitted as perceived by the RM Destination, so as to resolve any in-doubt status. Note that this specification make no restriction on the scope of an RM Source or RM Destination entities.

In addition, The protocol allows the RM Source and RM Destination to provide their respective Application Source and Application Destination a guarantee that a message that is sent by an Application Source will be delivered to the Application Destination.

This guarantee is specified as a delivery assurance. It is the responsibility of the RM Source and RM Destination to fulfill the delivery assurances on behalf of their respective Application counterparts, or raise an error. The protocol defined here allows endpoints to meet this guarantee for the delivery assurances defined below. However, the means by which these delivery assurances are manifested by either the RM Source or RM Destination roles is an implementation concern, and is out of scope of this specification.

Note that the underlying protocol defined in this specification remains the same regardless of the delivery assurance.

Persistence considerations related to an endpoint's ability to satisfy the delivery assurances defined below are the responsibility of the implementation and do not affect the wire protocol. As such, they are out of scope of this specification.

There are four basic delivery assurances that endpoints can provide:

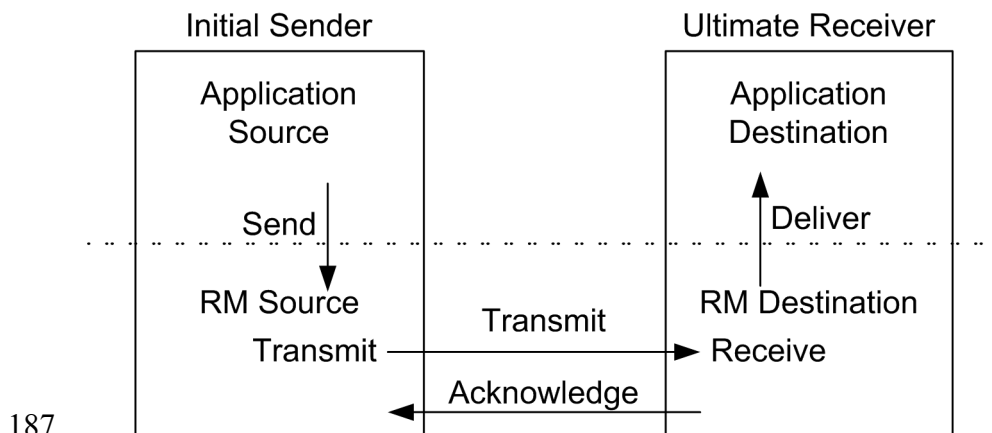
AtMostOnce Messages will be delivered at most once without duplication or an error will be raised on at least one endpoint. It is possible that some messages in a sequence may not be delivered.

AtLeastOnce Every message sent will be delivered or an error will be raised on at least one endpoint. Some messages may be delivered more than once.

170 **ExactlyOnce** Every message sent will be delivered without duplication or an error
171 will be raised on at least one endpoint. This delivery assurance is the logical "and" of
172 the two prior delivery assurances.

173 **InOrder** Messages will be delivered in the order that they were sent. This delivery
174 assurance may be combined with any of the above delivery assurances. It requires
175 that the messages within a Sequence will be delivered in an order so that the
176 message numbers are monotonically increasing. Note that this assurance says
177 nothing about duplications or omissions. Note also that it is only applicable to
178 messages in the same Sequence. Cross Sequence ordering of messages is not in the
179 scope of this specification.

180 Figure 1 below illustrates the entities and events in a simple reliable message
181 exchange. First, the Application Source Sends a message for reliable delivery. The
182 Reliable Messaging (RM) Source accepts the message and Transmits it one or more
183 times. After receiving the message, the RM Destination Acknowledges it. Finally,
184 the RM Destination delivers the message to the Application Destination. The exact
185 roles the entities play and the complete meaning of the events will be defined
186 throughout this specification.



188 Figure 1: Reliable Messaging Model

189 2.1 Glossary

190 The following definitions are used throughout this specification:

191 **Endpoint:** A referencable entity, processor, or resource where Web service messages
192 are originated or targeted.

193 **Application Source:** The endpoint that Sends a message.

194 **Application Destination:** The endpoint to which a message is Delivered.
195 **Delivery Assurance:** The guarantee that the messaging infrastructure provides on
196 the delivery of a message.
197 **Receive:** The act of reading a message from a network connection and qualifying it
198 as relevant to RM Destination functions.
199 **RM Source:** ~~For any one reliable message t~~The endpoint that transmits the
200 message.
201 **RM Destination:** ~~For any one reliable message t~~The endpoint that receives the
202 message.
203 **Send:** The act of submitting a message to the RM Source for reliable delivery. The
204 reliability guarantee begins at this point.
205 **Deliver:** The act of transferring a message from the RM Destination to the
206 Application Destination. The reliability guarantee is fulfilled at this point.
207 **Transmit:** The act of writing a message to a network connection.
208 **Receive:** The act of reading a message from a network connection.
209 **Acknowledgement:** The communication from the RM Destination to the RM Source
210 indicating the successful receipt of a message.

211 2.2 Protocol Preconditions

212 The correct operation of the protocol requires that a number of preconditions MUST
213 be established prior to the processing of the initial sequenced message:
214 • The RM Source MUST have an endpoint reference that uniquely identifies the RM Destination
215 ~~endpoint~~; correlations across messages addressed to the unique endpoint MUST be
216 meaningful.
217 • The RM Source MUST have knowledge of the destination's policies, if any, and the RM
218 Source MUST be capable of formulating messages that adhere to this policy.
219 If a secure exchange of messages is required, then the RM Source and RM
220 Destination MUST have a security context.

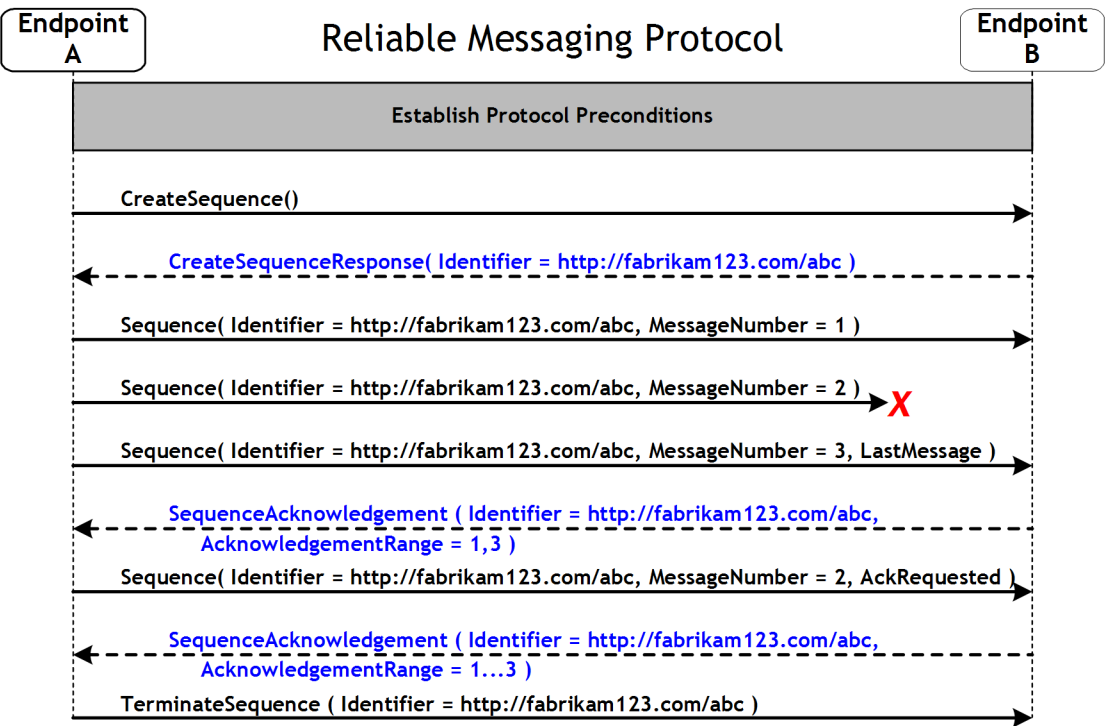
221 2.3 Protocol Invariants

222 During the lifetime of the protocol, two invariants are REQUIRED for correctness:
223 • The RM Source MUST assign each reliable message a sequence number (defined below)
224 beginning at 1 and increasing by exactly 1 for each subsequent reliable message.

225 Every acknowledgement issued by the RM Destination MUST include within an
226 acknowledgement range or ranges the sequence number of every message
227 successfully received by the RM Destination and MUST exclude sequence numbers of
228 any messages not yet received.

229 **2.4 Example Message Exchange**

230 Figure 2 illustrates a possible message exchange between two reliable messaging
231 endpoints A and B.



232 Figure 2: The WS-ReliableMessaging Protocol

- 233 1. The protocol preconditions are established. These include policy exchange,
234 endpoint resolution, establishing trust.
- 235 2. The RM Source requests creation of a new Sequence.
- 236 3. The RM Destination creates a Sequence by returning a globally unique identifier.
- 237 4. The RM Source begins sending messages beginning with MessageNumber 1. In
238 the figure the RM Source sends 3 messages.

- 239 5. Since the 3rd message is the last in this exchange, the RM Source includes a
240 <wsrm:LastMessage> token.
- 241 6. The 2nd message is lost in transit.
- 242 7. The RM Destination acknowledges receipt of message numbers 1 and 3 in
243 response to the RM Source's <wsrm:LastMessage> token.
- 244 8. The RM Source retransmits the 2nd message. This is a new message on the
245 underlying transport, but since it has the same sequence identifier and message
246 number so the RM Destination can recognize it as equivalent to the earlier
247 message, in case both are received.
- 248 9. The RM Source includes an <wsrm:AckRequested> element so the RM Destination
249 will expedite an acknowledgement.
- 250 10. The RM Destination receives the second transmission of the message with
251 MessageNumber 2 and acknowledges receipt of message numbers 1, 2, and 3
252 which carried the <wsrm:LastMessage> token.
- 253 11. The RM Source receives this acknowledgement and sends a TerminateSequence
254 message to the RM Destination indicating that the sequence is completed and
255 reclaims any resources associated with the Sequence.
- 256 12. The RM Destination receives the TerminateSequence message indicating that the
257 RM Source will not be sending any more messages, and reclaims any resources
258 associated with the Sequence.
- 259 Now that the basic model has been outlined, the details of the elements used in this
260 protocol are now provided in Section 3.

3 RM Protocol Elements

The protocol elements define extensibility points at various places. Additional children elements and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver SHOULD ignore the extension.

3.1 Sequences

The RM protocol uses a `<wsrm:Sequence>` header block to track and manage the reliable delivery of messages. Messages for which the delivery assurance applies MUST contain a `<wsrm:Sequence>` header block. Each Sequence MUST have a unique `<wsrm:Identifier>` element and each message within a Sequence MUST have a `<wsrm:MessageNumber>` element that increments by 1 from an initial value of 1. These values are contained within a `<wsrm:Sequence>` header block accompanying each message being delivered in the context of a Sequence. In addition to mandatory `<wsrm:Identifier>` and `<wsrm:MessageNumber>` elements, the header MAY include a `<wsrm:LastMessage>` element.

There MUST be no more than one `<wsrm:Sequence>` header block in any message.

The purpose of the `<wsrm:LastMessage>` element is to signal to the RM Destination that the message represents the last message in the Sequence.

A following exemplar defines its syntax:

```
<wsrm:Sequence ...>
  <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
  <wsrm:MessageNumber> xs:unsignedLong </wsrm:MessageNumber>
  <wsrm:LastMessage/>?
  ...
</wsrm:Sequence>
```

The following describes the content model of the Sequence header block.

/wsrm:Sequence

This is the element containing Sequence information for WS-ReliableMessaging. The `<wsrm:Sequence>` element MUST be understood by the RM Destination. The `<wsrm:Sequence>` element MUST have a `mustUnderstand` attribute with a value 1/true from the namespace corresponding to the version of SOAP to which the `<wsrm:Sequence>` SOAP header block is bound.

/wsrm:Sequence/wsrm:Identifier

294 This REQUIRED element MUST contain an absolute URI conformant with RFC2396 that uniquely
295 identifies the Sequence.

296 /wsrm:Sequence/wsrm:Identifier/@{any}

297 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
298 to the element.

299 /wsrm:Sequence/wsrm:MessageNumber

300 This REQUIRED element MUST contain an xs:unsignedLong representing the ordinal position of
301 the message within a Sequence. Sequence MessageNumbers start at 1 and monotonically
302 increase throughout the Sequence. If the message number exceeds the internal limitations of an
303 RM Source or RM Destination or reaches the maximum value of an xs:unsignedLong
304 (18,446,744,073,709,551,615), the RM Source or Destination MUST issue a
305 MessageNumberRollover fault.

306 /wsrm:Sequence/wsrm:LastMessage

307 This element MAY be included by the RM Source ~~endpoint~~. The <wsrm:LastMessage> element
308 has no content.

309 /wsrm:Sequence/{any}

310 This is an extensibility mechanism to allow different types of information, based on a schema, to
311 be passed.

312 /wsrm:Sequence/@{any}

313 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
314 to the element.

315 A RM Source ~~endpoint~~ MUST include a <wsrm:LastMessage> element in the
316 <wsrm:Sequence> element for the last message in a Sequence. An RM Destination
317 ~~endpoint~~ MUST respond with a <wsrm:SequenceAcknowledgement> upon receipt of a
318 <wsrm:LastMessage> element. A Sequence MUST NOT use a <wsrm:MessageNumber>
319 value greater than that which accompanies a <wsrm:LastMessage> element. An RM
320 Destination MUST generate a LastMessageNumberExceeded (See Section 4.6) fault
321 upon receipt of such a message. In the event that an RM Source needs to close a
322 Sequence and there is no application message, the RM Source MAY send a message
323 with an empty body containing <wsrm:Sequence> header with the
324 <wsrm:LastMessage> element. In this usage, the action URI MUST be:

325 `http://docs.oasis-open.org/wsrm/200510/LastMessage`

326 in preference to the pattern defined in Section 1.2.

327 The following example illustrates a Sequence header block.

328 `<wsrm:Sequence>`

```

329     <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
330     <wsrm:MessageNumber>10</wsrm:MessageNumber>
331     <wsrm:LastMessage/>
332 </wsrm:Sequence>

```

333 3.2 Sequence Acknowledgement

334 The RM Destination informs the RM Source of successful message receipt using a
 335 <wsrm:SequenceAcknowledgement> header block. The
 336 <wsrm:SequenceAcknowledgement> header block MAY be transmitted independently
 337 or included on return messages. The RM Destination MAY send a
 338 <wsrm:SequenceAcknowledgement> header block at any point during which the
 339 sequence is valid. The timing of acknowledgements can be advertised using policy
 340 and acknowledgements can be explicitly requested using the <wsrm:AckRequested>
 341 directive (see Section 3.3). If a non-mustUnderstand fault occurs when processing
 342 an RM Header that was piggy-backed on another message, a fault MUST be
 343 generated, but the processing of the original message MUST NOT be affected.

344 The following exemplar defines its syntax:

```

345 <wsrm:SequenceAcknowledgement ...>
346   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
347   [ [ <wsrm:AcknowledgementRange ...
348     Upper="xs:unsignedLong"
349     Lower="xs:unsignedLong"/> +
350     <wsrm:Final/> ? ]
351   | <wsrm:Nack> xs:unsignedLong </wsrm:Nack> +
352   | <wsrm:None/> ]
353   ...
354 </wsrm:SequenceAcknowledgement>

```

355 The following describes the content model of the <wsrm:SequenceAcknowledgement>
 356 header block.

357 /wsrm:SequenceAcknowledgement

358 This element contains the Sequence acknowledgement information.

359 /wsrm:SequenceAcknowledgement/wsrm:Identifier

360 This REQUIRED element MUST contain an absolute URI conformant with RFC2396 that uniquely
 361 identifies the Sequence.

362 /wsrm:SequenceAcknowledgement/wsrm:Identifier/@{any}

363 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 364 to the element.

365 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange

366 This OPTIONAL element, if present, can occur 1 or more times. It contains a range of message

367 Sequence MessageNumbers successfully received by the receiving endpoint manager. The

368 ranges SHOULD NOT overlap. This element MUST NOT be present if either the <wsrm:Nack>

369 or <wsrm:None> elements are also present as a child of

370 <wsrm:SequenceAcknowledgement>.

371 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper

372 This REQUIRED attribute contains an xs:unsignedLong representing the

373 <wsrm:MessageNumber> of the highest contiguous message in a Sequence range received by

374 the RM Destination.

375 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower

376 This REQUIRED attribute contains an xs:unsignedLong representing the

377 <wsrm:MessageNumber> of the lowest contiguous message in a Sequence range received by

378 the RM Destination.

379 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}

380 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added

381 to the element.

382 /wsrm:SequenceAcknowledgement/wsrm:Final

383 This OPTIONAL element, if present, indicates that the RM Destination is not receiving new

384 messages for the specified Sequence. The RM Source can be assured that the ranges of

385 messages acknowledged by this SequenceAcknowledgement header block will not change in the

386 future. This element MUST be present when the Sequence is no longer receiving new message

387 for the specified sequence. Note: this element MUST NOT be used when sending a Nack, it can

388 only be used when sending AcknowledgementRanges.

389 /wsrm:SequenceAcknowledgement/wsrm:Nack

390 This OPTIONAL element, if present, MUST contain an xs:unsignedLong representing the

391 <wsrm:MessageNumber> of an unreceived message in a Sequence. This element permits the

392 gap analysis of the <wsrm:AcknowledgementRange> elements to be performed at the RM

393 Destination rather than at the RM Source which may yield performance benefits in certain

394 environments. The <wsrm:Nack> element MUST NOT be present if either the

395 <wsrm:AcknowledgementRange> or <wsrm:None> elements are also present as a child of

396 <wsrm:SequenceAcknowledgement>. Upon the receipt of a Nack, an RM Source SHOULD

397 retransmit the message identified by the Nack. The RM Destination MUST NOT issue a

398 <wsrm:SequenceAcknowledgement> containing a <wsrm:Nack> for a message that it has

399 previously acknowledged within a <wsrm:AcknowledgementRange>. The RM Source SHOULD

400 ignore a <wsrm:SequenceAcknowledgement> containing a <wsrm:Nack> for a message

401 that has previously been acknowledged within a <wsrm:AcknowledgementRange>.

402 /wsrm:SequenceAcknowledgement/wsrm:None

403 This OPTIONAL element, if present, MUST be used when the RM Destination has not received
404 any messages for the specified sequence. The <wsrm:None> element MUST NOT be present if
405 either the <wsrm:AcknowledgementRange> or <wsrm:Nack> elements are also present as a
406 child of the <wsrm:SequenceAcknowledgement>.

407 /wsrm:SequenceAcknowledgement/{any}

408 This is an extensibility mechanism to allow different (extensible) types of information, based on a
409 schema, to be passed.

410 /wsrm:SequenceAcknowledgement/@{any}

411 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
412 to the element.

413 The following examples illustrate <wsrm:SequenceAcknowledgement> elements:

- 414 • Message numbers 1..10 inclusive in a Sequence have been received by the RM Destination.

```
415 <wsrm:SequenceAcknowledgement>  
416   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
417   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
418 </wsrm:SequenceAcknowledgement>
```

- 419 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been received by the
420 RM Destination, messages 3 and 7 have not been received.

```
421 <wsrm:SequenceAcknowledgement>  
422   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
423   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
424   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
425   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
426 </wsrm:SequenceAcknowledgement>
```

- 427 • Message number 3 in a Sequence has not been received by the RM Destination.

```
428 <wsrm:SequenceAcknowledgement>  
429   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
430   <wsrm:Nack>3</wsrm:Nack>  
431 </wsrm:SequenceAcknowledgement>
```

432 3.3 Request Acknowledgement

433 The purpose of the <wsrm:AckRequested> header block is to signal to the RM
434 Destination that the RM Source is requesting that a
435 <wsrm:SequenceAcknowledgement> be returned.

436 At any time, the RM Source may request an acknowledgement message from the RM
437 Destination **endpoint** using an `<wsrm:AckRequested>` header block.

438 The RM Source **endpoint** requests this acknowledgement by including an
439 `<wsrm:AckRequested>` header block in the message. An RM Destination that receives
440 a message that contains an `<wsrm:AckRequested>` header block MUST respond with
441 a message containing a `<wsrm:SequenceAcknowledgement>` header block. If a non-
442 mustUnderstand fault occurs when processing an RM Header that was piggy-backed
443 on another message, a fault MUST be generated, but the processing of the original
444 message MUST NOT be affected.

445 The following exemplar defines its syntax:

```
446 <wsrm:AckRequested ...>  
447   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
448   <wsrm:MessageNumber> xs:unsignedLong </wsrm:MessageNumber> ?  
449   ...  
450 </wsrm:AckRequested>
```

451 `/wsrm:AckRequested`

452 This element requests an acknowledgement for the identified sequence.

453 `/wsrm:AckRequested/wsrm:Identifier`

454 This REQUIRED element MUST contain an absolute URI, conformant with RFC2396, that
455 uniquely identifies the Sequence to which the request applies.

456 `/wsrm:AckRequested/wsrm:Identifier/@{any}`

457 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
458 to the element.

459 `/wsrm:AckRequested/wsrm:MessageNumber`

460 This OPTIONAL element, if present, MUST contain an *xs:unsignedLong* representing the highest
461 `<wsrm:MessageNumber>` sent by the RM Source within the Sequence. If present, it MAY be
462 treated as a hint to the RM Destination as an optimization to the process of preparing to transmit a
463 `<wsrm:SequenceAcknowledgement>`.

464 `/wsrm:AckRequested/{any}`

465 This is an extensibility mechanism to allow different (extensible) types of information, based on a
466 schema, to be passed.

467 `/wsrm:AckRequested/@{any}`

468 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
469 to the element.

3.4 Sequence Creation

The RM Source MUST request creation of an outbound Sequence by sending a `<wsrm:CreateSequence>` element in the body of a message to the RM Destination which in turn responds either with a `<wsrm:CreateSequenceResponse>` or a `CreateSequenceRefused` fault in the body of the response message. `<wsrm:CreateSequence>` MAY carry an offer to create an inbound sequence which is either accepted or rejected in the `<wsrm:CreateSequenceResponse>`.

~~The RM Destination of the outbound sequence is the WS-Addressing EndpointReference [WS-Addressing] to which `<wsrm:CreateSequence>` is sent. The RM Destination of the inbound sequence is the WS-Addressing `<wsa:ReplyTo>` of the `<wsrm:CreateSequence>`.~~

The following exemplar defines the `<wsrm:CreateSequence>` syntax:

```
<wsrm:CreateSequence ...>
  <wsrm:AcksTo ...> wsa:EndpointReferenceType </wsrm:AcksTo>
  <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
  <wsrm:Offer ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
    ...
  </wsrm:Offer> ?
  ...
</wsrm:CreateSequence>
```

`/wsrm:CreateSequence`

This element requests creation of a new Sequence between the RM Source that sends it, and the RM Destination to which it is sent. This element MUST NOT be sent as a header block. The RM Destination MUST respond either with a `<wsrm:CreateSequenceResponse>` response message or a `CreateSequenceRefused` fault.

`/wsrm:CreateSequence/wsrm:AcksTo`

This REQUIRED element, of type `wsa:EndpointReferenceType` as specified by WS-Addressing [WS-Addressing] specifies the endpoint reference to which `<wsrm:SequenceAcknowledgement>` messages and faults related to the created Sequence are to be sent.

`/wsrm:CreateSequence/wsrm:Expires`

This element, if present, of type `xs:duration` specifies the RM Source's requested duration for the Sequence. The RM Destination MAY either accept the requested duration or assign a lesser value of its choosing. A value of 'PT0S' indicates that the Sequence will never expire. Absence of the element indicates an implied value of 'PT0S'.

507 /wsrm:CreateSequence/wsrm:Expires/@{any}
 508 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 509 to the element.

510 /wsrm:CreateSequence/wsrm:Offer
 511 This element, if present, enables an RM Source to offer a corresponding Sequence for the reliable
 512 exchange of messages transmitted from RM Destination to RM Source.

513 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier
 514 This REQUIRED element MUST contain an absolute URI conformant with RFC2396 that uniquely
 515 identifies the offered Sequence.

516 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier/@{any}
 517 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 518 to the element.

519 /wsrm:CreateSequence/wsrm:Offer/wsrm:Expires
 520 This element, if present, of type `xs:duration` specifies the duration for the Sequence. A value
 521 of 'PT0S' indicates that the Sequence will never expire. Absence of the element indicates an
 522 implied value of 'PT0S'.

523 /wsrm:CreateSequence/wsrm:Offer/wsrm:Expires/@{any}
 524 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 525 to the element.

526 /wsrm:CreateSequence/wsrm:Offer/{any}
 527 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 528 schema, to be passed.

529 /wsrm:CreateSequence/wsrm:Offer/@{any}
 530 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 531 schema, to be passed.

532 OPTIONAL/wsrn:CreateSequence/{any}
 533 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 534 schema, to be passed.

535 /wsrm:CreateSequence/@{any}
 536 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 537 to the element.

538 A `<wsrm:CreateSequenceResponse>` is sent in the body of a response message by an
539 RM Destination in response to receipt of a `<wsrm:CreateSequence>` request
540 message. It carries the `<wsrm:Identifier>` of the created Sequence and indicates
541 that the RM Source may begin sending messages in the context of the identified
542 Sequence.

543 The following exemplar defines the `<wsrm:CreateSequenceResponse>` syntax:

```
544 <wsrm:CreateSequenceResponse ...>
545   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
546   <wsrm:Expires> xs:duration </wsrm:Expires> ?
547   <wsrm:Accept ...>
548     <wsrm:AcksTo ...> wsa:EndpointReferenceType </wsrm:AcksTo>
549     ...
550   </wsrm:Accept> ?
551   ...
552 </wsrm:CreateSequenceResponse>
```

553 `/wsrm:CreateSequenceResponse`

554 This element is sent in the body of the response message in response to a
555 `<wsrm:CreateSequence>` request message. It indicates that the RM Destination has created
556 a new Sequence at the request of the RM Source. This element MUST NOT be sent as a header
557 block.

558 `/wsrm:CreateSequenceResponse/wsrm:Identifier`

559 This REQUIRED element MUST contain an absolute URI conformant with RFC2396 of the
560 Sequence that has been created by the RM Destination.

561 `/wsrm:CreateSequenceResponse/wsrm:Identifier/@{any}`

562 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
563 to the element.

564 `/wsrm:CreateSequenceResponse/wsrm:Expires`

565 This element, if present, of type *xs:duration* accepts or refines the RM Source's requested
566 duration for the Sequence. A value of 'PT0S' indicates that the Sequence will never expire.
567 Absence of the element indicates an implied value of 'PT0S'. This value MUST be equal or lesser
568 than the value requested by the RM Source in the corresponding `<wsrm:CreateSequence>`
569 message.

570 `/wsrm:CreateSequenceResponse/wsrm:Expires/@{any}`

571 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
572 to the element.

573 `/wsrm:CreateSequenceResponse/wsrm:Accept`

574 This element, if present, enables an RM Destination to accept the offer of a corresponding
575 Sequence for the reliable exchange of messages transmitted from RM Destination to RM Source.
576 This element MUST be present if the corresponding `<wsrm:CreateSequence>` message
577 contained an `<wsrm:Offer>` element.

578 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

579 This REQUIRED element, of type `wsa:EndpointReferenceType` as specified by WS-Addressing
580 [WS-Addressing], specifies the endpoint reference to which
581 `<wsrm:SequenceAcknowledgement>` messages related to the accepted Sequence are to be
582 sent.

583 `/wsrm:CreateSequenceResponse/wsrm:Accept/{any}`

584 This is an extensibility mechanism to allow different (extensible) types of information, based on a
585 schema, to be passed.

586 `/wsrm:CreateSequenceResponse/wsrm:Accept/@{any}`

587 This is an extensibility mechanism to allow different (extensible) types of information, based on a
588 schema, to be passed.

589 `/wsrm:CreateSequenceResponse/{any}`

590 This is an extensibility mechanism to allow different (extensible) types of information, based on a
591 schema, to be passed.

592 `/wsrm:CreateSequenceResponse/@{any}`

593 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
594 to the element.

595 **3.5 Sequence Termination**

596 When the RM Source has completed its use of the Sequence, it sends a
597 `<wsrm:TerminateSequence>` element, in the body of a message to the RM
598 Destination to indicate that the Sequence is complete, and that it will not be sending
599 any further messages related to the Sequence. The RM Destination can safely reclaim
600 any resources associated with the Sequence upon receipt of the
601 `<wsrm:TerminateSequence>` message. Note, under normal usage the RM source will
602 complete its use of the sequence when all of the messages in the Sequence have
603 been acknowledged. However, the RM Source is free to Terminate or Close a
604 Sequence at any time regardless of the acknowledgement state of the messages.

605 The following exemplar defines the TerminateSequence syntax:

```
606 <wsrm:TerminateSequence ...>  
607   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
```

608 ...

609 </wsrm:TerminateSequence>

610 /wsrm:TerminateSequence

611 This element is sent by an RM Source to indicate it has completed its use of the Sequence, i.e. it

612 MUST NOT send any additional message to the RM Destination referencing this sequence. It

613 indicates that the RM Destination can safely reclaim any resources related to the identified

614 Sequence. This element MUST NOT be sent as a header block.

615 /wsrm:TerminateSequence/wsrm:Identifier

616 This REQUIRED element MUST contain an absolute URI conformant with RFC2396 of the

617 Sequence that is being terminated.

618 /wsrm:TerminateSequence/wsrm:Identifier/@{any}

619 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added

620 to the element.

621 /wsrm:TerminateSequence/{any}

622 This is an extensibility mechanism to allow different (extensible) types of information, based on a

623 schema, to be passed.

624 /wsrm:TerminateSequence/@{any}

625 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added

626 to the element.

627 **3.6 Closing A Sequence**

628 There may be times during the use of an RM Sequence that the RM Source or RM

629 Destination will wish to discontinue using a Sequence even if some of the messages

630 have not been successfully delivered to the RM Destination.

631 In the case where the RM Source wishes to discontinue use of a sequence, while it

632 can send a TerminateSequence to the RM Destination, since this is a one-way

633 message and due to the possibility of late arriving (or lost) messages and

634 Acknowledgements, this would leave the RM Source unsure of the final ranges of

635 messages that were successfully delivered to the RM Destination.

636 To alleviate this, the RM Source can send a <wsrm:CloseSequence> element, in the

637 body of a message, to the RM Destination to indicate that RM Destination MUST NOT

638 receive any new messages for the specified sequence, other than those already

639 received at the time the <wsrm:CloseSequence> element is interpreted by the RMD.

640 Upon receipt of this message the RM Destination MUST send a

641 SequenceAcknowledgement to the RM Source. Note, this
642 SequenceAcknowledgement MUST include the <wsrm:Final> element.

643 While the RM Destination MUST NOT receive any new messages for the specified
644 sequence it MUST still process RM protocol messages. For example, it MUST respond
645 to AckRequested, TerminateSequence as well as CloseSequence messages. Note,
646 subsequent CloseSequence messages have no effect on the state of the sequence.

647 In the case where the RM Destination wishes to discontinue use of a sequence it may
648 'close' the sequence itself. Please see wsrm:Final above and the SequenceClosed
649 fault below. Note, the SequenceClosed Fault SHOULD be used in place of the
650 SequenceTerminated Fault, whenever possible, to allow the RM Source to still receive
651 Acknowledgements.

652 The following exemplar defines the CloseSequence syntax:

```
653 <wsrm:CloseSequence wsrm:Identifier="xs:anyURI"/>
```

654 /wsrm:CloseSequence

655 This element is sent by an RM Source to indicate that the RM Destination MUST NOT receive any
656 new messages for this sequence. A SequenceClosed fault MUST be generated by the RM
657 Destination when it receives a message for a sequence that is closed.

658 /wsrm:CloseSequence@Identifier

659 This REQUIRED attribute contains an absolute URI conformant with RFC2396 that uniquely
660 identifies the sequence.

661 /wsrm:CloseSequence/{any}

662 This is an extensibility mechanism to allow different (extensible) types of information, based on a
663 schema, to be passed.

664 /wsrm:CloseSequence@{any}

665 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
666 to the element.

667 A <wsrm:CloseSequenceResponse> is sent in the body of a response message by an
668 RM Destination in response to receipt of a <wsrm:CloseSequence> request message.
669 It indicates that the RM Destination has closed the sequence.

670 The following exemplar defines the <wsrm:CloseSequenceResponse> syntax:

```
671 /wsrm:CloseSequenceResponse
```

672 /wsrm:CloseSequenceResponse

673 This element is sent in the body of a response message by an RM Destination in response to
674 receipt of a <wsrm:CloseSequence> request message. It indicates that the RM Destination has
675 closed the sequence.

676 /wsrm:CloseSequenceResponse/{any}

677 This is an extensibility mechanism to allow different (extensible) types of information, based on a
678 schema, to be passed.

679 /wsrm:CloseSequenceResponse@{any}

680 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
681 to the element.

4 Faults

The fault definitions defined in this section reference certain abstract properties, such as [fault endpoint], that are defined in section 3 of the WS-Addressing [WS-Addressing] specification. Endpoints compliant with this specification MUST include required Message Addressing Properties on all fault messages.

Sequence creation uses a CreateSequence, CreateSequenceResponse request-response pattern. Faults for this operation are treated as defined in WS-Addressing. CreateSequenceRefused is a possible fault reply for this operation.

UnknownSequence is a fault generated by endpoints when messages carrying RM header blocks targeted at unrecognized sequences are detected, these faults are also treated as defined in WS-Addressing. All other faults in this section relate to the processing of RM header blocks targeted at known sequences and are collectively referred to as sequence faults. Sequence faults SHOULD be sent to the same [destination] as <wsrm:SequenceAcknowledgement> messages. These faults are correlated using the Sequence identifier carried in the detail.

WS-ReliableMessaging faults MUST include as the [action] property the default fault action URI defined in the version of WS-Addressing used in the message. The value from the current version is below for informational purposes:

`http://schemas.xmlsoap.org/ws/2004/08/addressing/fault`

The faults defined in this section are generated if the condition stated in the preamble is met. Fault handling rules are defined in section 4 of WS-Addressing.

The definitions of faults use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

The [Code] property MUST be either "Sender" or "Receiver". These properties are serialized into text XML as follows:

SOAP Version	Sender	Receiver
SOAP 1.1	S11:Client	S11:Server
SOAP 1.2	S:Sender	S:Receiver

The properties above bind to a SOAP 1.2 fault as follows:

issue10 10/18/2005

```

711 <S:Envelope>
712   <S:Header>
713     <wsa:Action>
714       http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
715     </wsa:Action>
716     <!-- Headers elided for clarity. -->
717   </S:Header>
718   <S:Body>
719     <S:Fault>
720       <S:Code>
721         <S:Value> [Code] </S:Value>
722         <S:Subcode>
723           <S:Value> [Subcode] </S:Value>
724         </S:Subcode>
725       </S:Code>
726       <S:Reason>
727         <S:Text xml:lang="en"> [Reason] </S:Text>
728       </S:Reason>
729       <S:Detail>
730         [Detail]
731         ...
732       </S:Detail>
733     </S:Fault>
734   </S:Body>
735 </S:Envelope>

```

736 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered
737 by processing an RM header block:

```

738 <S11:Envelope>
739   <S11:Header>
740     <wsrm:SequenceFault>
741       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
742       ...
743     </wsrm:SequenceFault>
744     <!-- Headers elided for clarity. -->
745   </S11:Header>
746   <S11:Body>
747     <S11:Fault>
748       <faultcode> [Code] </faultcode>
749       <faultstring> [Reason] </faultstring>
750     </S11:Fault>
751   </S11:Body>

```

752 </S11:Envelope>

753 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a
754 result of processing a <wsrm:CreateSequence> request message:

```
755       <S11:Envelope>
756       <S11:Body>
757       <S11:Fault>
758        <faultcode> [Subcode] </faultcode>
759        <faultstring xml:lang="en"> [Reason] </faultstring>
760       </S11:Fault>
761       </S11:Body>
762       </S11:Envelope>
```

763 4.1 SequenceFault Element

764 The purpose of the <wsrm:SequenceFault> element is to carry the specific details of
765 a fault generated during the reliable messaging specific processing of a message
766 belonging to a Sequence. The <wsrm:SequenceFault> container MUST only be used
767 in conjunction with the SOAP1.1 fault mechanism. It MUST NOT be used in
768 conjunction with the SOAP1.2 binding.

769 The following exemplar defines its syntax:

```
770       <wsrm:SequenceFault ...>
771        <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
772        ...
773       </wsrm:SequenceFault>
```

774 The following describes the content model of the *SequenceFault* element.

775 /wsrm:SequenceFault

776 This is the element containing Sequence information for WS-ReliableMessaging

777 /wsrm:SequenceFault/wsrm:FaultCode

778 This element, if present, MUST contain a qualified name from the set of fault codes defined
779 below.

780 /wsrm:SequenceFault/{any}

781 This is an extensibility mechanism to allow different (extensible) types of information, based on a
782 schema, to be passed.

783 /wsrm:SequenceFault/@{any}

784 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
785 to the element.

786 4.2 Sequence Terminated

787 This fault is sent by either the RM Source or the RM Destination to indicate that ~~the~~
788 ~~endpoint that generated the fault it~~ has either encountered an unrecoverable
789 condition, or has detected a violation of the protocol and as a consequence, has
790 chosen to terminate the sequence. The endpoint that generates this fault should
791 make every reasonable effort to notify the corresponding endpoint of this decision.

792 Properties:

793 [Code] Sender or Receiver

794 [Subcode] wsrn:SequenceTerminated

795 [Reason] The Sequence has been terminated due to an unrecoverable error.

796 [Detail]

797 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

798 4.3 Unknown Sequence

799 This fault is sent by either the RM Source or the RM Destination in response to a
800 message containing an unknown sequence identifier.

801 Properties:

802 [Code] Sender

803 [Subcode] wsrn:UnknownSequence

804 [Reason] The value of wsrn:Identifier is not a known Sequence identifier.

805 [Detail]

806 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

807 4.4 Invalid Acknowledgement

808 This fault is sent by the RM Source in response to a
809 `<wsrm:SequenceAcknowledgement>` that violates the cumulative acknowledgement
810 invariant. An example of such a violation would be a SequenceAcknowledgement
811 covering messages that have not been sent.

812 [Code] Sender

813 [Subcode] wsrn:InvalidAcknowledgement

814 [Reason] The SequenceAcknowledgement violates the cumulative acknowledgement
815 invariant.

816 [Detail]

817 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

818 **4.5 Message Number Rollover**

819 This fault is sent to indicate that message numbers for a sequence have been
820 exhausted.

821 Properties:

822 [Code] Sender

823 [Subcode] wsrn:MessageNumberRollover

824 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

825 [Detail]

826 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

827 **4.6 Last Message Number Exceeded**

828 This fault is sent by an RM Destination to indicate that it has received a message that
829 has a `<wsrm:MessageNumber>` within a Sequence that exceeds the value of the
830 `<wsrm:MessageNumber>` element that accompanied a `<wsrm:LastMessage>` element
831 for the Sequence.

832 Properties:

833 [Code] Sender

834 [Subcode] wsrn:LastMessageNumberExceeded

835 [Reason] The value for wsrn:MessageNumber exceeds the value of the
836 MessageNumber accompanying a LastMessage element in this Sequence.

837 [Detail]

838 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

839 **4.7 Create Sequence Refused**

840 This fault is sent in response to a create sequence request that cannot be satisfied.

841 Properties:

842 [Code] Sender

843 [Subcode] wsrn:CreateSequenceRefused

844 [Reason] The create sequence request has been refused by the RM Destination.

845 [Detail] empty

846 **4.8 Sequence Closed**

847 This fault is sent by an RM Destination to indicate that the specified sequence has
848 been closed. This fault MUST be generated when an RM Destination is asked to
849 receive a message for a sequence that is closed.

850 Properties:

851 [Code] Sender

852 [Subcode] wsrn:SequenceClosed

853 [Reason] The sequence is closed and can not receive new messages.

854 [Detail] <wsrm:Identifier...> xs:anyURI </wsrm:Identifier>

5 Security Considerations

It is strongly recommended that the communication between services be secured using the mechanisms described in WS-Security. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the `<wsrm:Sequence>` header needs to be signed with the body in order to "bind" the two together. The `<wsrm:SequenceAcknowledgement>` header may be signed independently because a reply independent of the message is not a security concern.

Because Sequences are expected to exchange a number of messages, it is recommended that a security context be established using the mechanisms described in WS-Trust and WS-SecureConversation [SecureConversation]. If a Sequence is bound to a specific endpoint, then the security context needs to be established or shared with the endpoint servicing the Sequence. While the context can be established at any time, it is critical that the messages establishing the Sequence be secured even if they precede security context establishment. However, it is recommended that the security context be established first. Security contexts are independent of reliable messaging Sequences. Consequently, security contexts can come and go independent of the lifetime of the Sequence. In fact, it is recommended that the lifetime of a security context be less than the lifetime of the Sequence unless the Sequence is very short-lived.

It is common for message Sequences to exchange a number of messages (or a large amount of data). As a result, the usage profile of a Sequence is such that it is susceptible to key attacks. For this reason it is strongly recommended that the keys be changed frequently. This "re-keying" can be effected a number of ways. The following list outlines four common techniques:

- Closing and re-establishing a security context
- Exchanging new secrets between the parties
- Using a derived key sequence and switch "generations"
- Attaching a nonce to each message and using it in a derived key function with the shared secret

The security context may be re-established using the mechanisms described in WS-Trust and WS-SecureConversation. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret should not be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

891 There is a core tension between security and reliable messaging that can be
892 problematic if not considered in implementations. That is, one aspect of security is
893 to prevent message replay and the core tenet of reliable messaging is to replay
894 messages until they are acknowledged. Consequently, if the security sub-system
895 processes a message but a failure occurs before the reliable messaging sub-system
896 records the message (or the message is considered "processed"), then it is possible
897 (and likely) that the security sub-system will treat subsequent copies as replays and
898 discard them. At the same time, the reliable messaging sub-system will likely
899 continue to expect and even solicit the missing message(s). Care should be taken to
900 avoid and prevent this rare condition.

901 The following list summarizes common classes of attacks that apply to this protocol
902 and identifies the mechanism to prevent/mitigate the attacks:

- 903 • **Message alteration** – Alteration is prevented by including signatures of the message
904 information using WS-Security.
- 905 • **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-
906 Security.
- 907 • **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by
908 comparing secured policies – see WS-Policy and WS-SecurityPolicy).
- 909 • **Authentication** – Authentication is established using the mechanisms described in WS-
910 Security and WS-Trust. Each message is authenticated using the mechanisms described in
911 WS-Security.
- 912 • **Accountability** – Accountability is a function of the type of and string of the key and
913 algorithms being used. In many cases, a strong symmetric key provides sufficient
914 accountability. However, in some environments, strong PKI signatures are required.
- 915 • **Availability** – All reliable messaging services are subject to a variety of availability attacks.
916 Replay detection is a common attack and it is recommended that this be addressed by the
917 mechanisms described in WS-Security. (Note that because of legitimate message replays,
918 detection should include a differentiator besides message id such as a timestamp). Other
919 attacks, such as network-level denial of service attacks are harder to avoid and are outside
920 the scope of this specification. That said, care should be taken to ensure that minimal state is
921 saved prior to any authenticating sequences.

922 6 References

923 6.1 Normative

924 [KEYWORDS]

925 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard
926 University, March 1997

927 [SOAP]

928 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

929 [URI]

930 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#),"
931 RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

932 [XML-ns]

933 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

934 [XML-Schema1]

935 W3C Recommendation, "[XML Schema Part 1: Structures](#)," 2 May 2001.

936 [XML-Schema2]

937 W3C Recommendation, "[XML Schema Part 2: Datatypes](#)," 2 May 2001.

938 [WSSecurity]

939 "[OASIS Web Services Security: SOAP Message Security 1.0 \(WS-Security 2004\)](#)",
940 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds, OASIS
941 Standard 200401, March 2004.

942 [Tanenbaum]

943 "Computer Networks," Andrew S. Tanenbaum, Prentice Hall PTR, 2003.

944 [WSDL]

945 W3C Note, "[Web Services Description Language \(WSDL 1.1\)](#)," 15 March 2001.

946 [WS-Addressing]

947 D. Box, et al, "[Web Services Addressing \(WS-Addressing\)](#)," August 2004.

948 6.2 Non-Normative

949 [WS-Policy]

issue10

10/18/2005

Copyright © OASIS Open 2005. All Rights Reserved.

Page 34 of 54

950 D. Box, et al, "[Web Services Policy Framework \(WS-Policy\)](#)," September 2004.

951 **[WS-PolicyAttachment]**

952 D. Box, et al, "[Web Services Policy Attachment \(WS-PolicyAttachment\)](#)," September 2004.

953 **[SecurityPolicy]**

954 G. Della-Libra, "[Web Services Security Policy Language \(WS-SecurityPolicy\)](#)," December 2002.

955 **[SecureConversation]**

956 S. Anderson, et al, "[Web Services Secure Conversation Language \(WS-SecureConversation\)](#),"

957 May 2004.

958

959 Appendix A.Schema

960 The normative schema for WS-ReliableMessaging is located at:

961 <http://docs.oasis-open.org/wsrn/200510/wsrn.xsd>

962 The following copy is provided for reference.

```
963 <xs:schema targetNamespace="http://docs.oasis-open.org/wsrn/200510/"
964 xmlns:xs="http://www.w3.org/2001/XMLSchema"
965 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
966 xmlns:wsrm="http://docs.oasis-open.org/wsrn/200510/"
967 elementFormDefault="qualified" attributeFormDefault="unqualified">
968   <xs:import
969 namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
970 schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
971   <!-- Protocol Elements -->
972   <xs:complexType name="SequenceType">
973     <xs:sequence>
974       <xs:element ref="wsrm:Identifier"/>
975       <xs:element name="MessageNumber" type="xs:unsignedLong"/>
976       <xs:element name="LastMessage" minOccurs="0">
977         <xs:complexType>
978           <xs:sequence/>
979         </xs:complexType>
980       </xs:element>
981       <xs:any namespace="##other" processContents="lax" minOccurs="0"
982 maxOccurs="unbounded"/>
983     </xs:sequence>
984     <xs:anyAttribute namespace="##other" processContents="lax"/>
985   </xs:complexType>
986   <xs:element name="Sequence" type="wsrm:SequenceType"/>
987   <xs:element name="SequenceAcknowledgement">
988     <xs:complexType>
989       <xs:sequence>
990         <xs:element ref="wsrm:Identifier"/>
991         <xs:choice>
992           <ws:sequence>
993             <xs:element name="AcknowledgementRange"
994 maxOccurs="unbounded">
995               <xs:complexType>
996                 <xs:sequence/>
```

```

997         <xs:attribute name="Upper" type="xs:unsignedLong"
998 use="required"/>
999         <xs:attribute name="Lower" type="xs:unsignedLong"
1000 use="required"/>
1001         <xs:anyAttribute namespace="##other"
1002 processContents="lax"/>
1003     </xs:complexType>
1004 </xs:element>
1005     <ws:element name="Final" minOccurs="0">
1006         <xs:complexType>
1007             <xs:sequence/>
1008         </xs:complexType>
1009     </ws:element>
1010 </ws:sequence>
1011     <xs:element name="Nack" type="xs:unsignedLong"
1012 minOccurs="unbounded"/>
1013     <xs:element name="None" minOccurs="0">
1014         <xs:complexType>
1015             <xs:sequence/>
1016         </xs:complexType>
1017     </xs:element>
1018 </xs:choice>
1019     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1020 minOccurs="unbounded"/>
1021 </xs:sequence>
1022     <xs:anyAttribute namespace="##other" processContents="lax"/>
1023 </xs:complexType>
1024 </xs:element>
1025 <xs:complexType name="AckRequestedType">
1026     <xs:sequence>
1027         <xs:element ref="wsrm:Identifier"/>
1028         <xs:element name="MessageNumber" type="xs:unsignedLong"
1029 minOccurs="0"/>
1030     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1031 minOccurs="unbounded"/>
1032 </xs:sequence>
1033     <xs:anyAttribute namespace="##other" processContents="lax"/>
1034 </xs:complexType>
1035 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1036 <xs:element name="Identifier">
1037     <xs:complexType>
1038         <xs:annotation>

```

```

1039     <xs:documentation>
1040 This type is for elements whose [children] is an anyURI and can have
1041 arbitrary attributes.
1042     </xs:documentation>
1043 </xs:annotation>
1044 <xs:simpleContent>
1045     <xs:extension base="xs:anyURI">
1046         <xs:anyAttribute namespace="##other" processContents="lax"/>
1047     </xs:extension>
1048 </xs:simpleContent>
1049 </xs:complexType>
1050 </xs:element>
1051 <!-- Fault Container and Codes -->
1052 <xs:simpleType name="FaultCodes">
1053     <xs:restriction base="xs:QName">
1054         <xs:enumeration value="wsrm:UnknownSequence"/>
1055         <xs:enumeration value="wsrm:SequenceTerminated"/>
1056         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1057         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1058         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1059         <xs:enumeration value="wsrm:LastMessageNumberExceeded"/>
1060     </xs:restriction>
1061 </xs:simpleType>
1062 <xs:complexType name="SequenceFaultType">
1063     <xs:sequence>
1064         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1065         <xs:any namespace="##any" processContents="lax" minOccurs="0"
1066 maxOccurs="unbounded"/>
1067     </xs:sequence>
1068     <xs:anyAttribute namespace="##any" processContents="lax"/>
1069 </xs:complexType>
1070 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1071 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1072 <xs:element name="CreateSequenceResponse"
1073 type="wsrm:CreateSequenceResponseType"/>
1074 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1075 <xs:element name="CloseSequenceResponse"
1076 type="wsrm:CloseSequenceResponseType"/>
1077 <xs:element name="TerminateSequence"
1078 type="wsrm:TerminateSequenceType"/>
1079 <xs:complexType name="CreateSequenceType">
1080     <xs:sequence>

```

```

1081     <xs:element ref="wsrm:AcksTo"/>
1082     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1083     <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1084     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1085 maxOccurs="unbounded">
1086         <xs:annotation>
1087             <xs:documentation>
1088 It is the authors intent that this extensibility be used to transfer a
1089 Security Token Reference as defined in WS-Security.
1090 </xs:documentation>
1091         </xs:annotation>
1092     </xs:any>
1093 </xs:sequence>
1094 <xs:anyAttribute namespace="##other" processContents="lax"/>
1095 </xs:complexType>
1096 <xs:complexType name="CreateSequenceResponseType">
1097     <xs:sequence>
1098         <xs:element ref="wsrm:Identifier"/>
1099         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1100         <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1101         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1102 maxOccurs="unbounded"/>
1103     </xs:sequence>
1104     <xs:anyAttribute namespace="##other" processContents="lax"/>
1105 </xs:complexType>
1106 <xs:complexType name="CloseSequenceType">
1107     <xs:sequence>
1108         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1109 maxOccurs="unbounded"/>
1110     </xs:sequence>
1111     <xs:attribute name="Identifier" type="xs:anyURI" use="required"/>
1112     <xs:anyAttribute namespace="##other" processContents="lax"/>
1113 </xs:complexType>
1114 <xs:complexType name="CloseSequenceResponseType">
1115     <xs:sequence>
1116         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1117 maxOccurs="unbounded"/>
1118     </xs:sequence>
1119     <xs:anyAttribute namespace="##other" processContents="lax"/>
1120 </xs:complexType>
1121 <xs:complexType name="TerminateSequenceType">
1122     <xs:sequence>

```

```

1123     <xs:element ref="wsrm:Identifier"/>
1124     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1125 maxOccurs="unbounded"/>
1126   </xs:sequence>
1127   <xs:anyAttribute namespace="##other" processContents="lax"/>
1128 </xs:complexType>
1129 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1130 <xs:complexType name="OfferType">
1131   <xs:sequence>
1132     <xs:element ref="wsrm:Identifier"/>
1133     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1134     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1135 maxOccurs="unbounded"/>
1136   </xs:sequence>
1137   <xs:anyAttribute namespace="##other" processContents="lax"/>
1138 </xs:complexType>
1139 <xs:complexType name="AcceptType">
1140   <xs:sequence>
1141     <xs:element ref="wsrm:AcksTo"/>
1142     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1143 maxOccurs="unbounded"/>
1144   </xs:sequence>
1145   <xs:anyAttribute namespace="##other" processContents="lax"/>
1146 </xs:complexType>
1147 <xs:element name="Expires">
1148   <xs:complexType>
1149     <xs:simpleContent>
1150       <xs:extension base="xs:duration">
1151         <xs:anyAttribute namespace="##other" processContents="lax"/>
1152       </xs:extension>
1153     </xs:simpleContent>
1154   </xs:complexType>
1155 </xs:element>
1156 </xs:schema>

```


1157 **Appendix B.Message Examples**

1158 B.1.Create Sequence

1159 Create Sequence

```
1160 <?xml version="1.0" encoding="UTF-8"?>
1161 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1162 xmlns:wsrm="http://docs.oasis-open.org/wsrn/200510/"
1163 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1164   <S:Header>
1165     <wsa:MessageID>
1166       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1167     </wsa:MessageID>
1168     <wsa:To>http://example.com/serviceB/123</wsa:To>
1169     <wsa:Action>http://docs.oasis-
1170 open.org/wsrn/200510/CreateSequence</wsa:Action>
1171     <wsa:ReplyTo>
1172       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1173     </wsa:ReplyTo>
1174   </S:Header>
1175   <S:Body>
1176     <wsrm:CreateSequence>
1177       <wsrm:AcksTo>
1178         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1179       </wsrm:AcksTo>
1180     </wsrm:CreateSequence>
1181   </S:Body>
1182 </S:Envelope>
```

1183 Create Sequence Response

```
1184 <?xml version="1.0" encoding="UTF-8"?>
1185 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1186 xmlns:wsrm="http://docs.oasis-open.org/wsrn/200510/"
1187 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1188   <S:Header>
1189     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1190     <wsa:RelatesTo>
1191       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1192     </wsa:RelatesTo>
1193     <wsa:Action>
1194       http://docs.oasis-open.org/wsrn/200510/CreateSequenceResponse
```

```
1195     </wsa:Action>
1196 </S:Header>
1197 <S:Body>
1198   <wsrm:CreateSequenceResponse>
1199     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1200   </wsrm:CreateSequenceResponse>
1201 </S:Body>
1202 </S:Envelope>
```

B.2. Initial Transmission

The following example WS-ReliableMessaging headers illustrate the message exchange in the above figure. The three messages have the following headers; the third message is identified as the last message in the sequence:

Message 1

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsm="http://docs.oasis-open.org/wsm/200510/"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
    </wsa:MessageID>
    <wsa:To>http://example.com/serviceB/123</wsa:To>
    <wsa:From>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:From>
    <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
    <wsm:Sequence>
      <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
      <wsm:MessageNumber>1</wsm:MessageNumber>
    </wsm:Sequence>
  </S:Header>
  <S:Body>
    <!-- Some Application Data -->
  </S:Body>
</S:Envelope>
```

Message 2

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsm="http://docs.oasis-open.org/wsm/200510/"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
    </wsa:MessageID>
    <wsa:To>http://example.com/serviceB/123</wsa:To>
```

```

1240     <wsa:From>
1241         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1242     </wsa:From>
1243     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1244     <wsrm:Sequence>
1245         <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1246         <wsrm:MessageNumber>2</wsrm:MessageNumber>
1247     </wsrm:Sequence>
1248 </S:Header>
1249 <S:Body>
1250     <!-- Some Application Data -->
1251 </S:Body>
1252 </S:Envelope>

```

1253 Message 3

```

1254 <?xml version="1.0" encoding="UTF-8"?>
1255 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1256 xmlns:wsrm="http://docs.oasis-open.org/wsrm/200510/"
1257 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1258     <S:Header>
1259         <wsa:MessageID>
1260             http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1261         </wsa:MessageID>
1262         <wsa:To>http://example.com/serviceB/123</wsa:To>
1263         <wsa:From>
1264             <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1265         </wsa:From>
1266         <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1267         <wsrm:Sequence>
1268             <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1269             <wsrm:MessageNumber>3</wsrm:MessageNumber>
1270             <wsrm:LastMessage/>
1271         </wsrm:Sequence>
1272     </S:Header>
1273     <S:Body>
1274         <!-- Some Application Data -->
1275     </S:Body>
1276 </S:Envelope>

```

1277 B.3.First Acknowledgement

1278 Message number 2 has not been received by the RM Destination due to some
1279 transmission error so it responds with an acknowledgement for messages 1 and 3:

```
1280 <?xml version="1.0" encoding="UTF-8"?>
1281 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1282 xmlns:wsrm="http://docs.oasis-open.org/wsrn/200510/"
1283 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1284   <S:Header>
1285     <wsa:MessageID>
1286       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1287     </wsa:MessageID>
1288     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1289     <wsa:From>
1290       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1291     </wsa:From>
1292     <wsa:Action>
1293       http://docs.oasis-open.org/wsrn/200510/SequenceAcknowledgement
1294     </wsa:Action>
1295     <wsrm:SequenceAcknowledgement>
1296       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1297       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1298       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1299     </wsrm:SequenceAcknowledgement>
1300   </S:Header>
1301   <S:Body/>
1302 </S:Envelope>
```

1303 B.4.Retransmission

1304 The ~~sending-endpoint~~RM Source discovers that message number 2 was not received
1305 so it resends the message and requests an acknowledgement:

```
1306 <?xml version="1.0" encoding="UTF-8"?>
1307 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1308 xmlns:wsrm="http://docs.oasis-open.org/wsrm/200510/"
1309 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
1310   <S:Header>
1311     <wsa:MessageID>
1312       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1313     </wsa:MessageID>
1314     <wsa:To>http://example.com/serviceB/123</wsa:To>
1315     <wsa:From>
1316       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1317     </wsa:From>
1318     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1319     <wsrm:Sequence>
1320       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1321       <wsrm:MessageNumber>2</wsrm:MessageNumber>
1322     </wsrm:Sequence>
1323     <wsrm:AckRequested>
1324       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1325     </wsrm:AckRequested>
1326   </S:Header>
1327   <S:Body>
1328     <!-- Some Application Data -->
1329   </S:Body>
1330 </S:Envelope>
```

B.5.Termination

The RM Destination now responds with an acknowledgement for the complete sequence which can then be terminated:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsm="http://docs.oasis-open.org/wsm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
    </wsa:MessageID>
    <wsa:To>http://Business456.com/serviceA/789</wsa:To>
    <wsa:From>
      <wsa:Address>http://example.com/serviceB/123</wsa:Address>
    </wsa:From>
    <wsa:Action>
      http://docs.oasis-open.org/wsm/200510/SequenceAcknowledgement
    </wsa:Action>
    <wsm:SequenceAcknowledgement>
      <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
      <wsm:AcknowledgementRange Upper="3" Lower="1"/>
    </wsm:SequenceAcknowledgement>
  </S:Header>
  <S:Body/>
</S:Envelope>
```

Terminate Sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsm="http://docs.oasis-open.org/wsm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
    </wsa:MessageID>
    <wsa:To>http://example.com/serviceB/123</wsa:To>
    <wsa:Action>
      http://docs.oasis-open.org/wsm/200510/TerminateSequence
    </wsa:Action>
```



```
1369     <wsa:From>
1370     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1371     </wsa:From>
1372 </S:Header>
1373 <S:Body>
1374     <wsrm:TerminateSequence>
1375     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1376     </wsrm:TerminateSequence>
1377 </S:Body>
1378 </S:Envelope>
```

1379 Appendix C.WSDL

1380 The non-normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1381 <http://docs.oasis-open.org/wsrn/200510/wsd1/wsrn.wsd1>

1382 The following non-normative copy is provided for reference.

```
1383 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1384 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1385 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1386 xmlns:rm="http://docs.oasis-open.org/wsrn/200510/"
1387 xmlns:tns="http://docs.oasis-open.org/wsrn/200510/wsd1"
1388 targetNamespace="http://docs.oasis-open.org/wsrn/200510/wsd1">
1389 <wSDL:types>
1390   <xs:schema>
1391     <xs:import namespace="http://docs.oasis-open.org/wsrn/200510/"
1392     schemaLocation="http://docs.oasis-open.org/wsrn/200510/wsrn.xsd"/>
1393     <xs:import
1394     namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
1395     schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
1396   </xs:schema>
1397 </wSDL:types>
1398 <wSDL:message name="CreateSequence">
1399   <wSDL:part name="create" element="rm:CreateSequence"/>
1400 </wSDL:message>
1401 <wSDL:message name="CreateSequenceResponse">
1402   <wSDL:part name="createResponse"
1403   element="rm:CreateSequenceResponse"/>
1404 </wSDL:message>
1405 <wSDL:message name="CloseSequence">
1406   <wSDL:part name="close" element="rm:CloseSequence"/>
1407 </wSDL:message>
1408 <wSDL:message name="CloseSequenceResponse">
1409   <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1410 </wSDL:message>
1411 <wSDL:message name="TerminateSequence">
1412   <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1413 </wSDL:message>
1414 <wSDL:portType name="SequenceAbstractPortType">
1415   <wSDL:operation name="CreateSequence">
```

```
1416         <wsdl:input message="tns:CreateSequence"
1417 wsa:Action="http://docs.oasis-open.org/wsrn/200510/CreateSequence"/>
1418         <wsdl:output message="tns:CreateSequenceResponse"
1419 wsa:Action="http://docs.oasis-
1420 open.org/wsrn/200510/CreateSequenceResponse"/>
1421     </wsdl:operation>
1422     <wsdl:operation name="CloseSequence">
1423         <wsdl:input name="tns:CloseSequence"
1424 wsa:Action="http://docs.oasis-open.org/wsrn/200510/CloseSequence"/>
1425         <wsdl:output name="tns:CloseSequenceResponse"
1426 wsa:Action="http://docs.oasis-
1427 open.org/wsrn/200510/CloseSequenceResponse"/>
1428     </wsdl:operation>
1429     <wsdl:operation name="TerminateSequence">
1430         <wsdl:input message="tns:TerminateSequence"
1431 wsa:Action="http://docs.oasis-
1432 open.org/wsrn/200510/CreateSequenceResponse"/>
1433     </wsdl:operation>
1434 </wsdl:portType>
1435 </wsdl:definitions>
```

1436 **Appendix D.Acknowledgments**

1437 This document is based on initial contribution to OASIS WS-RX Technical Committee by the
1438 following authors: Ruslan Bilorusets, BEA, Don Box, Microsoft, Luis Felipe Cabrera, Microsoft,
1439 Doug Davis, IBM, Donald Ferguson, IBM, Christopher Ferris, IBM (Editor), Tom Freund, IBM,
1440 Mary Ann Hondo, IBM, John Ibbotson, IBM, Lei Jin, BEA, Chris Kaler, Microsoft, David
1441 Langworthy, Microsoft (Editor), Amelia Lewis, TIBCO Software, Rodney Limprecht, Microsoft,
1442 Steve Lucco, Microsoft, Don Mullen, TIBCO Software, Anthony Nadalin, IBM, Mark Nottingham,
1443 BEA, David Orchard, BEA, Jamie Roots, IBM, Shivajee Samdarshi, TIBCO Software, John
1444 Shewchuk, Microsoft, Tony Storey, IBM

1445 The following individuals have provided invaluable input into the initial contribution:

1446 Keith Ballinger, Microsoft, Stefan Batres, Microsoft, Allen Brown, Microsoft, Michael Conner, IBM,
1447 George Copeland, Microsoft, Francisco Curbera, IBM, Paul Fremantle, IBM, Steve Graham, IBM,
1448 Pat Helland, Microsoft, Rick Hill, Microsoft, Scott Hinkelman, IBM, Tim Holloway, IBM, Efim Hudis,
1449 Microsoft, Gopal Kakivaya, Microsoft, Johannes Klein, Microsoft, Frank Leymann, IBM, Martin
1450 Nally, IBM, Peter Niblett, IBM, Jeffrey Schlimmer, Microsoft, James Snell, IBM, Keith Stobie,
1451 Microsoft, Satish Thatte, Microsoft, Stephen Todd, IBM, Sanjiva Weerawarana, IBM, Roger
1452 Wolter, Microsoft

1453 The following individuals were members of the committee during the development of this
1454 specification:

1455 TBD

1456 Appendix E.Revision History

Rev	Date	By Whom	What
wd-01	2005-07-07	Christopher Ferris	Initial version created based on submission by the authors.
ws-02	2005-07-21	Doug Davis	I011 (PT0S) added
wd-02	2005-08-16	Anish Karmarkar	Trivial editorial changes
ws-03	2005-09-15	Doug Davis	I019 and i028 (CloseSeq) added
wd-05	2005-09-26	Gilbert Pilz	i005 (Source resend of nacks messages when ack already received) added.
wd-05	2005-09-27	Doug Davis	i027 (InOrder delivery assurance spanning multiple sequences) added
wd-05	2005-09-27	Doug Davis	i020 (Semantics of "At most once" Delivery Assurance) added
wd-05	2005-09-27	Doug Davis	i034 (Fault while processing a piggy-backed RM header) added
wd-05	2005-09-27	Doug Davis	i033 (Processing model of NACKs) added
wd-05	2005-09-27	Doug Davis	i031 (AckRequested schema inconsistency) added
wd-05	2005-09-27	Doug Davis	i025 (SeqAck/None) added
wd-05	2005-09-27	Doug Davis	i029 (Remove dependency on WS-Security) added
wd-05	2005-09-27	Doug Davis	i039 (What does 'have a mU attribute' mean) added
wd-05	2005-09-27	Doug Davis	i040 (Change 'optional'/'required' to 'OPTIONAL'/'REQUIRED') added
wd-05	2005-09-30	Anish Karmarkar	i017 (Change NS to http://docs.oasis-open.org/wsrn/200510/)
wd-05	2005-09-30	Anish Karmarkar	i045 (Include SecureConversation as a reference and move it to non-normative citation)
wd-05	2005-09-30	Anish Karmarkar	i046 (change the type of wsrn:FaultCode element)

1457 **Appendix F. Notices**

1458 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1459 that might be claimed to pertain to the implementation or use of the technology described in this
1460 document or the extent to which any license under such rights might or might not be available;
1461 neither does it represent that it has made any effort to identify any such rights. Information on
1462 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1463 website. Copies of claims of rights made available for publication and any assurances of licenses
1464 to be made available, or the result of an attempt made to obtain a general license or permission
1465 for the use of such proprietary rights by implementors or users of this specification, can be
1466 obtained from the OASIS Executive Director.

1467 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1468 applications, or other proprietary rights which may cover technology that may be required to
1469 implement this specification. Please address the information to the OASIS Executive Director.

1470 Copyright (C) OASIS Open (2005). All Rights Reserved.

1471 This document and translations of it may be copied and furnished to others, and derivative works
1472 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1473 published and distributed, in whole or in part, without restriction of any kind, provided that the
1474 above copyright notice and this paragraph are included on all such copies and derivative works.
1475 However, this document itself may not be modified in any way, such as by removing the copyright
1476 notice or references to OASIS, except as needed for the purpose of developing OASIS
1477 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1478 Property Rights document must be followed, or as required to translate it into languages other
1479 than English.

1480 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1481 successors or assigns.

1482 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1483 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1484 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1485 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1486 PARTICULAR PURPOSE.