



1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

3 Working Draft 16, November 20, 2006

4 Document identifier:

5 wsrn-1.1-spec-wd-16

6 Location:

7 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-spec-wd-16.pdf>

8 Editors:

9 Doug Davis, IBM <dug@us.ibm.com>
10 Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>
11 Gilbert Pilz, BEA <gpilz@bea.com>
12 Steve Winkler, SAP <steve.winkler@sap.com>
13 Ümit Yalçınalp, SAP <umit.yalcinalp@sap.com>

14 Contributors:

15 See the Acknowledgments (Appendix E).

16 Abstract:

17 This specification (WS-ReliableMessaging) describes a protocol that allows messages to be transferred
18 reliably between nodes implementing this protocol in the presence of software component, system, or
19 network failures. The protocol is described in this specification in a transport-independent manner
20 allowing it to be implemented using different network technologies. To support interoperable Web
21 services, a SOAP binding is defined within this specification.

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

25 By using the XML [XML], SOAP [SOAP 1.1], [SOAP 1.2] and WSDL [WSDL 1.1] extensibility model,
26 SOAP-based and WSDL-based specifications are designed to be composed with each other to define a
27 rich Web services environment. As such, WS-ReliableMessaging by itself does not define all the features
28 required for a complete messaging solution. WS-ReliableMessaging is a building block that is used in
29 conjunction with other specifications and application-specific protocols to accommodate a wide variety of
30 requirements and scenarios related to the operation of distributed Web services.

31 Status:

32 This document was last revised or approved by the WS-RX on the above date. The level of approval is
33 also listed above. Check the current location noted above for possible later revisions of this document.
34 This document is updated periodically on no particular schedule. Technical Committee members should
35 send comments on this specification to the Technical Committee's email list. Others should send
36 comments to the Technical Committee by using the "Send A Comment" button on the Technical
37 Committee's web page at <http://www.oasis-open.org/committees/ws-rx>. For information on whether any
38 patents have been disclosed that may be essential to implementing this specification, and any offers of
39 patent licensing terms, please refer to the Intellectual Property Rights section of the Technical
40 Committee web page (<http://www.oasis-open.org/committees/ws-rx/ipr.php>). The non-normative errata
41 page for this specification is located at <http://www.oasis-open.org/committees/ws-rx>.

42 Table of Contents

42	1 Introduction.....	4
43	1.1 Notational Conventions.....	4
44	1.2 Namespace.....	5
45	1.3 Conformance.....	5
46	2 Reliable Messaging Model.....	6
47	2.1 Glossary.....	6
48	2.2 Protocol Preconditions.....	7
49	2.3 Protocol Invariants.....	8
50	2.4 Example Message Exchange.....	8
51	3 RM Protocol Elements.....	10
52	3.1 Considerations on the Use of Extensibility Points.....	10
53	3.2 Considerations on the Use of "Piggy-Backing".....	10
54	3.3 Composition with WS-Addressing.....	10
55	3.4 Sequence Creation.....	10
56	3.5 Closing A Sequence.....	15
57	3.6 Sequence Termination.....	16
58	3.7 Sequences.....	18
59	3.8 Request Acknowledgement.....	19
60	3.9 Sequence Acknowledgement.....	20
61	4 Faults.....	23
62	4.1 SequenceFault Element.....	24
63	4.2 Sequence Terminated.....	25
64	4.3 Unknown Sequence.....	25
65	4.4 Invalid Acknowledgement.....	26
66	4.5 Message Number Rollover.....	26
67	4.6 Create Sequence Refused.....	27
68	4.7 Sequence Closed.....	27
69	4.8 WSRM Required.....	28
70	5 Security Threats and Countermeasures.....	29
71	5.1 Threats and Countermeasures.....	29
72	5.1.1 Integrity Threats.....	29
73	5.1.1.1 Countermeasures.....	29
74	5.1.2 Resource Consumption Threats.....	30
75	5.1.2.1 Countermeasures.....	30
76	5.1.3 Sequence Spoofing Threats.....	30
77	5.1.3.1 Sequence Hijacking.....	30
78	5.1.3.2 Countermeasures.....	30

79	5.2 Security Solutions and Technologies.....	31
80	5.2.1 Transport Layer Security.....	31
81	5.2.1.1 Model.....	31
82	5.2.1.2 Countermeasure Implementation.....	32
83	5.2.2 SOAP Message Security.....	33
84	5.2.2.1 Model.....	33
85	5.2.2.2 Countermeasure Implementation.....	33
86	6 Securing Sequences.....	35
87	6.1 Securing Sequences Using WS-Security.....	35
88	6.2 Securing Sequences Using SSL/TLS.....	36
89	7 References.....	38
90	7.1 Normative.....	38
91	7.2 Non-Normative.....	39
92	Appendix A. Schema.....	41
93	Appendix B. WSDL.....	46
94	Appendix C. Message Examples.....	48
95	Appendix C.1 Create Sequence.....	48
96	Appendix C.2 Initial Transmission.....	48
97	Appendix C.3 First Acknowledgement.....	50
98	Appendix C.4 Retransmission.....	50
99	Appendix C.5 Termination.....	51
100	Appendix D. State Tables.....	53
101	Appendix E. Acknowledgments.....	57
102	Appendix F. Revision History.....	58
103	Appendix G. Notices.....	64

104 **1 Introduction**

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

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

114 **1.1 Notational Conventions**

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

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

- 119 • The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- 120 • Characters are appended to elements and attributes to indicate cardinality:
 - 121 ○ "?" (0 or 1)
 - 122 ○ "*" (0 or more)
 - 123 ○ "+" (1 or more)
- 124 • The character "|" is used to indicate a choice between alternatives.
- 125 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group
126 with respect to cardinality or choice.
- 127 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content
128 specified in this document. Additional children elements and/or attributes MAY be added at the
129 indicated extension points but they MUST NOT contradict the semantics of the parent and/or
130 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 131 • XML namespace prefixes (See Section [1.2](#)) are used to indicate the namespace of the element
132 being defined.

133 Elements and Attributes defined by this specification are referred to in the text of this document using
134 XPath 1.0 [[XPATH 1.0](#)] expressions. Extensibility points are referred to using an extended version of this
135 syntax:

- 136 • An element extensibility point is referred to using {any} in place of the element name. This
137 indicates that any element name can be used, from any namespace other than the wsrn:
138 namespace.
- 139 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
140 indicates that any attribute name can be used, from any namespace other than the wsrn:
141 namespace.

142 1.2 Namespace

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

144 <http://docs.oasis-open.org/ws-rx/wsrn/200608>

145 Dereferencing the above URI will produce the Resource Directory Description Language [RDDL 2.0]
146 document that describes this namespace.

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

149 Table 1

Prefix	Namespace
S	(Either SOAP 1.1 or 1.2)
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
wsrn	http://docs.oasis-open.org/ws-rx/wsrn/200608
wsa	http://www.w3.org/2005/08/addressing
wsaw	http://www.w3.org/2006/05/addressing/wsdl
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

150 The normative schema for WS-ReliableMessaging can be found linked from the namespace document
151 that is located at the namespace URI specified above.

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

153 1.3 Conformance

154 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or
155 REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace
156 identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is conformant with
157 this specification.

158 Normative text within this specification takes precedence over normative outlines, which in turn take
159 precedence over the XML Schema [XML Schema Part 1, Part 2] descriptions.

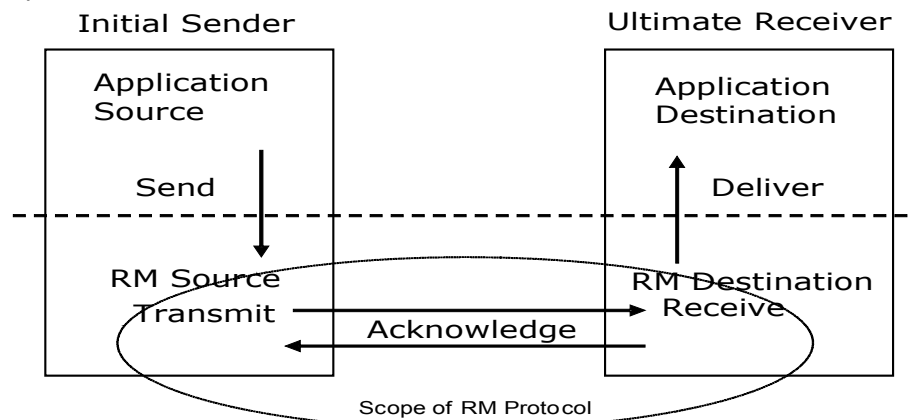
160 2 Reliable Messaging Model

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

163 The WS-ReliableMessaging specification defines an interoperable protocol that enables a Reliable
164 Messaging (RM) Source to accurately determine the disposition of each message it Transmits as
165 perceived by the RM Destination, so as to allow it to resolve any in-doubt status regarding receipt of the
166 message Transmitted. The protocol also enables an RM Destination to efficiently determine which of
167 those messages it Receives have been previously Received, enabling it to filter out duplicate message
168 transmissions caused by the retransmission, by the RM Source, of an unacknowledged message. It also
169 enables an RM Destination to Deliver the messages it Receives to the Application Destination in the order
170 in which they were sent by an Application Source, in the event that they are Received out of order. Note
171 that this specification places no restriction on the scope of the RM Source or RM Destination entities. For
172 example, either can span multiple WSDL Ports or Endpoints.

173 The protocol enables the implementation of a broad range of reliability features which include ordered
174 Delivery, duplicate elimination, and guaranteed receipt. The protocol can also be implemented with a
175 range of robustness characteristics ranging from in-memory persistence that is scoped to a single process
176 lifetime, to replicated durable storage that is recoverable in all but the most extreme circumstances. It is
177 expected that the Endpoints will implement as many or as few of these reliability characteristics as
178 necessary for the correct operation of the application using the protocol. Regardless of which of the
179 reliability features is enabled, the wire protocol does not change.

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



186 Figure 1: Reliable Messaging Model

187 2.1 Glossary

188 The following definitions are used throughout this specification:

189 **Accept:** The act of qualifying a message by the RM Destination such that it becomes eligible for Delivery
190 and acknowledgement.

191 **Acknowledgement:** The communication from the RM Destination to the RM Source indicating the
192 successful receipt of a message.

193 **Acknowledgement Message:** A message containing a `SequenceAcknowledgement` header block.
194 Acknowledgement Messages may or may not contain a SOAP body.

195 **Acknowledgement Request:** A message containing an `AckRequested` header. Acknowledgement
196 Requests may or may not contain a SOAP body.

197 **Application Destination:** The Endpoint to which a message is Delivered.

198 **Application Source:** The Endpoint that Sends a message.

199 **Back-channel:** When the underlying transport provides a mechanism to return a transport-protocol
200 specific response, capable of carrying a SOAP message, without initiating a new connection, this
201 specification refers to this mechanism as a back-channel.

202 **Deliver:** The act of transferring a message from the RM Destination to the Application Destination.

203 **Endpoint:** As defined in the WS-Addressing specification [[WS-Addressing](#)]; a Web service Endpoint is a
204 (referenceable) entity, processor, or resource to which Web service messages can be addressed.
205 Endpoint references (EPRs) convey the information needed to address a Web service Endpoint.

206 **Receive:** The act of reading a message from a network connection and accepting it.

207 **RM Destination:** The Endpoint that Receives messages Transmitted reliably from an RM Source.

208 **RM Protocol Header Block:** One of `Sequence`, `SequenceAcknowledgement`, or `AckRequested`.

209 **RM Source:** The Endpoint that Transmits messages reliably to an RM Destination.

210 **Send:** The act of transferring a message from the Application Source to the RM Source for reliable
211 transfer.

212 **Sequence Lifecycle Message:** A message that contains one of: `CreateSequence`,
213 `CreateSequenceResponse`, `CloseSequence`, `CloseSequenceResponse`, `TerminateSequence`,
214 `TerminateSequenceResponse` as the child element of the SOAP body element.

215 **Sequence Traffic Message:** A message containing a `Sequence` header block.

216 **Transmit:** The act of writing a message to a network connection.

217 **2.2 Protocol Preconditions**

218 The correct operation of the protocol requires that a number of preconditions **MUST** be established prior
219 to the processing of the initial sequenced message:

- 220 • For any single message exchange the RM Source **MUST** have an endpoint reference that uniquely
221 identifies the RM Destination Endpoint.
- 222 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 223 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's
224 policies.
- 225 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**
226 have a security context.

227 2.3 Protocol Invariants

228 During the lifetime of a Sequence, two invariants are REQUIRED for correctness:

- 229 • The RM Source MUST assign each message within a Sequence a message number (defined
- 230 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers
- 231 MUST be assigned in the same order in which messages are sent by the Application Source.
- 232 • Within every Acknowledgement Message it issues, the RM Destination MUST include one or more
- 233 AcknowledgementRange child elements that contain, in their collective ranges, the message
- 234 number of every message accepted by the RM Destination. The RM Destination MUST exclude, in
- 235 the AcknowledgementRange elements, the message numbers of any messages it has not
- 236 accepted. If no messages have been received the RM Destination MUST return None instead of an
- 237 AcknowledgementRange (s). The RM Destination MAY transmit a Nack for a specific message
- 238 or messages in stead of an AcknowledgementRange (s).

239 2.4 Example Message Exchange

240 Figure 2 illustrates a possible message exchange between two reliable messaging Endpoints A and B.

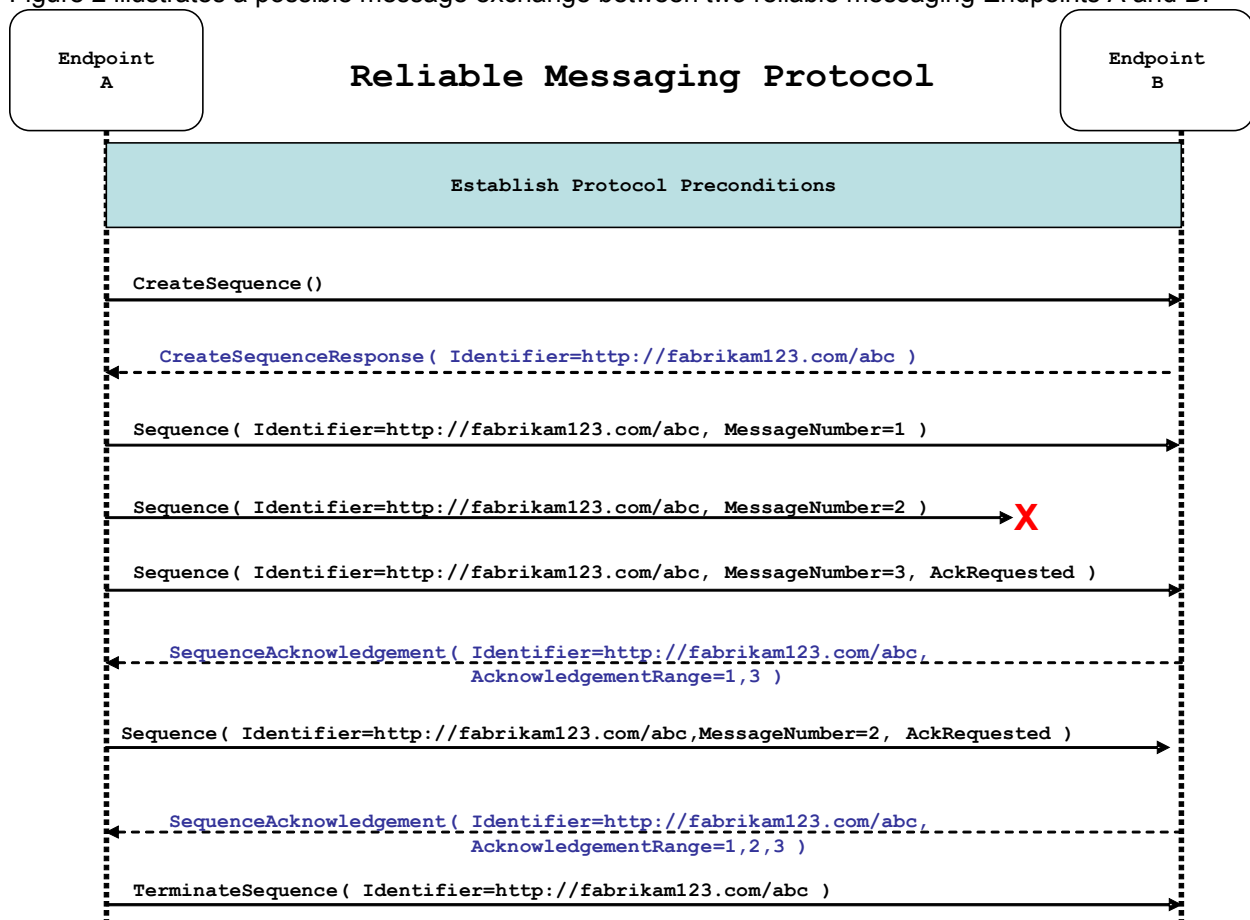


Figure 2: The WS-ReliableMessaging Protocol

- 241 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
- 242 and establishing trust.

- 243 2. The RM Source requests creation of a new Sequence.
- 244 3. The RM Destination creates a new Sequence and returns its unique identifier.
- 245 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1.
246 In the figure above, the RM Source sends 3 messages in the Sequence.
- 247 5. The 2nd message in the Sequence is lost in transit.
- 248 6. The 3rd message is the last in this Sequence and the RM Source includes an `AckRequested`
249 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 250 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the
251 RM Source's `AckRequested` header.
- 252 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new
253 message from the perspective of the underlying transport, but it has the same Sequence Identifier
254 and MessageNumber so the RM Destination can recognize it as a duplicate of the earlier message,
255 in case the original and retransmitted messages are both Received. The RM Source includes an
256 `AckRequested` header in the retransmitted message so the RM Destination will expedite an
257 acknowledgement.
- 258 9. The RM Destination Receives the second transmission of the message with MessageNumber 2
259 and acknowledges receipt of message numbers 1, 2, and 3.
- 260 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to the
261 RM Destination indicating that the Sequence is completed and reclaims any resources associated
262 with the Sequence.
- 263 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source will
264 not be sending any more messages. The RM Destination sends a `TerminateSequenceResponse`
265 message to the RM Source and reclaims any resources associated with the Sequence.

266 The RM Source will expect to Receive Acknowledgements from the RM Destination during the course of a
267 message exchange at occasions described in Section 3 below. Should an Acknowledgement not be
268 Received in a timely fashion, the RM Source MUST re-transmit the message since either the message or
269 the associated Acknowledgement might have been lost. Since the nature and dynamic characteristics of
270 the underlying transport and potential intermediaries are unknown in the general case, the timing of re-
271 transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been
272 demonstrated to cause transport or intermediary flooding which are counterproductive to the intention of
273 providing a reliable exchange of messages. Consequently, implementers are encouraged to utilize
274 adaptive mechanisms that dynamically adjust re-transmission time and the back-off intervals that are
275 appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP
276 transports, a mechanism similar to that described as RTTM in RFC 1323 [[RTTM](#)] SHOULD be
277 considered.

278 Now that the basic model has been outlined, the details of the elements used in this protocol are now
279 provided in Section 3.

280 **3 RM Protocol Elements**

281 The following sub-sections define the various RM protocol elements, and prescribe their usage by a
282 conformant implementations.

283 **3.1 Considerations on the Use of Extensibility Points**

284 The following protocol elements define extensibility points at various places. Implementations MAY add
285 child elements and/or attributes at the indicated extension points but MUST NOT contradict the semantics
286 of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver
287 SHOULD ignore the extension.

288 **3.2 Considerations on the Use of "Piggy-Backing"**

289 Some RM header blocks may be added to messages that are targeted to the same Endpoint to which
290 those headers are to be sent (a concept often referred to as "piggy-backing"), thus saving the overhead of
291 an additional message exchange. Reference parameters MUST be considered when determining whether
292 two EPRs are targeted to the same Endpoint. See the sections that define each RM header block to know
293 which ones may be considered for piggy-backing.

294 **3.3 Composition with WS-Addressing**

295 When the RM protocol, defined in this specification, is composed with the WS-Addressing specification,
296 the following rules prescribe the constraints on the value of the `wsa:Action` header:

- 297 1. When an Endpoint generates a message that carries an RM protocol element, that is defined in
298 section 3 below, in the body of a SOAP envelope that Endpoint MUST include in that envelope a
299 `wsa:Action` SOAP header block whose value is an IRI that is a concatenation of the WS-RM
300 namespace URI, followed by a "/", followed by the value of the local name of the child element of
301 the SOAP body. For example, for a Sequence creation request message as described in section
302 3.4 below, the value of the `wsa:Action` IRI would be:

```
303 http://docs.oasis-open.org/ws-rx/wsrn/200608/CreateSequence
```

- 304 2. When an Endpoint generates an Acknowledgement Message that has no element content in the
305 SOAP body, then the value of the `wsa:Action` IRI MUST be:

```
306 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
```

- 307 3. When an Endpoint generates an Acknowledgement Request that has no element content in the
308 SOAP body, then the value of the `wsa:Action` IRI MUST be:

```
309 http://docs.oasis-open.org/ws-rx/wsrn/200608/AckRequested
```

- 310 4. When an Endpoint generates an RM fault as defined in section 4 below, the value of the
311 `wsa:Action` IRI MUST be as defined in section 4 below.

312 **3.4 Sequence Creation**

313 The RM Source MUST request creation of an outbound Sequence by sending a `CreateSequence`
314 element in the body of a message to the RM Destination which in turn responds either with a message
315 containing `CreateSequenceResponse` or a `CreateSequenceRefused` fault. The RM Source MAY
316 include an offer to create an inbound Sequence within the `CreateSequence` message. This offer is
317 either accepted or rejected by the RM Destination in the `CreateSequenceResponse` message.

318 The SOAP version used for the `CreateSequence` message SHOULD be used for all subsequent
319 messages in or for that Sequence, sent by either the RM Source or the RM Destination.

320 The following exemplar defines the `CreateSequence` syntax:

```
321 <wsrm:CreateSequence ...>
322   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
323   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
324   <wsrm:Offer ...>
325     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
326     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
327     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
328     <wsrm:IncompleteSequenceBehavior>
329       wsrml:IncompleteSequenceBehaviorType
330     </wsrm:IncompleteSequenceBehavior> ?
331     ...
332   </wsrm:Offer> ?
333   ...
334 </wsrm:CreateSequence>
```

335 The following describes the content model of the `CreateSequence` element.

336 `/wsrm:CreateSequence`

337 This element requests creation of a new Sequence between the RM Source that sends it, and the RM
338 Destination to which it is sent. The RM Source MUST NOT send this element as a header block. The RM
339 Destination MUST respond either with a `CreateSequenceResponse` response message or a
340 `CreateSequenceRefused` fault.

341 `/wsrm:CreateSequence/wsrm:AcksTo`

342 The RM Source MUST include this element in any `CreateSequence` message it sends. This element is of
343 type `wsa:EndpointReferenceType` (as specified by WS-Addressing). It specifies the endpoint
344 reference to which messages containing `SequenceAcknowledgement` header blocks and faults related
345 to the created Sequence are to be sent, unless otherwise noted in this specification (for example, see
346 Section 3.5).

347 Implementations MUST NOT use an endpoint reference in the `AcksTo` element that would prevent the
348 sending of Sequence Acknowledgements back to the RM Source. For example, using the WS-Addressing
349 "`http://www.w3.org/2005/08/addressing/none`" IRI would make it impossible for the RM Destination to ever
350 send Sequence Acknowledgements.

351 `/wsrm:CreateSequence/wsrm:Expires`

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

356 `/wsrm:CreateSequence/wsrm:Expires/@{any}`

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

359 `/wsrm:CreateSequence/wsrm:Offer`

360 This element, if present, enables an RM Source to offer a corresponding Sequence for the reliable
361 exchange of messages Transmitted from RM Destination to RM Source.

362 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier`

363 The RM Source MUST set the value of this element to an absolute URI (conformant with RFC3986 [URI])
364 that uniquely identifies the offered Sequence.

365 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier/@{any}

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

368 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint

369 An RM Source MUST include this element, of type `wsa:EndpointReferenceType` (as specified by
370 WS-Addressing). This element specifies the endpoint reference to which Sequence Lifecycle Messages,
371 Sequence Traffic Messages, Acknowledgement Requests, and fault messages related to the offered
372 Sequence are to be sent.

373 Implementations MUST NOT use an endpoint reference in the Endpoint element that would prevent the
374 sending of Sequence Lifecycle Message, Sequence Traffic Message, etc. For example, using the WS-
375 Addressing "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM
376 Destination to ever send Sequence Lifecycle Messages (e.g. `TerminateSequence`) to the RM Source
377 for the Offered Sequence. Implementations MAY use the WS-MakeConnection anonymous URI template
378 and doing so implies that messages will be retrieved using a mechanism such as the `MakeConnection`
379 message.

380 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires

381 This element, if present, of type `xs:duration` specifies the duration for the offered Sequence. A value of
382 "PT0S" indicates that the offered Sequence will never expire. Absence of the element indicates an implied
383 value of "PT0S".

384 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires/@{any}

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

387 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior

388 This element, if present, specifies the behavior that the destination will exhibit upon the closure or
389 termination of an incomplete Sequence. For the purposes of defining the values used, the term "discard"
390 refers to behavior equivalent to the Application Destination never processing a particular message.

391 A value of "DiscardEntireSequence" indicates that the entire Sequence MUST be discarded if the
392 Sequence is closed, or terminated, when there are one or more gaps in the final
393 `SequenceAcknowledgement`.

394 A value of "DiscardFollowingFirstGap" indicates that messages in the Sequence beyond the first gap
395 MUST be discarded when there are one or more gaps in the final `SequenceAcknowledgement`.

396 The default value of "NoDiscard" indicates that no acknowledged messages in the Sequence will be
397 discarded.

398 /wsmr:CreateSequence/wsmr:Offer/{any}

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

401 /wsmr:CreateSequence/wsmr:Offer/@{any}

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

404 /wsmr:CreateSequence/{any}

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

407 /wsmr:CreateSequence/@{any}

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

410 A `CreateSequenceResponse` is sent in the body of a response message by an RM Destination in
411 response to receipt of a `CreateSequence` request message. It carries the `Identifier` of the created
412 Sequence and indicates that the RM Source can begin sending messages in the context of the identified
413 Sequence.

414 The following exemplar defines the `CreateSequenceResponse` syntax:

```
415 <wsmr:CreateSequenceResponse ...>  
416   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
417   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
418   <wsmr:IncompleteSequenceBehavior>  
419     wsmr:IncompleteSequenceBehaviorType  
420   </wsmr:IncompleteSequenceBehavior> ?  
421   <wsmr:Accept ...>  
422     <wsmr:AcksTo wsa:EndpointReferenceType </wsmr:AcksTo>  
423     ...  
424   </wsmr:Accept> ?  
425   ...  
426 </wsmr:CreateSequenceResponse>
```

427 The following describes the content model of the `CreateSequenceResponse` element.

428 /wsmr:CreateSequenceResponse

429 This element is sent in the body of the response message in response to a `CreateSequence` request
430 message. It indicates that the RM Destination has created a new Sequence at the request of the RM
431 Source. The RM Destination MUST NOT send this element as a header block.

432 /wsmr:CreateSequenceResponse/wsmr:Identifier

433 The RM Destination MUST include this element within any `CreateSequenceResponse` message it sends.
434 The RM Destination MUST set the value of this element to the absolute URI (conformant with RFC3986)
435 that uniquely identifies the Sequence that has been created by the RM Destination.

436 /wsmr:CreateSequenceResponse/wsmr:Identifier/@{any}

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

439 /wsmr:CreateSequenceResponse/wsmr:Expires

440 This element, if present, of type `xs:duration` accepts or refines the RM Source's requested duration for
441 the Sequence. It specifies the amount of time after which any resources associated with the Sequence
442 SHOULD be reclaimed thus causing the Sequence to be silently terminated. At the RM Destination this
443 duration is measured from a point proximate to Sequence creation and at the RM Source this duration is
444 measured from a point approximate to the successful processing of the `CreateSequenceResponse`. A
445 value of "PT0S" indicates that the Sequence will never expire. Absence of the element indicates an
446 implied value of "PT0S". The RM Destination MUST set the value of this element to be equal to or less
447 than the value requested by the RM Source in the corresponding `CreateSequence` message.

448 /wsmr:CreateSequenceResponse/wsmr:Expires/@{any}

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

451 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

452 This element, if present, specifies the behavior that the destination will exhibit upon the closure or
453 termination of an incomplete Sequence. For the purposes of defining the values used, the term "discard"
454 refers to behavior equivalent to the Application Destination never processing a particular message.

455 A value of "DiscardEntireSequence" indicates that the entire Sequence MUST be discarded if the
456 Sequence is closed, or terminated, when there are one or more gaps in the final
457 `SequenceAcknowledgement`.

458 A value of "DiscardFollowingFirstGap" indicates that messages in the Sequence beyond the first gap
459 MUST be discarded when there are one or more gaps in the final `SequenceAcknowledgement`.

460 The default value of "NoDiscard" indicates that no acknowledged messages in the Sequence will be
461 discarded.

462 `/wsrm:CreateSequenceResponse/wsrm:Accept`

463 This element, if present, enables an RM Destination to accept the offer of a corresponding Sequence for
464 the reliable exchange of messages Transmitted from RM Destination to RM Source.

465 **Note:** If a `CreateSequenceResponse` is returned without a child `Accept` in response to a
466 `CreateSequence` that did contain a child `Offer`, then the RM Source MAY immediately reclaim any
467 resources associated with the unused offered Sequence.

468 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

469 The RM Destination MUST include this element, of type `wsa:EndpointReferenceType` (as specified
470 by WS-Addressing). It specifies the endpoint reference to which messages containing
471 `SequenceAcknowledgement` header blocks and faults related to the created Sequence are to be sent,
472 unless otherwise noted in this specification (for example, see Section 3.5).

473 Implementations MUST NOT use an endpoint reference in the `AcksTo` element that would prevent the
474 sending of Sequence Acknowledgements back to the RM Source. For example, using the WS-Addressing
475 "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM Destination to ever
476 send Sequence Acknowledgements.

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

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

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

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

483 `/wsrm:CreateSequenceResponse/{any}`

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

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

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

489 3.5 Closing A Sequence

490 There are times during the use of an RM Sequence that the RM Source or RM Destination will wish to
491 discontinue using a Sequence. Simply terminating the Sequence discards the state managed by the RM
492 Destination, leaving the RM Source unaware of the final ranges of messages that were successfully
493 transferred to the RM Destination. To ensure that the Sequence ends with a known final state either the
494 RM Source or RM Destination MAY choose to close the Sequence before terminating it.

495 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of
496 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept
497 any new messages for the specified Sequence, other than those already accepted at the time the
498 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or
499 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST
500 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`
501 element) header block on any messages associated with the Sequence destined to the RM Source,
502 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM
503 Source.

504 If the RM Destination decides to close a Sequence of its own volition, it MAY inform the RM Source of this
505 event by sending a `CloseSequence` element, in the body of a message, to the `AcksTo` EPR of that
506 Sequence. The RM Destination MUST include a final `SequenceAcknowledgement` (within which the RM
507 Destination MUST include the `Final` element) header block in this message and any subsequent
508 messages associated with the Sequence destined to the RM Source.

509 While the RM Destination MUST NOT accept any new messages for the specified Sequence it MUST still
510 process Sequence Lifecycle Messages and Acknowledgement Requests. For example, it MUST respond to
511 `AckRequested`, `TerminateSequence` as well as `CloseSequence` messages. Note, subsequent
512 `CloseSequence` messages have no effect on the state of the Sequence.

513 In the case where the RM Destination wishes to discontinue use of a Sequence it is RECOMMENDED
514 that it close the Sequence. Please see `Final` and the `SequenceClosed` fault. Whenever possible the
515 `SequenceClosed` fault SHOULD be used in place of the `SequenceTerminated` fault to allow the RM
516 Source to still Receive Acknowledgements.

517 The following exemplar defines the `CloseSequence` syntax:

```
518 <wsmr:CloseSequence ...>  
519   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
520   ...  
521 </wsmr:CloseSequence>
```

522 The following describes the content model of the `CloseSequence` element.

523 `/wsmr:CloseSequence`

524 This element MAY be sent by an RM Source to indicate that the RM Destination MUST NOT accept any
525 new messages for this Sequence. This element MAY also be sent by an RM Destination to indicate that it
526 will not accept any new messages for this Sequence.

527 `/wsmr:CloseSequence/wsmr:Identifier`

528 The RM Source or RM Destination MUST include this element in any `CloseSequence` messages it sends.
529 The RM Source or RM Destination MUST set the value of this element to the absolute URI (conformant
530 with RFC3986) of the Sequence that is being closed.

531 `/wsmr:CloseSequence/wsmr:Identifier/@{any}`

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

534 /wsmr:CloseSequence/{any}

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

537 /wsmr:CloseSequence@{any}

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

540 A `CloseSequenceResponse` is sent in the body of a message in response to receipt of a
541 `CloseSequence` request message. It indicates that the responding party has closed the Sequence.

542 The following exemplar defines the `CloseSequenceResponse` syntax:

```
543 <wsmr:CloseSequenceResponse ...>  
544   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
545   ...  
546 </wsmr:CloseSequenceResponse>
```

547 The following describes the content model of the `CloseSequenceResponse` element.

548 /wsmr:CloseSequenceResponse

549 This element is sent in the body of a message in response to receipt of a `CloseSequence` request
550 message. It indicates that the responding party has closed the Sequence.

551 /wsmr:CloseSequenceResponse/wsmr:Identifier

552 The responding party (RMS or RMD) MUST include this element in any `CloseSequenceResponse`
553 message it sends. The responding party MUST set the value of this element to the absolute URI
554 (conformant with RFC3986) of the Sequence that is being closed.

555 /wsmr:CloseSequenceResponse/wsmr:Identifier/@{any}

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

558 /wsmr:CloseSequenceResponse/{any}

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

561 /wsmr:CloseSequenceResponse@{any}

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

564 3.6 Sequence Termination

565 When the RM Source has completed its use of the Sequence it sends a `TerminateSequence` element,
566 in the body of a message, to the RM Destination to indicate that the Sequence is complete and that it will
567 not be sending any further messages related to the Sequence. The RM Destination can safely reclaim any
568 resources associated with the Sequence upon receipt of the `TerminateSequence` message. Under
569 normal usage the RM Source will complete its use of the Sequence when all of the messages in the
570 Sequence have been acknowledged. However, the RM Source is free to Terminate or Close a Sequence
571 at any time regardless of the acknowledgement state of the messages.

572 If the RM Destination decides to terminate a Sequence of its own volition, it MAY inform the RM Source of
573 this event by sending a `TerminateSequence` element, in the body of a message, to the AcksTo EPR for
574 that Sequence. The RM Destination MUST include a final `SequenceAcknowledgement` (within which
575 the RM Destination MUST include the `Final` element) header block in this message.

576 The following exemplar defines the `TerminateSequence` syntax:

```
577 <wsm:TerminateSequence ...>  
578   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
579   ...  
580 </wsm:TerminateSequence>
```

581 The following describes the content model of the `TerminateSequence` element.

582 `/wsm:TerminateSequence`

583 This element MAY be sent by an RM Source to indicate it has completed its use of the Sequence. It
584 indicates that the RM Destination can safely reclaim any resources related to the identified Sequence. The
585 RM Source MUST NOT send this element as a header block. The RM Source MAY retransmit this
586 element. Once this element is sent, other than this element, the RM Source MUST NOT send any
587 additional message to the RM Destination referencing this Sequence.

588 This element MAY also be sent by the RM Destination to indicate that it has unilaterally terminated the
589 Sequence. Upon sending this message the RM Destination MUST NOT accept any additional messages
590 (with the exception of the corresponding `TerminateSequenceResponse`) for this Sequence. Upon
591 receipt of a `TerminateSequence` the RM Source MUST NOT send any additional messages (with the
592 exception of the corresponding `TerminateSequenceResponse`) for this Sequence.

593 `/wsm:TerminateSequence/wsm:Identifier`

594 The RM Source or RM Destination MUST include this element in any `TerminateSequence` message it
595 sends. The RM Source or RM Destination MUST set the value of this element to the absolute URI
596 (conformant with RFC3986) of the Sequence that is being terminated.

597 `/wsm:TerminateSequence/wsm:Identifier/@{any}`

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

600 `/wsm:TerminateSequence/{any}`

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

603 `/wsm:TerminateSequence/@{any}`

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

606 A `TerminateSequenceResponse` is sent in the body of a message in response to receipt of a
607 `TerminateSequence` request message. It indicates that the responding party has terminated the
608 Sequence.

609 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
610 <wsm:TerminateSequenceResponse ...>  
611   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
612   ...  
613 </wsm:TerminateSequenceResponse>
```

614 The following describes the content model of the `TerminateSequence` element.

615 `/wsrm:TerminateSequenceResponse`

616 This element is sent in the body of a message in response to receipt of a `TerminateSequence` request
617 message. It indicates that the responding party has terminated the Sequence. The responding party
618 MUST NOT send this element as a header block.

619 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

620 The responding party (RMS or RMD) MUST include this element in any `TerminateSequenceResponse`
621 message it sends. The responding party MUST set the value of this element to the absolute URI
622 (conformant with RFC3986) of the Sequence that is being terminated.

623 `/wsrm:TerminateSequenceResponse/wsrm:Identifier/@{any}`

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

626 `/wsrm:TerminateSequenceResponse/{any}`

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

629 `/wsrm:TerminateSequenceResponse/@{any}`

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

632 On receipt of a `TerminateSequence` message the receiving party (RMS or RMD) MUST respond with a
633 corresponding `TerminateSequenceResponse` message or generate a fault `UnknownSequenceFault`
634 if the Sequence is not known.

635 **3.7 Sequences**

636 The RM protocol uses a Sequence header block to track and manage the reliable transfer of messages.
637 The RM Source MUST include a `Sequence` header block in all messages for which reliable transfer is
638 REQUIRED. The RM Source MUST identify Sequences with unique Identifier elements and the RM
639 Source MUST assign each message within a Sequence a `MessageNumber` element that increments by 1
640 from an initial value of 1. These values are contained within a `Sequence` header block accompanying
641 each message being transferred in the context of a Sequence.

642 The RM Source MUST NOT include more than one `Sequence` header block in any message.

643 A following exemplar defines its syntax:

```
644 <wsrm:Sequence ...>  
645   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
646   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
647   ...  
648 </wsrm:Sequence>
```

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

650 `/wsrm:Sequence`

651 This protocol element associates the message in which it is contained with a previously established RM
652 Sequence. It contains the Sequence's unique identifier and the containing message's ordinal position
653 within that Sequence. The RM Destination MUST understand the `Sequence` header block. The RM
654 Source MUST assign a `mustUnderstand` attribute with a value 1/true (from the namespace

655 corresponding to the version of SOAP to which the `Sequence` SOAP header block is bound) to the
656 `Sequence` header block element.

657 `/wsmr:Sequence/wsmr:Identifier`

658 An RM Source that includes a `Sequence` header block in a SOAP envelope MUST include this element in
659 that header block. The RM Source MUST set the value of this element to the absolute URI (conformant
660 with RFC3986) that uniquely identifies the `Sequence`.

661 `/wsmr:Sequence/wsmr:Identifier/@{any}`

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

664 `/wsmr:Sequence/wsmr:MessageNumber`

665 The RM Source MUST include this element within any `Sequence` headers it creates. This element is of
666 type `MessageNumberType`. It represents the ordinal position of the message within a `Sequence`.
667 `Sequence` message numbers start at 1 and monotonically increase by 1 throughout the `Sequence`. See
668 Section 4.5 for `Message Number Rollover` fault.

669 `/wsmr:Sequence/{any}`

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

672 `/wsmr:Sequence/@{any}`

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

675 The following example illustrates a `Sequence` header block.

```
676 <wsmr:Sequence>  
677   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
678   <wsmr:MessageNumber>10</wsmr:MessageNumber>  
679 </wsmr:Sequence>
```

680 **3.8 Request Acknowledgement**

681 The purpose of the `AckRequested` header block is to signal to the RM Destination that the RM Source is
682 requesting that a `SequenceAcknowledgement` be sent.

683 The RM Source MAY request an `Acknowledgement Message` from the RM Destination at any time by
684 transmitting an `AckRequested` header block independently or it MAY include an `AckRequested` header
685 block in any message targeted to the RM Destination. An RM Destination that Receives a message that
686 contains an `AckRequested` header block MUST send a message containing a
687 `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference (see Section 3.4) for a
688 known `Sequence` or else generate an `UnknownSequence` fault. If a non-mustUnderstand fault occurs
689 when processing an RM header that was piggy-backed on another message, a fault MUST be generated,
690 but the processing of the original message MUST NOT be affected. It is RECOMMENDED that the RM
691 Destination return a `AcknowledgementRange` or `None` element instead of a `Nack` element (see Section
692 3.9).

693 The following exemplar defines its syntax:

```
694 <wsmr:AckRequested ...>  
695   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
696   ...
```

```
697 </wsm:AckRequested>
```

698 The following describes the content model of the `AckRequested` header block.

```
699 /wsm:AckRequested
```

700 This element requests an Acknowledgement for the identified Sequence.

```
701 /wsm:AckRequested/wsm:Identifier
```

702 An RM Source that includes an `AckRequested` header block in a SOAP envelope MUST include this
703 element in that header block. The RM Source MUST set the value of this element to the absolute URI,
704 (conformant with RFC3986), that uniquely identifies the Sequence to which the request applies.

```
705 /wsm:AckRequested/wsm:Identifier/@{any}
```

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

```
708 /wsm:AckRequested/{any}
```

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

```
711 /wsm:AckRequested/@{any}
```

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

714 3.9 Sequence Acknowledgement

715 The RM Destination informs the RM Source of successful message receipt using a
716 `SequenceAcknowledgement` header block. The RM Destination MAY Transmit the
717 `SequenceAcknowledgement` header block independently or it MAY include the
718 `SequenceAcknowledgement` header block on any message targeted to the `AcksTo` EPR.
719 Acknowledgements can be explicitly requested using the `AckRequested` directive (see Section 3.8). If a
720 non-mustUnderstand fault occurs when processing an RM header that was piggy-backed on another
721 message, a fault MUST be generated, but the processing of the original message MUST NOT be
722 affected.

723 A RM Destination MAY include a `SequenceAcknowledgement` header block on any SOAP envelope
724 targeted to the endpoint referenced by the `AcksTo` EPR.

725 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
726 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
727 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
728 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
729 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
730 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
731 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`
732 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination
733 SHOULD respond on the protocol binding-specific back-channel provided by the Received message
734 containing the `AckRequested` header block.

735 The following exemplar defines its syntax:

```
736 <wsm:SequenceAcknowledgement ...>  
737   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>
```

```

738 [ [ [ <wsm:AcknowledgementRange ...
739         Upper="wsm:MessageType"
740         Lower="wsm:MessageType"/> +
741         | <wsm:None/> ]
742         <wsm:Final/> ? ]
743         | <wsm:Nack> wsm:MessageType </wsm:Nack> + ]
744
745     ...
746 </wsm:SequenceAcknowledgement>

```

747 The following describes the content model of the `SequenceAcknowledgement` header block.

748 `/wsm:SequenceAcknowledgement`

749 This element contains the Sequence Acknowledgement information.

750 `/wsm:SequenceAcknowledgement/wsm:Identifier`

751 An RM Destination that includes a `SequenceAcknowledgement` header block in a SOAP envelope
752 MUST include this element in that header block. The RM Destination MUST set the value of this element
753 to the absolute URI (conformant with RFC3986) that uniquely identifies the Sequence. The RM
754 Destination MUST NOT include multiple `SequenceAcknowledgement` header blocks that share the
755 same value for `Identifier` within the same SOAP envelope.

756 `/wsm:SequenceAcknowledgement/wsm:Identifier/@{any}`

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

759 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange`

760 The RM Destination MAY include one or more instances of this element within a
761 `SequenceAcknowledgement` header block. It contains a range of Sequence message numbers
762 successfully accepted by the RM Destination. The ranges MUST NOT overlap. The RM Destination
763 MUST NOT include this element if a sibling `Nack` or `None` element is also present as a child of
764 `SequenceAcknowledgement`.

765 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@Upper`

766 The RM Destination MUST set the value of this attribute equal to the message number of the highest
767 contiguous message in a Sequence range accepted by the RM Destination.

768 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@Lower`

769 The RM Destination MUST set the value of this attribute equal to the message number of the lowest
770 contiguous message in a Sequence range accepted by the RM Destination.

771 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@{any}`

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

774 `/wsm:SequenceAcknowledgement/wsm:None`

775 The RM Destination MUST include this element within a `SequenceAcknowledgement` header block if
776 the RM Destination has not accepted any messages for the specified Sequence. The RM Destination
777 MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack` element is also present
778 as a child of the `SequenceAcknowledgement`.

779 `/wsm:SequenceAcknowledgement/wsm:Final`

780 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. This
781 element indicates that the RM Destination is not receiving new messages for the specified Sequence. The
782 RM Source can be assured that the ranges of messages acknowledged by this
783 `SequenceAcknowledgement` header block will not change in the future. The RM Destination MUST
784 include this element when the Sequence is closed. The RM Destination MUST NOT include this element
785 when sending a `Nack`; it can only be used when sending `AcknowledgementRange` elements or a `None`.

786 `/wsrm:SequenceAcknowledgement/wsrm:Nack`

787 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If
788 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing
789 the `MessageNumber` of an unreceived message in a Sequence. The RM Destination MUST NOT include
790 a `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of
791 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the
792 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`
793 containing a `Nack` for a message that it has previously acknowledged within a
794 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing
795 a `Nack` for a message that has previously been acknowledged within a `AcknowledgementRange`.

796 `/wsrm:SequenceAcknowledgement/{any}`

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

799 `/wsrm:SequenceAcknowledgement/@{any}`

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

802 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
804 <wsrm:SequenceAcknowledgement>  
805   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
806   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
807 </wsrm:SequenceAcknowledgement>
```

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

```
810 <wsrm:SequenceAcknowledgement>  
811   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
812   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
813   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
814   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
815 </wsrm:SequenceAcknowledgement>
```

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

```
817 <wsrm:SequenceAcknowledgement>  
818   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
819   <wsrm:Nack>3</wsrm:Nack>  
820 </wsrm:SequenceAcknowledgement>
```

821 4 Faults

822 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. Create
823 Sequence Refused is a possible fault reply for this operation. Unknown Sequence is a fault generated by
824 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences
825 are detected. WSRM Required is a fault generated an RM Destination that requires the use of WS-RM on
826 a Received message that did not use the protocol. All other faults in this section relate to known
827 Sequences. Destinations that generate faults related to known sequences SHOULD transmit those faults.
828 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement
829 messages.

830 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault
831 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

832 `http://docs.oasis-open.org/ws-rx/wsrn/200608/fault`

833 The faults defined in this section are generated if the condition stated in the preamble is met. Fault
834 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

835 The definitions of faults use the following properties:

836 [Code] The fault code.

837 [Subcode] The fault subcode.

838 [Reason] The English language reason element.

839 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail
840 element is defined for a fault, implementations MUST include the elements in the order that they are
841 specified.

842 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
843 "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

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

```
845 <S:Envelope>
846   <S:Header>
847     <wsa:Action>
848       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault
849     </wsa:Action>
850     <!-- Headers elided for brevity. -->
851   </S:Header>
852   <S:Body>
853     <S:Fault>
854       <S:Code>
855         <S:Value> [Code] </S:Value>
856         <S:Subcode>
857           <S:Value> [Subcode] </S:Value>
858         </S:Subcode>
859       </S:Code>
860       <S:Reason>
861         <S:Text xml:lang="en"> [Reason] </S:Text>
862       </S:Reason>
863     <S:Detail>
```

```

864     [Detail]
865     ...
866     </S:Detail>
867     </S:Fault>
868     </S:Body>
869     </S:Envelope>

```

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

```

872 <S11:Envelope>
873   <S11:Header>
874     <wsrm:SequenceFault>
875       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
876       <wsrm:Detail> [Detail] </wsrm:Detail>
877       ...
878     </wsrm:SequenceFault>
879     <!-- Headers elided for brevity. -->
880   </S11:Header>
881   <S11:Body>
882     <S11:Fault>
883       <faultcode> [Code] </faultcode>
884       <faultstring> [Reason] </faultstring>
885     </S11:Fault>
886   </S11:Body>
887 </S11:Envelope>

```

888 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
889 `CreateSequence` request message:

```

890 <S11:Envelope>
891   <S11:Body>
892     <S11:Fault>
893       <faultcode> [Subcode] </faultcode>
894       <faultstring> [Reason] </faultstring>
895     </S11:Fault>
896   </S11:Body>
897 </S11:Envelope>

```

898 4.1 SequenceFault Element

899 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during
900 the reliable messaging specific processing of a message belonging to a Sequence. WS-
901 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP
902 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in
903 conjunction with the SOAP 1.2 binding.

904 The following exemplar defines its syntax:

```

905 <wsrm:SequenceFault ...>
906   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
907   <wsrm:Detail> ... </wsrm:Detail> ?
908   ...
909 </wsrm:SequenceFault>

```

910 The following describes the content model of the `SequenceFault` element.

911 `/wsrm:SequenceFault`

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

913 /wsm:SequenceFault/wsm:FaultCode
 914 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a
 915 qualified name from the set of fault [Subcodes] defined below.

916 /wsm:SequenceFault/wsm:Detail
 917 This element, if present, carries application specific error information related to the fault being described.

918 /wsm:SequenceFault/wsm:Detail/{any}
 919 The application specific error information related to the fault being described.

920 /wsm:SequenceFault/wsm:Detail/@{any}
 921 The application specific error information related to the fault being described.

922 /wsm:SequenceFault/{any}
 923 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
 924 to be passed.

925 /wsm:SequenceFault/@{any}
 926 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
 927 element.

928 4.2 Sequence Terminated

929 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 930 Endpoint of this decision.

931 Properties:

932 [Code] Sender or Receiver
 933 [Subcode] wsm:SequenceTerminated
 934 [Reason] The Sequence has been terminated due to an unrecoverable error.
 935 [Detail]

936 `<wsm:Identifier ...> xs:anyURI </wsm:Identifier>`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source or RM Destination.	Encountering an unrecoverable condition or detection of violation of the protocol.	Sequence termination.	MUST terminate the Sequence if not otherwise terminated.

937 4.3 Unknown Sequence

938 Properties:

939 [Code] Sender
 940 [Subcode] wsm:UnknownSequence

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

942 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source or RM Destination.	In response to a message containing an unknown or terminated Sequence identifier.	None.	MUST terminate the Sequence if not otherwise terminated.

944 **4.4 Invalid Acknowledgement**

945 An example of when this fault is generated is when a message is Received by the RM Source containing
946 a SequenceAcknowledgement covering messages that have not been sent.

947 [Code] Sender

948 [Subcode] wsrn:InvalidAcknowledgement

949 [Reason] The SequenceAcknowledgement violates the cumulative Acknowledgement invariant.

950 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source.	In response to a SequenceAcknowledgement that violate the invariants stated in 2.3 or any of the requirements in 3.9 about valid combinations of AckRange, Nack and None in a single SequenceAcknowledgement element or with respect to already Received such elements.	Unspecified.	Unspecified.

952 **4.5 Message Number Rollover**

953 If the condition listed below is reached, the RM Destination MUST generate this fault.

954 Properties:

955 [Code] Sender

956 [Subcode] wsrn:MessageNumberRollover

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

958 [Detail]

```
959 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
960 <wsrm:MaxMessageNumber> wsrm:MessageNumberType </wsrm:MaxMessageNumber>
```

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	Message number in /wsrm:Sequence/wsrm:MessageNumber of a Received message exceeds the internal limitations of an RM Destination or reaches the maximum value of 9,223,372,036,854,775,807.	RM Destination SHOULD continue to accept undelivered messages until the Sequence is closed or terminated.	RM Source SHOULD continue to retransmit undelivered messages until the Sequence is closed or terminated.

961 4.6 Create Sequence Refused

962 Properties:

963 [Code] Sender or Receiver

964 [Subcode] wsrm:CreateSequenceRefused

965 [Reason] The Create Sequence request has been refused by the RM Destination.

966 [Detail]

```
967 xs:any
```

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	In response to a CreateSequence message when the RM Destination does not wish to create a new Sequence.	Unspecified.	Sequence terminated.

968 4.7 Sequence Closed

969 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

970 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
971 is closed.

972 Properties:

973 [Code] Sender

974 [Subcode] wsrm:SequenceClosed

975 [Reason] The Sequence is closed and can not accept new messages.

976 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	In response to a message that belongs to a Sequence that is already closed.	Unspecified.	Sequence closed.

978 **4.8 WSRM Required**

979 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
980 message that did not use this protocol.

981 Properties:

982 [Code] Sender

983 [Subcode] wsrm:WSRMRequired

984 [Reason] The RM Destination requires the use of WSRM.

985 [Detail]

986 `xs:any`

987 **5 Security Threats and Countermeasures**

988 This specification considers two sets of security requirements, those of the applications that use the WS-
989 RM protocol and those of the protocol itself.

990 This specification makes no assumptions about the security requirements of the applications that use WS-
991 RM. However, once those requirements have been satisfied within a given operational context, the
992 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
993 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

994 There are many other security concerns that one may need to consider when implementing or using this
995 protocol. The material below should not be considered as a "check list". Implementers and users of this
996 protocol are urged to perform a security analysis to determine their particular threat profile and the
997 appropriate responses to those threats.

998 Implementers are also advised that there is a core tension between security and reliable messaging that
999 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1000 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1001 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1002 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1003 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1004 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1005 avoid and prevent this condition.

1006 **5.1 Threats and Countermeasures**

1007 The primary security requirement of this protocol is to protect the specified semantics and protocol
1008 invariants against various threats. The following sections describe several threats to the integrity and
1009 operation of this protocol and provide some general outlines of countermeasures to those threats.
1010 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1011 to all operational contexts.

1012 **5.1.1 Integrity Threats**

1013 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1014 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1015 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1016 to its intended message represents a threat to the WS-RM protocol.

1017 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM
1018 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1019 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be
1020 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1021 **5.1.1.1 Countermeasures**

1022 Integrity threats are generally countered via the use of digital signatures some level of the communication
1023 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1024 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers
1025 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which
1026 they occur, implementations MUST allow for signatures that cover only these headers.

1027 **5.1.2 Resource Consumption Threats**

1028 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1029 implement that RM Destination. These resources can include network connections, database tables,
1030 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1031 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1032 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1033 message Delivery and use this Sequence to send a stream of very large messages to that service,
1034 making sure to omit message number “1” from that stream.

1035 **5.1.2.1 Countermeasures**

1036 There are a number of countermeasures against the described resource consumption threats. The
1037 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1038 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1039 some cases, allows the identity of any attackers to be determined.

1040 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and
1041 authenticate the RM Source that issued the `CreateSequence` message.

1042 **5.1.3 Sequence Spoofing Threats**

1043 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1044 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1045 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1046 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the
1047 current `MessageNumber` for their target Sequence.

1048 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP
1049 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier
1050 to attack the Sequence. These attacks are “two-way” in that an attacker may choose to target the RM
1051 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends
1052 to the `AcksTo` EPR of an RM Source.

1053 **5.1.3.1 Sequence Hijacking**

1054 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject
1055 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1056 messages.

1057 Note that “sequence hijacking” should not be equated with “security session hijacking”. Although a
1058 Sequence may be bound to some form of a security session in order to counter the threats described in
1059 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1060 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1061 established by the security infrastructure to make such determinations. Failure to observe this rule
1062 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1063 the ability to authenticate its peers even though the necessary security processing has taken place.

1064 **5.1.3.2 Countermeasures**

1065 There are a number of countermeasures against sequence spoofing threats. The technique advocated by
1066 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1067 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1068 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1069 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1070 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1071 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1072 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1073 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1074 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1075 sequence peer it MUST be able to identify and authenticate the entity that sent the
1076 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1077 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1078 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1079 creation time.

1080 **5.2 Security Solutions and Technologies**

1081 The security threats described in the previous sections are neither new nor unique. The solutions that
1082 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1083 section maps the facilities provided by common web services security solutions against countermeasures
1084 described in the previous sections.

1085 Before continuing this discussion, however, some examination of the underlying requirements of the
1086 previously described countermeasures is necessary. Specifically it should be noted that the technique
1087 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1088 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check
1089 against this authenticated identity and determines if the RM Source is permitted to create Sequences with
1090 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,
1091 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of
1092 such facilities is considered to be beyond the scope of this specification.

1093 **5.2.1 Transport Layer Security**

1094 This section describes how the facilities provided by SSL/TLS [RFC 4346] can be used to implement the
1095 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1096 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1097 The description provided here is general in nature and is not intended to serve as a complete definition on
1098 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1099 choice of features as well as the manner in which they will be used. The mechanisms described in the
1100 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the
1101 requirements and constraints of the use of SSL/TLS.

1102 **5.2.1.1 Model**

1103 The basic model for using SSL/TLS is as follows:

- 1104 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1105 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1106 Destination.

- 1107 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1108 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1109 synchronous `CreateSequenceResponse` using the session established in (1).
- 1110 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1111 any and all messages or faults that refer to that Sequence.
- 1112 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1113 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1114 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1115 5.2.1.2 Countermeasure Implementation

1116 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1117 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1118 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1119 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1120 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1121 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1122 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1123 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1124 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1125 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1126 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1127 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1128 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1129 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1130 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1131 Acknowledgement) using BasicAuth.
- 1132 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1133 connection authenticates itself to the party accepting the connection using an X.509 certificate
1134 that is exchanged during the SSL/TLS handshake.

1135 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1136 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1137 Source is authorized to create a Sequence with the RM Destination.

1138 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1139 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1140 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1141 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1142 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1143 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1144 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1145 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1146 to protect that Sequence.

1147 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1148 countermeasures (such as associating specific authentication information with a Sequence) although such
1149 methods are not covered by this document.

1150 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1151 session) are outside the scope of this specification.

1152 **5.2.2 SOAP Message Security**

1153 The mechanisms described in WS-Security may be used in various ways to implement the
1154 countermeasures described in the previous sections. This specification advocates using the protocol
1155 described by WS-SecureConversation [[SecureConversation](#)] (optionally in conjunction with WS-Trust
1156 [[Trust](#)]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1157 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1158 The description provided here is general in nature and is not intended to serve as a complete definition on
1159 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1160 need to agree on the choice of features as well as the manner in which they will be used. The
1161 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1162 describe the requirements and constraints of the use of WS-SecureConversation.

1163 **5.2.2.1 Model**

1164 The basic model for using WS-SecureConversation is as follows:

- 1165 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This
1166 may involve the participation of third parties such as a security token service. The tokens
1167 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1168 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1169 context that will be used to protect the Sequence. This is done so that, in cases where the
1170 `CreateSequence` message is signed by more than one security context, the RM Source can
1171 indicate which security context should be used to protect the newly created Sequence.
- 1172 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1173 associated with the security context to sign (as defined by WS-Security) at least the body and any
1174 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

1175 **5.2.2.2 Countermeasure Implementation**

1176 Without relying upon any authentication information, the per-message signatures provide the necessary
1177 integrity qualities to counter the threats described in Section 5.1.1.

1178 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1179 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1180 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1181 create a Sequence with the RM Destination.

1182 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1183 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1184 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1185 context rather than on any authentication claims that may have been established during security context
1186 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1187 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1188 document.

1189 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer
1190 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1191 the association between a Sequence and its protecting security context cannot always be established
1192 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1193 `CreateSequenceResponse` messages may be signed by more than one security context.

1194 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1195 amending or renewing contexts) are outside the scope of this specification.

1196 6 Securing Sequences

1197 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared
1198 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1199 achieving this objective depending upon the underlying security infrastructure.

1200 6.1 Securing Sequences Using WS-Security

1201 One mechanism for protecting a Sequence is to include a security token using a
1202 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1203 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1204 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1205 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1206 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1207 there may be more than one token on a `CreateSequence` message or inferred from the communication
1208 context (e.g. transport protection).

1209 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1210 if the token being referenced supports such mechanism.

1211 The following exemplar defines the `CreateSequence` syntax when extended to include a
1212 `wsse:SecurityTokenReference`:

```
1213 <wsrm:CreateSequence ...>  
1214   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1215   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1216   <wsrm:Offer ...>  
1217     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1218     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1219     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1220     <wsrm:IncompleteSequenceBehavior>  
1221       wsrml:IncompleteSequenceBehaviorType  
1222     </wsrm:IncompleteSequenceBehavior> ?  
1223     ...  
1224   </wsrm:Offer> ?  
1225   ...  
1226   <wsse:SecurityTokenReference>  
1227     ...  
1228   </wsse:SecurityTokenReference> ?  
1229   ...  
1230 </wsrm:CreateSequence>
```

1231 The following describes the content model of the additional `CreateSequence` elements.

1232 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1233 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in
1234 section 3.4) to communicate an explicit reference to the security token, using a
1235 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and Destination
1236 MUST use to authorize messages for the created (and, if present, the offered) Sequence(s). All
1237 subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST
1238 demonstrate proof-of-possession of the secret associated with the token (e.g., by using or deriving from a
1239 private or secret key).

1240 When a RM Source transmits a `CreateSequence` that has been extended to include a
1241 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and
1242 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include

1243 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1244 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1245 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1246 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1247 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1248 in WS-Security still applies.

1249 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1250 <wsm:UsesSequenceSTR ... />
```

1251 The following describes the content model of the `UsesSequenceSTR` header block.

1252 `/wsm:UsesSequenceSTR`

1253 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1254 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1255 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1256 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1257 Sequence creation.

1258 The following is an example of a `CreateSequence` message using the

1259 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1260 <soap:Envelope ...>  
1261   <soap:Header>  
1262     ...  
1263     <wsm:UsesSequenceSTR soap:mustUnderstand='true' />  
1264     ...  
1265   </soap:Header>  
1266   <soap:Body>  
1267     <wsm:CreateSequence>  
1268       <wsm:AcksTo>  
1269         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1270       </wsm:AcksTo>  
1271       <wsse:SecurityTokenReference>  
1272         ...  
1273       </wsse:SecurityTokenReference>  
1274     </wsm:CreateSequence>  
1275   </soap:Body>  
1276 </soap:Envelope>
```

1277 6.2 Securing Sequences Using SSL/TLS

1278 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1279 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1280 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1281 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1282 SOAP header block within the `CreateSequence` message.

1283 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1284 <wsm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1285 The following describes the content model of the `UsesSequenceSSL` header block.

1286 `/wsm:UsesSequenceSSL`

1287 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1288 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was

1289 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a
1290 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand
1291 and correctly implement the functionality described in Section 5.2.1 or else generate a
1292 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1293 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1294 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1295 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1296 `CreateSequenceResponse` message.

1297 **7 References**

1298 **7.1 Normative**

1299 **[KEYWORDS]**

1300 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,
1301 March 1997

1302 <http://www.ietf.org/rfc/rfc2119.txt>

1303 **[WS-RM Policy]**

1304 OASIS WS-RX Technical Committee Draft, "[Web Services ReliableMessaging Policy Assertion\(WS-RM
1305 Policy\)](#)" October 2006

1306 <http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-wd-11.pdf>

1307 **[SOAP 1.1]**

1308 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

1309 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1310 **[SOAP 1.2]**

1311 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

1312 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

1313 **[URI]**

1314 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,
1315 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1316 <http://ietf.org/rfc/rfc3986>

1317 **[UUID]**

1318 P. Leach, M. Mealling, R. Salz, "[A Universally Unique Identifier \(UUID\) URN Namespace](#)," RFC 4122,
1319 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1320 <http://www.ietf.org/rfc/rfc4122.txt>

1321 **[XML]**

1322 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#)", September 2006.

1323 <http://www.w3.org/TR/REC-xml/>

1324 **[XML-ns]**

1325 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

1326 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

1327 **[XML-Schema Part1]**

1328 W3C Recommendation, "[XML Schema Part 1: Structures](#)," October 2004.

1329 <http://www.w3.org/TR/xmlschema-1/>

1330 **[XML-Schema Part2]**

1331 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

1332 <http://www.w3.org/TR/xmlschema-2/>

1333 **[XPath 1.0]**

1334 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

1335 <http://www.w3.org/TR/xpath>

1336 **[WSDL 1.1]**

1337 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

1338 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1339 **[WS-Addressing]**

1340 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

1341 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

1342 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

1343 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1344 **7.2 Non-Normative**

1345 **[BSP 1.0]**

1346 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

1347 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

1348 **[RDDL 2.0]**

1349 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

1350 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

1351 **[RFC 2617]**

1352 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
1353 Authentication: Basic and Digest Access Authentication," June 1999.

1354 <http://www.ietf.org/rfc/rfc2617.txt>

1355 **[RFC 4346]**

1356 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

1357 <http://www.ietf.org/rfc/rfc4346.txt>

1358 **[WS-Policy]**

1359 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

1360 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

1361 **[WS-PolicyAttachment]**

1362 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

1363 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-
1364 20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1365 **[WS-Security]**

1366 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1367 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1368 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

1369 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1370 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

1371 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

1372 **[RTTM]**

1373 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May
1374 1992.

1375 <http://www.rfc-editor.org/rfc/rfc1323.txt>

1376 **[SecurityPolicy]**

1377 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

1378 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

1379 **[SecureConversation]**

1380 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February
1381 2005.

1382 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

1383 **[Trust]**

1384 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

1385 <http://schemas.xmlsoap.org/ws/2005/02/trust>

1386 Appendix A. Schema

1387 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1388 Schema Part2] is located at:

1389 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

1390 The following copy is provided for reference.

```
1391 <?xml version="1.0" encoding="UTF-8"?>
1392 <!--
1393 OASIS takes no position regarding the validity or scope of any intellectual
1394 property or other rights that might be claimed to pertain to the
1395 implementation or use of the technology described in this document or the
1396 extent to which any license under such rights might or might not be available;
1397 neither does it represent that it has made any effort to identify any such
1398 rights. Information on OASIS's procedures with respect to rights in OASIS
1399 specifications can be found at the OASIS website. Copies of claims of rights
1400 made available for publication and any assurances of licenses to be made
1401 available, or the result of an attempt made to obtain a general license or
1402 permission for the use of such proprietary rights by implementors or users of
1403 this specification, can be obtained from the OASIS Executive Director.
1404 OASIS invites any interested party to bring to its attention any copyrights,
1405 patents or patent applications, or other proprietary rights which may cover
1406 technology that may be required to implement this specification. Please
1407 address the information to the OASIS Executive Director.
1408 Copyright © OASIS Open 2002-2006. All Rights Reserved.
1409 This document and translations of it may be copied and furnished to others,
1410 and derivative works that comment on or otherwise explain it or assist in its
1411 implementation may be prepared, copied, published and distributed, in whole or
1412 in part, without restriction of any kind, provided that the above copyright
1413 notice and this paragraph are included on all such copies and derivative
1414 works. However, this document itself does not be modified in any way, such as
1415 by removing the copyright notice or references to OASIS, except as needed for
1416 the purpose of developing OASIS specifications, in which case the procedures
1417 for copyrights defined in the OASIS Intellectual Property Rights document must
1418 be followed, or as required to translate it into languages other than English.
1419 The limited permissions granted above are perpetual and will not be revoked by
1420 OASIS or its successors or assigns.
1421 This document and the information contained herein is provided on an "AS IS"
1422 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1423 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1424 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1425 FOR A PARTICULAR PURPOSE.
1426 -->
1427 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1428 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1429 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1430 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1431 elementFormDefault="qualified" attributeFormDefault="unqualified">
1432   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1433     schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1434   <!-- Protocol Elements -->
1435   <xs:complexType name="SequenceType">
1436     <xs:sequence>
1437       <xs:element ref="wsrm:Identifier"/>
1438       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1439       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1440 maxOccurs="unbounded"/>
1441     </xs:sequence>
```

```

1442     <xs:anyAttribute namespace="##other" processContents="lax"/>
1443 </xs:complexType>
1444 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1445 <xs:element name="SequenceAcknowledgement">
1446   <xs:complexType>
1447     <xs:sequence>
1448       <xs:element ref="wsrm:Identifier"/>
1449       <xs:choice>
1450         <xs:sequence>
1451           <xs:choice>
1452             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1453               <xs:complexType>
1454                 <xs:sequence/>
1455                 <xs:attribute name="Upper" type="xs:unsignedLong"
1456 use="required"/>
1457                 <xs:attribute name="Lower" type="xs:unsignedLong"
1458 use="required"/>
1459               <xs:anyAttribute namespace="##other" processContents="lax"/>
1460             </xs:complexType>
1461           </xs:element>
1462           <xs:element name="None">
1463             <xs:complexType>
1464               <xs:sequence/>
1465             </xs:complexType>
1466           </xs:element>
1467         </xs:choice>
1468         <xs:element name="Final" minOccurs="0">
1469           <xs:complexType>
1470             <xs:sequence/>
1471           </xs:complexType>
1472         </xs:element>
1473       </xs:sequence>
1474       <xs:element name="Nack" type="xs:unsignedLong"
1475 maxOccurs="unbounded"/>
1476     </xs:choice>
1477     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1478 maxOccurs="unbounded"/>
1479   </xs:sequence>
1480   <xs:anyAttribute namespace="##other" processContents="lax"/>
1481 </xs:complexType>
1482 </xs:element>
1483 <xs:complexType name="AckRequestedType">
1484   <xs:sequence>
1485     <xs:element ref="wsrm:Identifier"/>
1486     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1487 maxOccurs="unbounded"/>
1488   </xs:sequence>
1489   <xs:anyAttribute namespace="##other" processContents="lax"/>
1490 </xs:complexType>
1491 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1492 <xs:element name="Identifier">
1493   <xs:complexType>
1494     <xs:annotation>
1495       <xs:documentation>
1496         This type is for elements whose [children] is an anyURI and can have
1497 arbitrary attributes.
1498       </xs:documentation>
1499     </xs:annotation>
1500     <xs:simpleContent>
1501       <xs:extension base="xs:anyURI">
1502         <xs:anyAttribute namespace="##other" processContents="lax"/>
1503       </xs:extension>
1504     </xs:simpleContent>

```

```

1505     </xs:complexType>
1506 </xs:element>
1507 <xs:element name="Address">
1508   <xs:complexType>
1509     <xs:simpleContent>
1510       <xs:extension base="xs:anyURI">
1511         <xs:anyAttribute namespace="##other" processContents="lax"/>
1512       </xs:extension>
1513     </xs:simpleContent>
1514   </xs:complexType>
1515 </xs:element>
1516 <xs:simpleType name="MessageNumberType">
1517   <xs:restriction base="xs:unsignedLong">
1518     <xs:minInclusive value="1"/>
1519     <xs:maxInclusive value="9223372036854775807"/>
1520   </xs:restriction>
1521 </xs:simpleType>
1522 <!-- Fault Container and Codes -->
1523 <xs:simpleType name="FaultCodes">
1524   <xs:restriction base="xs:QName">
1525     <xs:enumeration value="wsrm:SequenceTerminated"/>
1526     <xs:enumeration value="wsrm:UnknownSequence"/>
1527     <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1528     <xs:enumeration value="wsrm:MessageNumberRollover"/>
1529     <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1530     <xs:enumeration value="wsrm:SequenceClosed"/>
1531     <xs:enumeration value="wsrm:WSRMRequired"/>
1532     <xs:enumeration value="wsrm:UnsupportedSelection"/>
1533   </xs:restriction>
1534 </xs:simpleType>
1535 <xs:complexType name="SequenceFaultType">
1536   <xs:sequence>
1537     <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1538     <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1539     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1540 maxOccurs="unbounded"/>
1541   </xs:sequence>
1542   <xs:anyAttribute namespace="##other" processContents="lax"/>
1543 </xs:complexType>
1544 <xs:complexType name="DetailType">
1545   <xs:sequence>
1546     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1547 maxOccurs="unbounded"/>
1548   </xs:sequence>
1549   <xs:anyAttribute namespace="##other" processContents="lax"/>
1550 </xs:complexType>
1551 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1552 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1553 <xs:element name="CreateSequenceResponse"
1554 type="wsrm:CreateSequenceResponseType"/>
1555 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1556 <xs:element name="CloseSequenceResponse"
1557 type="wsrm:CloseSequenceResponseType"/>
1558 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1559 <xs:element name="TerminateSequenceResponse"
1560 type="wsrm:TerminateSequenceResponseType"/>
1561 <xs:complexType name="CreateSequenceType">
1562   <xs:sequence>
1563     <xs:element ref="wsrm:AcksTo"/>
1564     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1565     <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1566     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1567 maxOccurs="unbounded"/>

```

```

1568     <xs:annotation>
1569         <xs:documentation>
1570             It is the authors intent that this extensibility be used to
1571 transfer a Security Token Reference as defined in WS-Security.
1572         </xs:documentation>
1573     </xs:annotation>
1574 </xs:any>
1575 </xs:sequence>
1576 <xs:anyAttribute namespace="##other" processContents="lax"/>
1577 </xs:complexType>
1578 <xs:complexType name="CreateSequenceResponseType">
1579     <xs:sequence>
1580         <xs:element ref="wsrm:Identifier"/>
1581         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1582         <xs:element name="IncompleteSequenceBehavior"
1583 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1584         <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1585         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1586 maxOccurs="unbounded"/>
1587     </xs:sequence>
1588     <xs:anyAttribute namespace="##other" processContents="lax"/>
1589 </xs:complexType>
1590 <xs:complexType name="CloseSequenceType">
1591     <xs:sequence>
1592         <xs:element ref="wsrm:Identifier"/>
1593         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1594 maxOccurs="unbounded"/>
1595     </xs:sequence>
1596     <xs:anyAttribute namespace="##other" processContents="lax"/>
1597 </xs:complexType>
1598 <xs:complexType name="CloseSequenceResponseType">
1599     <xs:sequence>
1600         <xs:element ref="wsrm:Identifier"/>
1601         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1602 maxOccurs="unbounded"/>
1603     </xs:sequence>
1604     <xs:anyAttribute namespace="##other" processContents="lax"/>
1605 </xs:complexType>
1606 <xs:complexType name="TerminateSequenceType">
1607     <xs:sequence>
1608         <xs:element ref="wsrm:Identifier"/>
1609         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1610 maxOccurs="unbounded"/>
1611     </xs:sequence>
1612     <xs:anyAttribute namespace="##other" processContents="lax"/>
1613 </xs:complexType>
1614 <xs:complexType name="TerminateSequenceResponseType">
1615     <xs:sequence>
1616         <xs:element ref="wsrm:Identifier"/>
1617         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1618 maxOccurs="unbounded"/>
1619     </xs:sequence>
1620     <xs:anyAttribute namespace="##other" processContents="lax"/>
1621 </xs:complexType>
1622 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1623 <xs:complexType name="OfferType">
1624     <xs:sequence>
1625         <xs:element ref="wsrm:Identifier"/>
1626         <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1627         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1628         <xs:element name="IncompleteSequenceBehavior"
1629 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1630         <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1631 maxOccurs="unbounded"/>
1632     </xs:sequence>
1633     <xs:anyAttribute namespace="##other" processContents="lax"/>
1634 </xs:complexType>
1635 <xs:complexType name="AcceptType">
1636     <xs:sequence>
1637         <xs:element ref="wsrm:AcksTo"/>
1638         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1639 maxOccurs="unbounded"/>
1640     </xs:sequence>
1641     <xs:anyAttribute namespace="##other" processContents="lax"/>
1642 </xs:complexType>
1643 <xs:element name="Expires">
1644     <xs:complexType>
1645         <xs:simpleContent>
1646             <xs:extension base="xs:duration">
1647                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1648             </xs:extension>
1649         </xs:simpleContent>
1650     </xs:complexType>
1651 </xs:element>
1652 <xs:simpleType name="IncompleteSequenceBehaviorType">
1653     <xs:restriction base="xs:string">
1654         <xs:enumeration value="DiscardEntireSequence"/>
1655         <xs:enumeration value="DiscardFollowingFirstGap"/>
1656         <xs:enumeration value="NoDiscard"/>
1657     </xs:restriction>
1658 </xs:simpleType>
1659 <xs:element name="UsesSequenceSTR">
1660     <xs:complexType>
1661         <xs:sequence/>
1662         <xs:anyAttribute namespace="##other" processContents="lax"/>
1663     </xs:complexType>
1664 </xs:element>
1665 <xs:element name="UsesSequenceSSL">
1666     <xs:complexType>
1667         <xs:sequence/>
1668         <xs:anyAttribute namespace="##other" processContents="lax"/>
1669     </xs:complexType>
1670 </xs:element>
1671 <xs:element name="UnsupportedElement">
1672     <xs:simpleType>
1673         <xs:restriction base="xs:QName"/>
1674     </xs:simpleType>
1675 </xs:element>
1676 </xs:schema>

```

1677 Appendix B. WSDL

1678 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1679 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1680 present in exchanges with that endpoint.

1681 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1682 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1683 for a higher-level mechanism to indicate that WS-RM is engaged.

1684 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1685 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

1686 The following non-normative copy is provided for reference.

```
1687 <?xml version="1.0" encoding="utf-8"?>
1688 <!--
1689 OASIS takes no position regarding the validity or scope of any intellectual
1690 property or other rights that might be claimed to pertain to the
1691 implementation or use of the technology described in this document or the
1692 extent to which any license under such rights might or might not be available;
1693 neither does it represent that it has made any effort to identify any such
1694 rights. Information on OASIS's procedures with respect to rights in OASIS
1695 specifications can be found at the OASIS website. Copies of claims of rights
1696 made available for publication and any assurances of licenses to be made
1697 available, or the result of an attempt made to obtain a general license or
1698 permission for the use of such proprietary rights by implementors or users of
1699 this specification, can be obtained from the OASIS Executive Director.
1700 OASIS invites any interested party to bring to its attention any copyrights,
1701 patents or patent applications, or other proprietary rights which may cover
1702 technology that may be required to implement this specification. Please
1703 address the information to the OASIS Executive Director.
1704 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
1705 This document and translations of it may be copied and furnished to others,
1706 and derivative works that comment on or otherwise explain it or assist in its
1707 implementation may be prepared, copied, published and distributed, in whole or
1708 in part, without restriction of any kind, provided that the above copyright
1709 notice and this paragraph are included on all such copies and derivative
1710 works. However, this document itself does not be modified in any way, such as
1711 by removing the copyright notice or references to OASIS, except as needed for
1712 the purpose of developing OASIS specifications, in which case the procedures
1713 for copyrights defined in the OASIS Intellectual Property Rights document must
1714 be followed, or as required to translate it into languages other than English.
1715 The limited permissions granted above are perpetual and will not be revoked by
1716 OASIS or its successors or assigns.
1717 This document and the information contained herein is provided on an "AS IS"
1718 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1719 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1720 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1721 FOR A PARTICULAR PURPOSE.
1722 -->
1723 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1724 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1725 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
1726 open.org/ws-rx/wsrn/200608" xmlns:tns="http://docs.oasis-open.org/ws-
1727 rx/wsrn/200608/wsd" targetNamespace="http://docs.oasis-open.org/ws-
1728 rx/wsrn/200608/wsd">
1729 <wsdl:types>
```

```

1730     <xs:schema>
1731         <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsr/200608"
1732 schemaLocation="http://docs.oasis-open.org/ws-rx/wsr/200608/wsr-1.1-schema-
1733 200608.xsd"/>
1734     </xs:schema>
1735 </wsdl:types>

1736     <wsdl:message name="CreateSequence">
1737         <wsdl:part name="create" element="rm:CreateSequence"/>
1738     </wsdl:message>
1739     <wsdl:message name="CreateSequenceResponse">
1740         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1741     </wsdl:message>
1742     <wsdl:message name="CloseSequence">
1743         <wsdl:part name="close" element="rm:CloseSequence"/>
1744     </wsdl:message>
1745     <wsdl:message name="CloseSequenceResponse">
1746         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1747     </wsdl:message>
1748     <wsdl:message name="TerminateSequence">
1749         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1750     </wsdl:message>
1751     <wsdl:message name="TerminateSequenceResponse">
1752         <wsdl:part name="terminateResponse"
1753 element="rm:TerminateSequenceResponse"/>
1754     </wsdl:message>

1755     <wsdl:portType name="SequenceAbstractPortType">
1756         <wsdl:operation name="CreateSequence">
1757             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1758 open.org/ws-rx/wsr/200608/CreateSequence"/>
1759             <wsdl:output message="tns:CreateSequenceResponse"
1760 wsaw:Action="http://docs.oasis-open.org/ws-
1761 rx/wsr/200608/CreateSequenceResponse"/>
1762         </wsdl:operation>
1763         <wsdl:operation name="CloseSequence">
1764             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1765 open.org/ws-rx/wsr/200608/CloseSequence"/>
1766             <wsdl:output message="tns:CloseSequenceResponse"
1767 wsaw:Action="http://docs.oasis-open.org/ws-
1768 rx/wsr/200608/CloseSequenceResponse"/>
1769         </wsdl:operation>
1770         <wsdl:operation name="TerminateSequence">
1771             <wsdl:input message="tns:TerminateSequence"
1772 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsr/200608/TerminateSequence"/>
1773             <wsdl:output message="tns:TerminateSequenceResponse"
1774 wsaw:Action="http://docs.oasis-open.org/ws-
1775 rx/wsr/200608/TerminateSequenceResponse"/>
1776         </wsdl:operation>
1777     </wsdl:portType>

1778 </wsdl:definitions>

```

1779 Appendix C. Message Examples

1780 Appendix C.1 Create Sequence

1781 Create Sequence

```
1782 <?xml version="1.0" encoding="UTF-8"?>
1783 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1784 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1785 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1786   <S:Header>
1787     <wsa:MessageID>
1788       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1789     </wsa:MessageID>
1790     <wsa:To>http://example.com/serviceB/123</wsa:To>
1791     <wsa:Action>http://docs.oasis-open.org/ws-
1792 rx/wsmr/200608/CreateSequence</wsa:Action>
1793     <wsa:ReplyTo>
1794       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1795     </wsa:ReplyTo>
1796   </S:Header>
1797   <S:Body>
1798     <wsmr:CreateSequence>
1799       <wsmr:AcksTo>
1800         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1801       </wsmr:AcksTo>
1802     </wsmr:CreateSequence>
1803   </S:Body>
1804 </S:Envelope>
```

1805 Create Sequence Response

```
1806 <?xml version="1.0" encoding="UTF-8"?>
1807 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1808 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1809 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1810   <S:Header>
1811     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1812     <wsa:RelatesTo>
1813       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1814     </wsa:RelatesTo>
1815     <wsa:Action>
1816       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
1817     </wsa:Action>
1818   </S:Header>
1819   <S:Body>
1820     <wsmr:CreateSequenceResponse>
1821       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1822     </wsmr:CreateSequenceResponse>
1823   </S:Body>
1824 </S:Envelope>
```

1825 Appendix C.2 Initial Transmission

1826 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1827 figure. The three messages have the following headers; the third message is identified as the last
1828 message in the Sequence:

1829 **Message 1**

```
1830 <?xml version="1.0" encoding="UTF-8"?>
1831 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1832 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1833 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1834   <S:Header>
1835     <wsa:MessageID>
1836       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1837     </wsa:MessageID>
1838     <wsa:To>http://example.com/serviceB/123</wsa:To>
1839     <wsa:From>
1840       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1841     </wsa:From>
1842     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1843     <wsmr:Sequence>
1844       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1845       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1846     </wsmr:Sequence>
1847   </S:Header>
1848   <S:Body>
1849     <!-- Some Application Data -->
1850   </S:Body>
1851 </S:Envelope>
```

1852 **Message 2**

```
1853 <?xml version="1.0" encoding="UTF-8"?>
1854 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1855 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1856 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1857   <S:Header>
1858     <wsa:MessageID>
1859       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1860     </wsa:MessageID>
1861     <wsa:To>http://example.com/serviceB/123</wsa:To>
1862     <wsa:From>
1863       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1864     </wsa:From>
1865     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1866     <wsmr:Sequence>
1867       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1868       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1869     </wsmr:Sequence>
1870   </S:Header>
1871   <S:Body>
1872     <!-- Some Application Data -->
1873   </S:Body>
1874 </S:Envelope>
```

1875 **Message 3**

```
1876 <?xml version="1.0" encoding="UTF-8"?>
1877 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1878 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1879 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1880   <S:Header>
1881     <wsa:MessageID>
1882       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1883     </wsa:MessageID>
1884     <wsa:To>http://example.com/serviceB/123</wsa:To>
1885     <wsa:From>
1886       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
```

```

1887 </wsa:From>
1888 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1889 <wsrm:Sequence>
1890 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1891 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1892 </wsrm:Sequence>
1893 <wsrm:AckRequested>
1894 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1895 </wsrm:AckRequested>
1896 </S:Header>
1897 <S:Body>
1898 <!-- Some Application Data -->
1899 </S:Body>
1900 </S:Envelope>

```

1901 **Appendix C.3 First Acknowledgement**

1902 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1903 responds with an Acknowledgement for messages 1 and 3:

```

1904 <?xml version="1.0" encoding="UTF-8"?>
1905 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1906 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
1907 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1908 <S:Header>
1909 <wsa:MessageID>
1910 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1911 </wsa:MessageID>
1912 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1913 <wsa:From>
1914 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1915 </wsa:From>
1916 <wsa:Action>
1917 http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
1918 </wsa:Action>
1919 <wsrm:SequenceAcknowledgement>
1920 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1921 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1922 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1923 </wsrm:SequenceAcknowledgement>
1924 </S:Header>
1925 <S:Body/>
1926 </S:Envelope>

```

1927 **Appendix C.4 Retransmission**

1928 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
1929 requests an Acknowledgement:

```

1930 <?xml version="1.0" encoding="UTF-8"?>
1931 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1932 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
1933 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1934 <S:Header>
1935 <wsa:MessageID>
1936 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1937 </wsa:MessageID>
1938 <wsa:To>http://example.com/serviceB/123</wsa:To>
1939 <wsa:From>
1940 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1941 </wsa:From>

```

```

1942 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1943 <wsrm:Sequence>
1944 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1945 <wsrm:MessageNumber>2</wsrm:MessageNumber>
1946 </wsrm:Sequence>
1947 <wsrm:AckRequested>
1948 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1949 </wsrm:AckRequested>
1950 </S:Header>
1951 <S:Body>
1952 <!-- Some Application Data -->
1953 </S:Body>
1954 </S:Envelope>

```

1955 Appendix C.5 Termination

1956 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
1957 be terminated:

```

1958 <?xml version="1.0" encoding="UTF-8"?>
1959 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1960 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1961 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1962 <S:Header>
1963 <wsa:MessageID>
1964 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
1965 </wsa:MessageID>
1966 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1967 <wsa:From>
1968 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1969 </wsa:From>
1970 <wsa:Action>
1971 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
1972 </wsa:Action>
1973 <wsrm:SequenceAcknowledgement>
1974 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1975 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
1976 </wsrm:SequenceAcknowledgement>
1977 </S:Header>
1978 <S:Body/>
1979 </S:Envelope>

```

1980 Terminate Sequence

```

1981 <?xml version="1.0" encoding="UTF-8"?>
1982 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1983 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1984 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1985 <S:Header>
1986 <wsa:MessageID>
1987 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
1988 </wsa:MessageID>
1989 <wsa:To>http://example.com/serviceB/123</wsa:To>
1990 <wsa:Action>
1991 http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence
1992 </wsa:Action>
1993 <wsa:From>
1994 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1995 </wsa:From>
1996 </S:Header>
1997 <S:Body>
1998 <wsrm:TerminateSequence>

```

```
1999     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2000     </wsrm:TerminateSequence>
2001     </S:Body>
2002     </S:Envelope>
```

2003 Terminate Sequence Response

```
2004     <?xml version="1.0" encoding="UTF-8"?>
2005     <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2006     xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrp/200608"
2007     xmlns:wsa="http://www.w3.org/2005/08/addressing">
2008     <S:Header>
2009     <wsa:MessageID>
2010     http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2011     </wsa:MessageID>
2012     <wsa:To>http://example.com/serviceA/789</wsa:To>
2013     <wsa:Action>
2014     http://docs.oasis-open.org/ws-rx/wsrp/200608/TerminateSequenceResponse
2015     </wsa:Action>
2016     <wsa:RelatesTo>
2017     http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2018     </wsa:RelatesTo>
2019     <wsa:From>
2020     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2021     </wsa:From>
2022     </S:Header>
2023     <S:Body>
2024     <wsrm:TerminateSequenceResponse>
2025     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2026     </wsrm:TerminateSequenceResponse>
2027     </S:Body>
2028     </S:Envelope>
```

2029 Appendix D. State Tables

2030 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2031 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2032 Legend:

2033 The first column of these tables contains the motivating event and has the following format:

Event
<i>Event name</i> [source] {ref}

2034 Where:

- 2035 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2036 described by the specification.
- 2037 ● [source]: indicates the source of the event; one of:
 - 2038 ● [msg] a Received message
 - 2039 ● [int]: an internal event such as the firing of a timer
 - 2040 ● [app]: the application
 - 2041 ● [unspec]: the source is unspecified

2042 Each event / state combination cell in the tables in this appendix has the following format:

State Name
<i>Action to take</i> [next state] {ref}

2043 Where:

- 2044 ● action to take: indicates that the state machine performs the following action. Actions surrounded
2045 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2046 "Transmit"
- 2047 ● [next state]: indicates the state to which the state machine will advance upon the performance of
2048 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2049 ● {ref} is a reference to the document section describing the behavior in this cell

2050 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2051 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2052 described in this specification and does not indicate normal protocol operation. Implementations MAY
2053 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2054 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2055 combinations.

2056 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
Create Sequence [unspec] {3.4}	Xmit Create Sequence [Creating] {3.4}	N/A	N/A	N/A	N/A	N/A
Create Sequence Response [msg] {3.4}		Process Create Sequence Response [Created] {3.4}				
Create Sequence Refused Fault [msg] {3.4}		No action [None] {4.6}				
Send message [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [Same] {2}	N/A	N/A
Retransmit of un-ack'd message [int]	N/A	N/A	Xmit message [Same] {2.4}	Xmit message [Same] {2.4}	N/A	N/A
SeqAck (non-final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}
Nack [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	<Xmit message(s)> [Same] {3.9}	<Xmit message(s)> [Same] {3.9}	No action [Same]	No action [Same]
Message Number Rollover Fault [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Rollover]	No action [Same]	No action [Same]	No action [Same]
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}
<Close Sequence> [int] {3.5}	N/A		Xmit Close Sequence [Closing] {3.5}	N/A	N/A	N/A
CloseSequenceResponse [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}		No action [Closed] {3.5}	No action [Same] {3.5}	No action [Same] {3.5}

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
SeqAck (final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Same]	Process Ack ranges [Same]
Sequence Closed Fault [msg] {4.7}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]
Unknown Sequence Fault [msg] {4.3}			Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
Sequence Terminated Fault [msg] {4.2}	N/A		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
TerminateSequence [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}
Terminate Sequence [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	N/A
Terminate Sequence Response [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}				Terminate Sequence [None] {3.6}
Expires exceeded [int]	N/A	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}
Invalid Acknowledgement [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

2056 Table 2 RM Destination Sequence State Transition Table

Events	Sequence States			
	None	Created	Closing	Closed
CreateSequence (successful) [msg/int] {3.4}	Xmit Create Sequence Response [Created] {3.4}	N/A	N/A	N/A

Events	Sequence States			
	None	Created	Closing	Closed
CreateSequence (unsuccessful) [msg/int] {3.4}	Generate Create Sequence Refused Fault [None] {3.4}	N/A		N/A
Message (with message number within range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Accept Message; <Xmit SeqAck> [Same]		Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}
Message (with message number outside of range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.7}{4.5}		Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}
<AckRequested> [msg] {3.8}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.8}	Xmit SeqAck+Final [Same] {3.9}	Xmit SeqAck+Final [Same] {3.9}
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Xmit CloseSequenceResponse with SeqAck+Final [Closed] {3.5}	Xmit CloseSequenceResponse with SeqAck+Final [Closed] {3.5}
<CloseSequence autonomously> [int]	N/A	Xmit CloseSequence with SeqAck+Final [Closing] {3.5}		N/A
CloseSequenceResponse [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}		No Action [Closed] {3.5}	No action [Closed] {3.5}
TerminateSequence [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit TerminateSequence Response [None] {3.6}	Xmit TerminateSequenceResponse [None] {3.6}	Xmit TerminateSequenceResponse [None] {3.6}
<TerminateSequence autonomously> [int]		Xmit Terminate Sequence [None] with SeqAck+Final {3.6}	Xmit Terminate Sequence [None] with SeqAck+Final {3.6}	Xmit Terminate Sequence [None] with SeqAck+Final {3.6}
TerminateSequenceResponse [msg]	Generate Unknown Sequence Fault [Same] {4.3}			
UnknownSequence Fault [msg] {4.3}		Terminate Sequence [None] {4.3}		Terminate Sequence [None] {4.3}
SequenceTerminated Fault [msg] {4.2}		Terminate Sequence [None] {4.2}		Terminate Sequence [None] {4.2}
Invalid Acknowledgement Fault [msg]	N/A			

Events	Sequence States			
	None	Created	Closing	Closed
{4.4}				
Expires exceeded [int] {3.4}	N/A	Terminate Sequence [None] {3.4}		Terminate Sequence [None] {3.4}
<Seq Acknowledgement autonomously> [int] {3.9}	N/A	Xmit SeqAck [Same] {3.9}		Xmit SeqAck+Final [Same] {3.9}
Non WSRM message when WSRM required [msg] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	Generate WSRMRequired Fault [Same] {4.8}		Generate WSRMRequired Fault [Same] {4.8}

2057 **Appendix E. Acknowledgments**

2058 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2059 authors:

2060 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),
2061 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),
2062 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),
2063 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don
2064 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),
2065 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony
2066 Storey(IBM).

2067 The following individuals have provided invaluable input into the initial contribution:

2068 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen
2069 Brown(Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM),
2070 Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott
2071 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal
2072 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter
2073 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish
2074 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2075 The following individuals were members of the committee during the development of this specification:

2076 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben
2077 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd
2078 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul
2079 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques
2080 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),
2081 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair
2082 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),
2083 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul
2084 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt
2085 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff
2086 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku
2087 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert
2088 Pilz(BEA), Martin Raepfle(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom
2089 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von
2090 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçinalp(SAP), Nobuyuki
2091 Yamamoto(Hitachi).

Appendix F. 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 'optiona'/'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)
wd-06	2005-11-02	Gilbert Pilz	Start wd-06 by changing title page from cd-01.
wd-06	2005-11-03	Gilbert Pilz	i047 (Reorder spec sections)
wd-07	2005-11-17	Gilbert Pilz	Start wd-07
wd-07	2005-11-28	Doug Davis	i071 – except for period in Appendix headings
wd-07	2005-11-28	Doug Davis	i10
wd-07	2005-11-28	Doug Davis	i030
wd-07	2005-11-28	Doug Davis	i037
wd-07	2005-11-28	Doug Davis	i038
wd-07	2005-11-28	Doug Davis	i041
wd-07	2005-11-28	Doug Davis	i043
wd-07	2005-11-28	Doug Davis	i044

Rev	Date	By Whom	What
wd-07	2005-11-28	Doug Davis	i048
wd-07	2005-11-28	Doug Davis	i051
wd-07	2005-11-28	Doug Davis	i053
wd-07	2005-11-28	Doug Davis	i059
wd-07	2005-11-28	Doug Davis	i062
wd-07	2005-11-28	Doug Davis	i063
wd-07	2005-11-28	Doug Davis	i065
wd-07	2005-11-28	Doug Davis	i067
wd-07	2005-11-28	Doug Davis	i068
wd-07	2005-11-28	Doug Davis	i069
wd-07	2005-11-28	Doug Davis	Fix bulleted list (#2) in section 2.3
wd-07	2005-11-29	Gilbert Pilz	i074 (Use of [tcShortName] in artifact locations namespaces, etc)
wd-07	2005-11-29	Gilbert Pilz	i071 – Fixed styles and formatting for TOC. Fixed styles of the appendix headings.
wd-07	2005-11-30	Doug Davis	Removed dup definition of "Receive"
wd-07	2005-11-30	Gilbert Pilz	Fixed lost formatting from heading for Namespace section. Fixed style of text body elements to match OASIS example documents. Fixed tables to match OASIS example documents.
wd-07	2005-12-01	Gilbert Pilz	Updated fix for i074 to eliminate trailing '/'. Added corresponding text around action IRI composition.
wd-07	2005-12-01	Gilbert Pilz	Use non-fixed fields for date values on both title page and body footers.
wd-07	2005-12-01	Doug Davis	Alphabetize the glossary
wd-07	2005-12-02	Doug Davis	i064
wd-07	2005-12-02	Doug Davis	i066
wd-08	2005-12-15	Doug Davis	Add back in RM Source to glossary
wd-08	2005-12-15	Steve Winkler	Doug added Steve's editorial nits
wd-08	2005-12-21	Doug Davis	i050
wd-08	2005-12-21	Doug Davis	i081
wd-08	2005-12-21	Doug Davis	i080 – but i050 negates the need for any changes
wd-08	2005-12-21	Doug Davis	i079
wd-08	2005-12-21	Doug Davis	i076 – didn't add text about "replies" since the RMD to RMS sequence could be used for any message not just replies
wd-08	2005-12-21	Umit Yalcinalp	Action Su03: removed wsse from Table 1
wd-08	2005-12-21	Umit Yalcinalp	i057 per Sunnyvale F2F 2005, Cleaned up some formatting errors in contributors
wd-08	2005-12-27	Doug Davis	i060
wd-08	2005-12-27	Gilbert Pilz	Moved schema and WSDL files to their own artifacts. Converted source document to

Rev	Date	By Whom	What
			OpenDocument Text format. Changed line numbers to be a single style.
wd-08	2005-12-28	Anish Karmarkar	Included a section link to c:\temp\wsrm-1.1-schema-200510.xsd and to c:\temp\wsrm-1.1-wsdl-200510.wsdl
wd-08	2006-01-04	Gilbert Pilz	Fixed formatting for included sections.
wd-08	2006-01-05	Gilbert Pilz	Created links for unused references. Fixed exemplars for CloseSequence and CloseSequenceResponse.
wd-09	2006-01-11	Doug Davis	Minor tweaks to text/typos.
wd-10	2006-01-23	Doug Davis	Accept all changes from wd-09 Make some minor editorial tweaks from Marc's comments.
wd-10	2006-02-14	Doug Davis	Issue 082 resolution
wd-10	2006-02-14	Doug Davis	Issue 083 resolution
wd-10	2006-02-14	Doug Davis	Issue 085 resolution
wd-10	2006-02-14	Doug Davis	Issues 086, 087 resolutions Defined MessageNumberType
wd-10	2006-02-15	Doug Davis	Issue 078 resolution
wd-10	2006-02-15	Doug Davis	Issue 094 resolution
wd-10	2006-02-15	Doug Davis	Issue 095 resolution
wd-10	2006-02-15	Gilbert Pilz	Issue 088 – added namespace URI link to namespace URI; added text explaining that this URI could be dereferenced to produce the RDDDL doc; added non-normative reference to RDDDL 2.0
wd-10	2006-02-17	Anish Karmarkar	Namespace changed to 200602 for both WSDL and XSD docs.
wd-10	2006-02-17	Anish Karmarkar	Issue i087 as it applies to WSRM spec.
wd-10	2006-02-17	Anish Karmarkar	Added titles and minor text for state table (issue i058).
wd-11	2006-02-22	Doug Davis	Accept all changes for new WD Minor typos fixed
wd-11	2006-02-23	Doug Davis	s'/close'/close/g – per Marc Goodner Added first ref to [URI] – per Marc G again
wd-11	2006-02-27	Doug Davis	Issue i061 applied
wd-11	2006-02-28	Doug Davis	Fixed typo around the use of "above" and "below"
wd-11	2006-03-01	Doug Davis	Minor typos found by Marc Goodner
wd-11	2006-03-02	Doug Davis	Minor typos found by Matt Lovett
wd-11	2006-03-08	Doug Davis	Issue 091 applied
wd-11	2006-03-08	Doug Davis	Issue 092 applied
wd-11	2006-03-08	Doug Davis	Issue 100 applied

Rev	Date	By Whom	What
wd-12	2006-03-20	Doug Davis	Added space in "SOAP1.x" – PaulCotton
wd-12	2006-04-11	Doug Davis	Issue 007 applied
wd-12	2006-04-11	Doug Davis	Issue 090 applied
wd-12	2006-04-11	Doug Davis	Issue 098 applied
wd-12	2006-04-11	Doug Davis	Issue 099 applied
wd-12	2006-04-11	Doug Davis	Issue 101 applied
wd-12	2006-04-11	Doug Davis	Issue 103 applied
wd-12	2006-04-11	Doug Davis	Issue 104 applied
wd-12	2006-04-11	Doug Davis	Issue 105 applied
wd-12	2006-04-11	Doug Davis	Issue 107 applied
wd-12	2006-04-11	Doug Davis	Issue 109 applied
wd-12	2006-04-11	Doug Davis	Issue 110 applied
wd-12	2006-04-12	Doug Davis	Used "generated" instead of "issue" or "send" when talking about faults.
wd-12	2006-04-24	Gilbert Pilz	Update references to WS-Addressing to the Proposed Recommendations; update WS-RM namespace to "200604".
wd-13	2006-05-08	Gilbert Pilz	i093 part 1; more work needed
wd-13	2006-05-10	Doug Davis	Issue 096 applied
wd-13	2006-05-26	Gilbert Pilz	i093 part 2; reflects decisions from 2006-05-25 meeting
wd-13	2006-05-28	Gilbert Pilz	Issue 106 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 118 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 120 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 114 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 116 applied
wd-14	2006-06-05	Gilbert Pilz	Accept all changes; bump WD number
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Marc Goodner
wd-14	2006-06-07	Doug Davis	Change a couple of period/sp/sp to period/sp
wd-14	2006-06-07	Doug Davis	Added a space in "URI]of" – per Marc Goodner
wd-14	2006-06-07	Doug Davis	Issue 131 applied
wd-14	2006-06-07	Doug Davis	Issue 132 applied
wd-14	2006-06-07	Doug Davis	Issue 119 applied
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Doug Davis
wd-14	2006-06-07	Doug Davis	s/"none"/"full-uri"/ - per Marc Goodner
wd-14	2006-06-12	Doug Davis	Complete i106
wd-14	2006-06-12	Doug Davis	Issues 089 applied
wd-14	2006-06-12	Doug Davis	Fix for several RFC2119 keywords – per Anish
wd-15	2006-06-12	Doug Davis	Accept all changed, dump WD number
wd-15	2006-06-12	Doug Davis	Move WSDL after Schema
wd-15	2006-06-12	Doug Davis	Nits – remove tabs, extra [yyy]'s ...
wd-15	2006-06-14	Doug Davis	Remove extra "OPTIONAL"s – Matt Lovett

Rev	Date	By Whom	What
wd-15	2006-06-14	Doug Davis	Remove blank rows/columns from state table. Fix italics in state table
wd-15	2006-06-15	Doug Davis	Typo – section D was empty
wd-15	2006-06-16	Doug Davis	Issue 125 applied
wd-15	2006-06-16	Doug Davis	Issue 126 applied
wd-15	2006-06-16	Doug Davis	Issue 127 applied
wd-15	2006-06-16	Doug Davis	Issue 133 applied
wd-15	2006-06-16	Doug Davis	Issue 136 applied
wd-15	2006-06-16	Doug Davis	Issue 138 applied
wd-15	2006-06-16	Doug Davis	Issue 135 applied
wd-15	2006-06-20	Doug Davis	Added all TC members to the ack list
wd-15	2006-06-22	Doug Davis	Issue 129 applied
wd-15	2006-06-22	Doug Davis	Issue 130 applied
wd-15	2006-06-22	Doug Davis	Issue 137 applied
wd-15	2006-06-26	Doug Davis	Issue 111 applied
wd-15	2006-06-26	Doug Davis	Missed a part of issue 129
wd-15	2006-06-30	Doug Davis	Fixed a typo in schema
wd-15	2006-06-30	Doug Davis	Issue 141 applied
wd-15	2006-06-30	Doug Davis	Issue 142 applied
wd-15	2006-06-30	Doug Davis	Issue 148 applied
wd-15	2006-06-30	Doug Davis	Issue 149 applied
wd-15	2006-06-30	Doug Davis	Issue 150 applied
wd-15	2006-07-06	Doug Davis	Issue 121 applied
wd-15	2006-07-21	Doug Davis	Issue 139 applied
wd-15	2006-07-21	Doug Davis	Issue 144 applied
wd-15	2006-07-21	Doug Davis	Issue 147 applied
wd-15	2006-07-21	Doug Davis	Issues 122-124 applied
wd-15	2006-07-27	Doug Davis	Updated list of oasis TC members (i134)
wd-15	2006-07-27	Doug Davis	Issue 140 applied
wd-15	2006-07-27	Doug Davis	Issue 145 applied
wd-15	2006-07-27	Doug Davis	Issue 143 applied
wd-15	2006-07-28	Doug Davis	Lots of minor typos found by Matt L.
wd-15	2006-07-28	Doug Davis	Issue 113 applied
wd-15	2006-08-04	Doug Davis	Update old namespaces – found by PaulC
wd-15	2006-08-04	Doug Davis	Issue 150 applied
wd-15	2006-08-04	Doug Davis	Minor typos – found by PeterN
wd-15	2006-08-04	Doug Davis	Verify all [refs]
wd-15	2006-08-04	Doug Davis	Change namespace to 2006/08
wd-15	2006-08-04	Doug Davis	Issue 148 applied
wd-15	2006-08-07	Doug Davis	Add some new glossary terms – per GilP
cd-04	2006-08-10	Gilbert Pilz	Formatting changes for better HTML rendering.

Rev	Date	By Whom	What
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.
wd-16	2006-10-25	Doug Davis	Accept all changes, update to wd16
wd-16	2006-10-26	Doug Davis	PR002 applied
wd-16	2006-10-26	Doug Davis	PR003 applied
wd-16	2006-10-26	Doug Davis	PR004 applied
wd-16	2006-10-27	Doug Davis	PR005 applied
wd-16	2006-10-27	Doug Davis	PR006 applied
wd-16	2006-10-27	Doug Davis	PR024 applied
wd-16	2006-11-13	Doug Davis	PR010 applied
wd-16	2006-11-13	Doug Davis	PR011 applied (technically as part of PR004)
wd-16	2006-11-13	Doug Davis	PR016 applied
wd-16	2006-11-13	Doug Davis	PR032 applied
wd-16	2006-11-20	Doug Davis	PR025 applied
wd-16	2006-11-20	Doug Davis	PR023 applied
wd-16	2006-12-03	Doug Davis	PR036 applied
wd-16	2006-12-03	Doug Davis	PR017 applied
wd-16	2006-12-11	Doug Davis	PR012 applied
wd-16	2006-12-14	Doug Davis	PR033 applied – changed a 'return' to 'generate' when talking about a fault
wd-16	2007-01-04	Doug Davis	PR018 applied
wd-16	2007-01-05	Doug Davis	Moved MakeConnection to new spec

2093 **Appendix G. Notices**

2094 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2095 might be claimed to pertain to the implementation or use of the technology described in this document or
2096 the extent to which any license under such rights might or might not be available; neither does it represent
2097 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2098 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2099 available for publication and any assurances of licenses to be made available, or the result of an attempt
2100 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2101 users of this specification, can be obtained from the OASIS Executive Director.

2102 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2103 other proprietary rights which may cover technology that may be required to implement this specification.
2104 Please address the information to the OASIS Executive Director.

2105 Copyright (C) OASIS Open (2006). All Rights Reserved.

2106 This document and translations of it may be copied and furnished to others, and derivative works that
2107 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2108 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2109 this paragraph are included on all such copies and derivative works. However, this document itself may
2110 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2111 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2112 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2113 into languages other than English.

2114 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2115 or assigns.

2116 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2117 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2118 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2119 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.