



1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

3 Working Draft 16, January 29, 2007

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

43	1 Introduction.....	4
44	1.1 Notational Conventions.....	4
45	1.2 Namespace.....	5
46	1.3 Conformance.....	5
47	2 Reliable Messaging Model.....	6
48	2.1 Glossary.....	6
49	2.2 Protocol Preconditions.....	7
50	2.3 Protocol Invariants.....	8
51	2.4 Example Message Exchange.....	8
52	3 RM Protocol Elements.....	10
53	3.1 Considerations on the Use of Extensibility Points.....	10
54	3.2 Considerations on the Use of "Piggy-Backing".....	10
55	3.3 Composition with WS-Addressing.....	10
56	3.4 Sequence Creation.....	10
57	3.5 Closing A Sequence.....	15
58	3.6 Sequence Termination.....	16
59	3.7 Sequences.....	18
60	3.8 Request Acknowledgement.....	19
61	3.9 Sequence Acknowledgement.....	20
62	4 Faults.....	23
63	4.1 SequenceFault Element.....	24
64	4.2 Sequence Terminated.....	25
65	4.3 Unknown Sequence.....	25
66	4.4 Invalid Acknowledgement.....	26
67	4.5 Message Number Rollover.....	26
68	4.6 Create Sequence Refused.....	27
69	4.7 Sequence Closed.....	27
70	4.8 WSRM Required.....	28
71	5 Security Threats and Countermeasures.....	29
72	5.1 Threats and Countermeasures.....	29
73	5.1.1 Integrity Threats.....	29
74	5.1.1.1 Countermeasures.....	29
75	5.1.2 Resource Consumption Threats.....	30
76	5.1.2.1 Countermeasures.....	30
77	5.1.3 Sequence Spoofing Threats.....	30
78	5.1.3.1 Sequence Hijacking.....	30
79	5.1.3.2 Countermeasures.....	30

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

105 **1 Introduction**

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

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

115 **1.1 Notational Conventions**

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

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

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

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

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

143 **1.2 Namespace**

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

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

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

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

150 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

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

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

154 **1.3 Conformance**

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

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

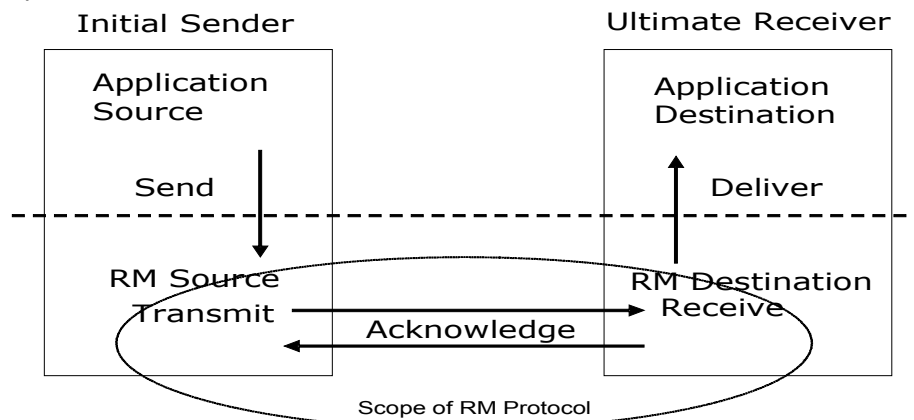
161 2 Reliable Messaging Model

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

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

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

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



187 Figure 1: Reliable Messaging Model

188 2.1 Glossary

189 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

218 2.2 Protocol Preconditions

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

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

228 2.3 Protocol Invariants

229 During the lifetime of a Sequence, the following invariants are REQUIRED for correctness:

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

242 2.4 Example Message Exchange

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

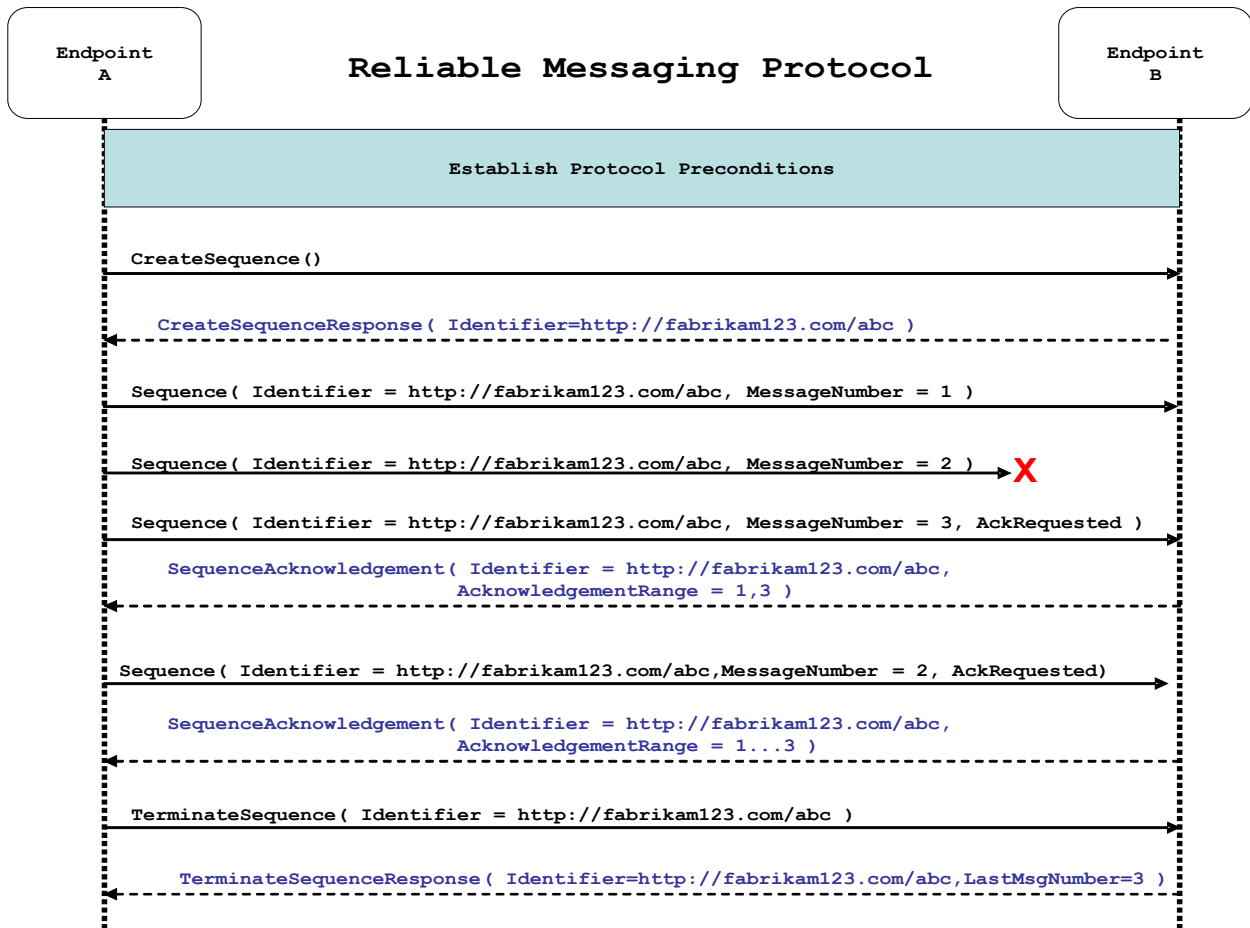


Figure 2: The WS-ReliableMessaging Protocol

- 244 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
245 and establishing trust.
- 246 2. The RM Source requests creation of a new Sequence.
- 247 3. The RM Destination creates a new Sequence and returns its unique identifier.
- 248 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1.
249 In the figure above, the RM Source sends 3 messages in the Sequence.
- 250 5. The 2nd message in the Sequence is lost in transit.
- 251 6. The 3rd message is the last in this Sequence and the RM Source includes an `AckRequested`
252 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 253 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the
254 RM Source's `AckRequested` header.
- 255 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new
256 message from the perspective of the underlying transport, but it has the same Sequence Identifier
257 and MessageNumber so the RM Destination can recognize it as a duplicate of the earlier message,
258 in case the original and retransmitted messages are both Received. The RM Source includes an
259 `AckRequested` header in the retransmitted message so the RM Destination will expedite an
260 acknowledgement.

261 9. The RM Destination Receives the second transmission of the message with MessageNumber 2
262 and acknowledges receipt of message numbers 1, 2, and 3.

263 10. The RM Source Receives this Acknowledgement and sends a TerminateSequence message to the
264 RM Destination indicating that the Sequence is completed. The TerminateSequence message
265 indicates that message number 3 was the last message in the Sequence. The RM Destination then
266 reclaims any resources associated with the Sequence.

267 11. The RM Destination Receives the TerminateSequence message indicating that the RM Source will
268 not be sending any more messages. The RM Destination sends a TerminateSequenceResponse
269 message to the RM Source and reclaims any resources associated with the Sequence.

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

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

284 **3 RM Protocol Elements**

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

287 **3.1 Considerations on the Use of Extensibility Points**

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

292 **3.2 Considerations on the Use of "Piggy-Backing"**

293 Some RM Protocol Header Blocks may be added to messages that are targeted to the same Endpoint to
294 which those headers are to be sent (a concept often referred to as "piggy-backing"), thus saving the
295 overhead of an additional message exchange. Reference parameters MUST be considered when
296 determining whether two EPRs are targeted to the same Endpoint. The determination of if and when a
297 Header Block will be piggy-backed onto another message is made by the entity (RM Source or RM
298 Destination) that is sending the header. In order to ensure optimal and successful processing of RM
299 Sequences, endpoints that receive RM-related messages SHOULD be prepared to process RM Protocol
300 Header Blocks that are included in any message it receives. See the sections that define each RM
301 Protocol Header Block to know which ones may be considered for piggy-backing.

302 **3.3 Composition with WS-Addressing**

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

- 305 1. When an Endpoint generates a message that carries an RM protocol element, that is defined in
306 the following sections, in the body of a SOAP envelope that Endpoint MUST include in that
307 envelope a `wsa:Action` SOAP header block whose value is an IRI that is a concatenation of the
308 WS-RM namespace URI, followed by a "/", followed by the value of the local name of the child
309 element of the SOAP body. For example, for a Sequence creation request message as described
310 in section 3.4 below, the value of the `wsa:Action` IRI would be:

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

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

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

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

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

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

320 **3.4 Sequence Creation**

321 The RM Source MUST request creation of an outbound Sequence by sending a `CreateSequence`
322 element in the body of a message to the RM Destination which in turn responds either with a message

323 containing `CreateSequenceResponse` or a `CreateSequenceRefused` fault. The RM Source MAY
324 include an offer to create an inbound Sequence within the `CreateSequence` message. This offer is
325 either accepted or rejected by the RM Destination in the `CreateSequenceResponse` message.

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

328 The following exemplar defines the `CreateSequence` syntax:

```
329 <wsrm:CreateSequence ...>  
330   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
331   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
332   <wsrm:Offer ...>  
333     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
334     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
335     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
336     <wsrm:IncompleteSequenceBehavior>  
337       wsrml:IncompleteSequenceBehaviorType  
338     </wsrm:IncompleteSequenceBehavior> ?  
339     ...  
340   </wsrm:Offer> ?  
341   ...  
342 </wsrm:CreateSequence>
```

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

344 `/wsrm:CreateSequence`

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

349 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

359 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

367 `/wsrm:CreateSequence/wsrm:Offer`

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

370 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier

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

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

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

376 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint

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

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

386 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires

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

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

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

393 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior

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

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

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

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

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

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

410 /wsmr:CreateSequence/{any}

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

413 /wsmr:CreateSequence/@{any}

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

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

420 The following exemplar defines the `CreateSequenceResponse` syntax:

```
421 <wsmr:CreateSequenceResponse ...>  
422   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
423   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
424   <wsmr:IncompleteSequenceBehavior>  
425     wsmr:IncompleteSequenceBehaviorType  
426   </wsmr:IncompleteSequenceBehavior> ?  
427   <wsmr:Accept ...>  
428     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
429     ...  
430   </wsmr:Accept> ?  
431   ...  
432 </wsmr:CreateSequenceResponse>
```

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

434 /wsmr:CreateSequenceResponse

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

438 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

445 /wsmr:CreateSequenceResponse/wsmr:Expires

446 This element, if present, of type `xs:duration` accepts or refines the RM Source's requested duration for
447 the Sequence. It specifies the amount of time after which any resources associated with the Sequence
448 SHOULD be reclaimed thus causing the Sequence to be silently terminated. At the RM Destination this
449 duration is measured from a point proximate to Sequence creation and at the RM Source this duration is
450 measured from a point approximate to the successful processing of the `CreateSequenceResponse`. A
451 value of "PT0S" indicates that the Sequence will never expire. Absence of the element indicates an

452 implied value of "PT0S". The RM Destination MUST set the value of this element to be equal to or less
453 than the value requested by the RM Source in the corresponding `CreateSequence` message.

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

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

457 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

468 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

474 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

489 `/wsrm:CreateSequenceResponse/{any}`

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

492 /wsrm:CreateSequenceResponse/@{any}

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

495 **3.5 Closing A Sequence**

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

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

510 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM
511 Source SHOULD include the `LastMsgNumber` element in any `CloseSequence` messages it sends. The
512 RM Destination can use this information, for example, to implement the behavior indicated by /
513 `wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the
514 `LastMsgNumber` element MUST be the same in all the `CloseSequence` messages for the closing
515 Sequence.

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

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

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

529 The following exemplar defines the `CloseSequence` syntax:

```
530 <wsrm:CloseSequence ...>  
531   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
532   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
533   ...  
534 </wsrm:CloseSequence>
```

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

536 /wsmr:CloseSequence

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

540 /wsmr:CloseSequence/wsmr:Identifier

541 The RM Source or RM Destination MUST include this element in any CloseSequence messages it sends.
542 The RM Source or RM Destination MUST set the value of this element to the absolute URI (conformant
543 with RFC3986) of the closing Sequence.

544 /wsmr:CloseSequence/wsmr:LastMessageNumber

545 The RM Source SHOULD include this element in any CloseSequence message it sends. The
546 LastMsgNumber element specifies the highest assigned message number of all the Sequence Traffic
547 Messages for the closing Sequence.

548 /wsmr:CloseSequence/wsmr:Identifier/@{any}

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

551 /wsmr:CloseSequence/{any}

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

554 /wsmr:CloseSequence@{any}

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

557 A CloseSequenceResponse is sent in the body of a message in response to receipt of a
558 CloseSequence request message. It indicates that the responder has closed the Sequence.

559 The following exemplar defines the CloseSequenceResponse syntax:

```
560 <wsmr:CloseSequenceResponse ...>  
561   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
562   ...  
563 </wsmr:CloseSequenceResponse>
```

564 The following describes the content model of the CloseSequenceResponse element.

565 /wsmr:CloseSequenceResponse

566 This element is sent in the body of a message in response to receipt of a CloseSequence request
567 message. It indicates that the responder has closed the Sequence.

568 /wsmr:CloseSequenceResponse/wsmr:Identifier

569 The responder (RM Source or RM Destination) MUST include this element in any
570 CloseSequenceResponse message it sends. The responder MUST set the value of this element to the
571 absolute URI (conformant with RFC3986) of the closing Sequence.

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

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

575 /wsmr:CloseSequenceResponse/{any}

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

578 /wsrm:CloseSequenceResponse@{any}

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

581 3.6 Sequence Termination

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

589 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM
590 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.
591 The RM Destination can use this information, for example, to implement the behavior indicated by /
592 `wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the
593 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the
594 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same
595 Sequence.

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

600 The following exemplar defines the `TerminateSequence` syntax:

```
601 <wsrm:TerminateSequence ...>  
602   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
603   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
604   ...  
605 </wsrm:TerminateSequence>
```

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

607 /wsrm:TerminateSequence

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

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

618 /wsrm:TerminateSequence/wsrm:Identifier

619 The RM Source or RM Destination MUST include this element in any TerminateSequence message it
620 sends. The RM Source or RM Destination MUST set the value of this element to the absolute URI
621 (conformant with RFC3986) of the terminating Sequence.

622 /wsm:TerminateSequence/wsm:LastMsgNumber

623 The RM Source SHOULD include this element in any TerminateSequence message it sends. The
624 LastMsgNumber element specifies the highest assigned message number of all the Sequence Traffic
625 Messages for the closing Sequence.

626 /wsm:TerminateSequence/wsm:Identifier/{any}

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

629 /wsm:TerminateSequence/{any}

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

632 /wsm:TerminateSequence/{any}

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

635 A TerminateSequenceResponse is sent in the body of a message in response to receipt of a
636 TerminateSequence request message. It indicates that responder has terminated the Sequence.

637 The following exemplar defines the TerminateSequenceResponse syntax:

```
638 <wsm:TerminateSequenceResponse ...>  
639   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
640   ...  
641 </wsm:TerminateSequenceResponse>
```

642 The following describes the content model of the TerminateSequence element.

643 /wsm:TerminateSequenceResponse

644 This element is sent in the body of a message in response to receipt of a TerminateSequence request
645 message. It indicates that the responder has terminated the Sequence. The responder MUST NOT send
646 this element as a header block.

647 /wsm:TerminateSequenceResponse/wsm:Identifier

648 The responder (RM Source or RM Destination) MUST include this element in any
649 TerminateSequenceResponse message it sends. The responder MUST set the value of this element
650 to the absolute URI (conformant with RFC3986) of the terminating Sequence.

651 /wsm:TerminateSequenceResponse/wsm:Identifier/{any}

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

654 /wsm:TerminateSequenceResponse/{any}

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

657 /wsm:TerminateSequenceResponse/{any}

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

660 On receipt of a `TerminateSequence` message the receiver (RM Source or RM Destination) MUST
661 respond with a corresponding `TerminateSequenceResponse` message or generate a fault
662 `UnknownSequenceFault` if the Sequence is not known.

663 3.7 Sequences

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

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

671 A following exemplar defines its syntax:

```
672 <wsrm:Sequence ...>  
673   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
674   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
675   ...  
676 </wsrm:Sequence>
```

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

678 `/wsrm:Sequence`

679 This protocol element associates the message in which it is contained with a previously established RM
680 Sequence. It contains the Sequence's unique identifier and the containing message's ordinal position
681 within that Sequence. The RM Destination MUST understand the `Sequence` header block. The RM
682 Source MUST assign a `mustUnderstand` attribute with a value 1/true (from the namespace
683 corresponding to the version of SOAP to which the `Sequence` SOAP header block is bound) to the
684 `Sequence` header block element.

685 `/wsrm:Sequence/wsrm:Identifier`

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

689 `/wsrm:Sequence/wsrm:Identifier/@{any}`

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

692 `/wsrm:Sequence/wsrm:MessageNumber`

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

697 `/wsrm:Sequence/{any}`

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

700 `/wsrm:Sequence/@{any}`

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

703 The following example illustrates a Sequence header block.

```
704 <wsrm:Sequence>  
705   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
706   <wsrm:MessageNumber>10</wsrm:MessageNumber>  
707 </wsrm:Sequence>
```

708 3.8 Request Acknowledgement

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

711 The RM Source MAY request an Acknowledgement Message from the RM Destination at any time by
712 independently transmitting an `AckRequested` header block (i.e. as a header of a SOAP envelope with an
713 empty body). Alternatively the RM Source MAY include an `AckRequested` header block in any message
714 targeted to the RM Destination. The RM Destination SHOULD process `AckRequested` header blocks
715 that are included in any message it receives. If a non-mustUnderstand fault occurs when processing an
716 `AckRequested` header block that was piggy-backed, a fault MUST be generated, but the processing of
717 the original message MUST NOT be affected.

718 An RM Destination that Receives a message that contains an `AckRequested` header block MUST send
719 a message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference
720 (see Section 3.4) for a known Sequence or else generate an `UnknownSequence` fault. It is
721 RECOMMENDED that the RM Destination return a `AcknowledgementRange` or `None` element instead
722 of a `Nack` element (see Section 3.9).

723 The following exemplar defines its syntax:

```
724 <wsrm:AckRequested ...>  
725   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
726   ...  
727 </wsrm:AckRequested>
```

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

729 `/wsrm:AckRequested`

730 This element requests an Acknowledgement for the identified Sequence.

731 `/wsrm:AckRequested/wsrm:Identifier`

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

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

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

738 `/wsrm:AckRequested/{any}`

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

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

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

744 **3.9 Sequence Acknowledgement**

745 The RM Destination informs the RM Source of successful message receipt using a
746 `SequenceAcknowledgement` header block. Acknowledgements can be explicitly requested using the
747 `AckRequested` directive (see Section 3.8).

748 The RM Destination MAY Transmit the `SequenceAcknowledgement` header block independently (i.e.
749 As a header of a SOAP envelope with an empty body). Alternatively, an RM Destination MAY include a
750 `SequenceAcknowledgement` header block on any SOAP envelope targeted to the endpoint referenced
751 by the `AcksTo` EPR. The RM Source SHOULD process `SequenceAcknowledgement` header blocks
752 that are included in any message it receives. If a non-mustUnderstand fault occurs when processing a
753 `SequenceAcknowledgement` header that was piggy-backed, a fault MUST be generated, but the
754 processing of the original message MUST NOT be affected.

755 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
756 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
757 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
758 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
759 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
760 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
761 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`
762 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination
763 SHOULD respond on the protocol binding-specific back-channel provided by the Received message
764 containing the `AckRequested` header block.

765 The following exemplar defines its syntax:

```
766 <wsrm:SequenceAcknowledgement ...>  
767   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
768   [ [ [ <wsrm:AcknowledgementRange ...  
769     Upper="wsrm:MessageNumberType"  
770     Lower="wsrm:MessageNumberType"/> +  
771     | <wsrm:None/> ]  
772     <wsrm:Final/> ? ]  
773     | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]  
774  
775     ...  
776 </wsrm:SequenceAcknowledgement>
```

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

778 `/wsrm:SequenceAcknowledgement`

779 This element contains the Sequence Acknowledgement information.

780 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

786 `/wsrm:SequenceAcknowledgement/wsrm:Identifier/@{any}`

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

789 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

795 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper`

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

798 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower`

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

801 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}`

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

804 `/wsrm:SequenceAcknowledgement/wsrm:None`

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

809 `/wsrm:SequenceAcknowledgement/wsrm:Final`

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

816 `/wsrm:SequenceAcknowledgement/wsrm:Nack`

817 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If
818 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing
819 the `MessageNumber` of an unreceived message in a Sequence. The RM Destination MUST NOT include
820 a `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of
821 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the
822 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`
823 containing a `Nack` for a message that it has previously acknowledged within a
824 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing
825 a `Nack` for a message that has previously been acknowledged within a `AcknowledgementRange`.

826 `/wsrm:SequenceAcknowledgement/{any}`

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

829 /wsrm:SequenceAcknowledgement/@{any}

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

832 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
834 <wsrm:SequenceAcknowledgement>  
835   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
836   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
837 </wsrm:SequenceAcknowledgement>
```

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

```
840 <wsrm:SequenceAcknowledgement>  
841   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
842   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
843   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
844   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
845 </wsrm:SequenceAcknowledgement>
```

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

```
847 <wsrm:SequenceAcknowledgement>  
848   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
849   <wsrm:Nack>3</wsrm:Nack>  
850 </wsrm:SequenceAcknowledgement>
```


851 4 Faults

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

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

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

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

865 The definitions of faults use the following properties:

866 [Code] The fault code.

867 [Subcode] The fault subcode.

868 [Reason] The English language reason element.

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

872 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
873 "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

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

```
875 <S:Envelope>
876   <S:Header>
877     <wsa:Action>
878       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault
879     </wsa:Action>
880     <!-- Headers elided for brevity. -->
881   </S:Header>
882   <S:Body>
883     <S:Fault>
884       <S:Code>
885         <S:Value> [Code] </S:Value>
886         <S:Subcode>
887           <S:Value> [Subcode] </S:Value>
888         </S:Subcode>
889       </S:Code>
890       <S:Reason>
891         <S:Text xml:lang="en"> [Reason] </S:Text>
892       </S:Reason>
893       <S:Detail>
894         [Detail]
```

```
895     ...
896     </S:Detail>
897 </S:Fault>
898 </S:Body>
899 </S:Envelope>
```

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

```
902 <S11:Envelope>
903   <S11:Header>
904     <wsrm:SequenceFault>
905       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
906       <wsrm:Detail> [Detail] </wsrm:Detail>
907       ...
908     </wsrm:SequenceFault>
909     <!-- Headers elided for brevity. -->
910   </S11:Header>
911   <S11:Body>
912     <S11:Fault>
913       <faultcode> [Code] </faultcode>
914       <faultstring> [Reason] </faultstring>
915     </S11:Fault>
916   </S11:Body>
917 </S11:Envelope>
```

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

```
920 <S11:Envelope>
921   <S11:Body>
922     <S11:Fault>
923       <faultcode> [Subcode] </faultcode>
924       <faultstring> [Reason] </faultstring>
925     </S11:Fault>
926   </S11:Body>
927 </S11:Envelope>
```

928 4.1 SequenceFault Element

929 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during
930 the reliable messaging specific processing of a message belonging to a Sequence. WS-
931 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP
932 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in
933 conjunction with the SOAP 1.2 binding.

934 The following exemplar defines its syntax:

```
935 <wsrm:SequenceFault ...>
936   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
937   <wsrm:Detail> ... </wsrm:Detail> ?
938   ...
939 </wsrm:SequenceFault>
```

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

941 /wsrm:SequenceFault

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

943 /wsrm:SequenceFault/wsrm:FaultCode

944 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a
945 qualified name from the set of fault [Subcodes] defined below.

946 `/wsrm:SequenceFault/wsrm:Detail`

947 This element, if present, carries application specific error information related to the fault being described.

948 `/wsrm:SequenceFault/wsrm:Detail/{any}`

949 The application specific error information related to the fault being described.

950 `/wsrm:SequenceFault/wsrm:Detail/@{any}`

951 The application specific error information related to the fault being described.

952 `/wsrm:SequenceFault/{any}`

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

955 `/wsrm:SequenceFault/@{any}`

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

958 4.2 Sequence Terminated

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

961 Properties:

962 [Code] Sender or Receiver

963 [Subcode] `wsrn:SequenceTerminated`

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

965 [Detail]

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

967 4.3 Unknown Sequence

968 Properties:

969 [Code] Sender

970 [Subcode] `wsrn:UnknownSequence`

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

972 [Detail]

973 `<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.

974 4.4 Invalid Acknowledgement

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

977 [Code] Sender

978 [Subcode] `wsrm:InvalidAcknowledgement`

979 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

980 [Detail]

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

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

982 4.5 Message Number Rollover

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

984 Properties:

985 [Code] Sender

986 [Subcode] `wsrm:MessageNumberRollover`

987 [Reason] The maximum value for `wsrm:MessageNumber` has been exceeded.

988 [Detail]

989 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`
 990 `<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.

991 **4.6 Create Sequence Refused**

992 Properties:

993 [Code] Sender or Receiver

994 [Subcode] wsrm:CreateSequenceRefused

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

996 [Detail]

997 `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.

998 **4.7 Sequence Closed**

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

1000 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
 1001 is closed.

1002 Properties:

1003 [Code] Sender

1004 [Subcode] wsrm:SequenceClosed

1005 [Reason] The Sequence is closed and can not accept new messages.

1006 [Detail]

1007 `<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.

1008 **4.8 WSRM Required**

1009 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1010 message that did not use this protocol.

1011 Properties:

1012 [Code] Sender

1013 [Subcode] wsrn:WSRMRequired

1014 [Reason] The RM Destination requires the use of WSRM.

1015 [Detail]

1016 *xs:any*

1017 **5 Security Threats and Countermeasures**

1018 This specification considers two sets of security requirements, those of the applications that use the WS-
1019 RM protocol and those of the protocol itself.

1020 This specification makes no assumptions about the security requirements of the applications that use WS-
1021 RM. However, once those requirements have been satisfied within a given operational context, the
1022 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1023 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1024 There are many other security concerns that one may need to consider when implementing or using this
1025 protocol. The material below should not be considered as a "check list". Implementers and users of this
1026 protocol are urged to perform a security analysis to determine their particular threat profile and the
1027 appropriate responses to those threats.

1028 Implementers are also advised that there is a core tension between security and reliable messaging that
1029 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1030 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1031 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1032 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1033 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1034 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1035 avoid and prevent this condition.

1036 **5.1 Threats and Countermeasures**

1037 The primary security requirement of this protocol is to protect the specified semantics and protocol
1038 invariants against various threats. The following sections describe several threats to the integrity and
1039 operation of this protocol and provide some general outlines of countermeasures to those threats.
1040 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1041 to all operational contexts.

1042 **5.1.1 Integrity Threats**

1043 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1044 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1045 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1046 to its intended message represents a threat to the WS-RM protocol.

1047 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM
1048 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1049 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be
1050 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1051 **5.1.1.1 Countermeasures**

1052 Integrity threats are generally countered via the use of digital signatures some level of the communication
1053 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1054 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers
1055 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which
1056 they occur, implementations MUST allow for signatures that cover only these headers.

1057 **5.1.2 Resource Consumption Threats**

1058 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1059 implement that RM Destination. These resources can include network connections, database tables,
1060 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1061 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1062 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1063 message Delivery and use this Sequence to send a stream of very large messages to that service,
1064 making sure to omit message number "1" from that stream.

1065 **5.1.2.1 Countermeasures**

1066 There are a number of countermeasures against the described resource consumption threats. The
1067 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1068 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1069 some cases, allows the identity of any attackers to be determined.

1070 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and
1071 authenticate the RM Source that issued the `CreateSequence` message.

1072 **5.1.3 Sequence Spoofing Threats**

1073 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1074 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1075 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1076 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the
1077 current `MessageNumber` for their target Sequence.

1078 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP
1079 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier
1080 to attack the Sequence. These attacks are "two-way" in that an attacker may choose to target the RM
1081 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends
1082 to the `AcksTo` EPR of an RM Source.

1083 **5.1.3.1 Sequence Hijacking**

1084 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject
1085 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1086 messages.

1087 Note that "sequence hijacking" should not be equated with "security session hijacking". Although a
1088 Sequence may be bound to some form of a security session in order to counter the threats described in
1089 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1090 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1091 established by the security infrastructure to make such determinations. Failure to observe this rule
1092 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1093 the ability to authenticate its peers even though the necessary security processing has taken place.

1094 **5.1.3.2 Countermeasures**

1095 There are a number of countermeasures against sequence spoofing threats. The technique advocated by
1096 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1097 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1098 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1099 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1100 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1101 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1102 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1103 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1104 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1105 sequence peer it MUST be able to identify and authenticate the entity that sent the
1106 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1107 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1108 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1109 creation time.

1110 **5.2 Security Solutions and Technologies**

1111 The security threats described in the previous sections are neither new nor unique. The solutions that
1112 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1113 section maps the facilities provided by common web services security solutions against countermeasures
1114 described in the previous sections.

1115 Before continuing this discussion, however, some examination of the underlying requirements of the
1116 previously described countermeasures is necessary. Specifically it should be noted that the technique
1117 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1118 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check
1119 against this authenticated identity and determines if the RM Source is permitted to create Sequences with
1120 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,
1121 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of
1122 such facilities is considered to be beyond the scope of this specification.

1123 **5.2.1 Transport Layer Security**

1124 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1125 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1126 defined in Section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1127 The description provided here is general in nature and is not intended to serve as a complete definition on
1128 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1129 choice of features as well as the manner in which they will be used. The mechanisms described in the
1130 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1131 requirements and constraints of the use of SSL/TLS.

1132 **5.2.1.1 Model**

1133 The basic model for using SSL/TLS is as follows:

- 1134 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1135 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1136 Destination.

- 1137 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1138 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1139 synchronous `CreateSequenceResponse` using the session established in (1).
- 1140 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1141 any and all messages or faults that refer to that Sequence.
- 1142 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1143 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1144 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1145 5.2.1.2 Countermeasure Implementation

1146 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1147 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1148 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1149 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1150 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1151 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1152 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1153 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1154 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1155 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1156 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1157 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1158 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1159 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1160 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1161 Acknowledgement) using BasicAuth.
- 1162 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1163 connection authenticates itself to the party accepting the connection using an X.509 certificate
1164 that is exchanged during the SSL/TLS handshake.

1165 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1166 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1167 Source is authorized to create a Sequence with the RM Destination.

1168 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1169 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1170 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1171 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1172 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1173 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1174 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1175 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1176 to protect that Sequence.

1177 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1178 countermeasures (such as associating specific authentication information with a Sequence) although such
1179 methods are not covered by this document.

1180 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1181 session) are outside the scope of this specification.

1182 **5.2.2 SOAP Message Security**

1183 The mechanisms described in WS-Security may be used in various ways to implement the
1184 countermeasures described in the previous sections. This specification advocates using the protocol
1185 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust
1186 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1187 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1188 The description provided here is general in nature and is not intended to serve as a complete definition on
1189 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1190 need to agree on the choice of features as well as the manner in which they will be used. The
1191 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1192 describe the requirements and constraints of the use of WS-SecureConversation.

1193 **5.2.2.1 Model**

1194 The basic model for using WS-SecureConversation is as follows:

- 1195 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This
1196 may involve the participation of third parties such as a security token service. The tokens
1197 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1198 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1199 context that will be used to protect the Sequence. This is done so that, in cases where the
1200 `CreateSequence` message is signed by more than one security context, the RM Source can
1201 indicate which security context should be used to protect the newly created Sequence.
- 1202 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1203 associated with the security context to sign (as defined by WS-Security) at least the body and any
1204 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

1205 **5.2.2.2 Countermeasure Implementation**

1206 Without relying upon any authentication information, the per-message signatures provide the necessary
1207 integrity qualities to counter the threats described in Section 5.1.1.

1208 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1209 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1210 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1211 create a Sequence with the RM Destination.

1212 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1213 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1214 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1215 context rather than on any authentication claims that may have been established during security context
1216 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1217 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1218 document.

1219 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer
1220 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1221 the association between a Sequence and its protecting security context cannot always be established
1222 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1223 `CreateSequenceResponse` messages may be signed by more than one security context.

1224 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1225 amending or renewing contexts) are outside the scope of this specification.

1226 6 Securing Sequences

1227 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared
1228 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1229 achieving this objective depending upon the underlying security infrastructure.

1230 6.1 Securing Sequences Using WS-Security

1231 One mechanism for protecting a Sequence is to include a security token using a
1232 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1233 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1234 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1235 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1236 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1237 there may be more than one token on a `CreateSequence` message or inferred from the communication
1238 context (e.g. transport protection).

1239 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1240 if the token being referenced supports such mechanism.

1241 The following exemplar defines the `CreateSequence` syntax when extended to include a
1242 `wsse:SecurityTokenReference`:

```
1243 <wsrm:CreateSequence ...>  
1244   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1245   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1246   <wsrm:Offer ...>  
1247     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1248     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1249     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1250     <wsrm:IncompleteSequenceBehavior>  
1251       wsrml:IncompleteSequenceBehaviorType  
1252     </wsrm:IncompleteSequenceBehavior> ?  
1253     ...  
1254   </wsrm:Offer> ?  
1255   ...  
1256   <wsse:SecurityTokenReference>  
1257     ...  
1258   </wsse:SecurityTokenReference> ?  
1259   ...  
1260 </wsrm:CreateSequence>
```

1261 The following describes the content model of the additional `CreateSequence` elements.

1262 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1263 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in
1264 section 3.4) to communicate an explicit reference to the security token, using a
1265 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and Destination
1266 MUST use to authorize messages for the created (and, if present, the offered) Sequence(s). All
1267 subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST
1268 demonstrate proof-of-possession of the secret associated with the token (e.g., by using or deriving from a
1269 private or secret key).

1270 When a RM Source transmits a `CreateSequence` that has been extended to include a
1271 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and
1272 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1273 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This

1274 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1275 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1276 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1277 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1278 in WS-Security still applies.

1279 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1280 <wsrm:UsesSequenceSTR ... />
```

1281 The following describes the content model of the `UsesSequenceSTR` header block.

1282 `/wsrm:UsesSequenceSTR`

1283 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1284 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1285 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1286 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1287 Sequence creation.

1288 The following is an example of a `CreateSequence` message using the

1289 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1290 <soap:Envelope ...>  
1291   <soap:Header>  
1292     ...  
1293     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1294     ...  
1295   </soap:Header>  
1296   <soap:Body>  
1297     <wsrm:CreateSequence>  
1298       <wsrm:AcksTo>  
1299         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1300       </wsrm:AcksTo>  
1301       <wsse:SecurityTokenReference>  
1302         ...  
1303       </wsse:SecurityTokenReference>  
1304     </wsrm:CreateSequence>  
1305   </soap:Body>  
1306 </soap:Envelope>
```

1307 6.2 Securing Sequences Using SSL/TLS

1308 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1309 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1310 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1311 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1312 SOAP header block within the `CreateSequence` message.

1313 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1314 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1315 The following describes the content model of the `UsesSequenceSSL` header block.

1316 `/wsrm:UsesSequenceSSL`

1317 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1318 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was
1319 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a
1320 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand

1321 and correctly implement the functionality described in Section 5.2.1 or else generate a
1322 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1323 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1324 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1325 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1326 `CreateSequenceResponse` message.

1327 **7 References**

1328 **7.1 Normative**

1329 **[KEYWORDS]**

1330 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,
1331 March 1997

1332 <http://www.ietf.org/rfc/rfc2119.txt>

1333 **[WS-RM Policy]**

1334 OASIS WS-RX Technical Committee Draft, "[Web Services ReliableMessaging Policy Assertion\(WS-RM
1335 Policy\)](#)" October 2006

1336 <http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-wd-11.pdf>

1337 **[SOAP 1.1]**

1338 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

1339 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1340 **[SOAP 1.2]**

1341 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

1342 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

1343 **[URI]**

1344 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,
1345 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1346 <http://ietf.org/rfc/rfc3986>

1347 **[UUID]**

1348 P. Leach, M. Mealling, R. Salz, "[A Universally Unique IDentifier \(UUID\) URN Namespace](#)," RFC 4122,
1349 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1350 <http://www.ietf.org/rfc/rfc4122.txt>

1351 **[XML]**

1352 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#)", September 2006.

1353 <http://www.w3.org/TR/REC-xml/>

1354 **[XML-ns]**

1355 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

1356 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

1357 **[XML-Schema Part1]**

1358 W3C Recommendation, "[XML Schema Part 1: Structures](#)," October 2004.

1359 <http://www.w3.org/TR/xmlschema-1/>

1360 **[XML-Schema Part2]**

1361 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

1362 <http://www.w3.org/TR/xmlschema-2/>

1363 **[XPath 1.0]**

1364 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

1365 <http://www.w3.org/TR/xpath>

1366 **[WSDL 1.1]**

1367 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

1368 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1369 **[WS-Addressing]**

1370 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

1371 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

1372 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

1373 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1374 **7.2 Non-Normative**

1375 **[BSP 1.0]**

1376 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

1377 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

1378 **[RDDL 2.0]**

1379 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

1380 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

1381 **[RFC 2617]**

1382 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
1383 Authentication: Basic and Digest Access Authentication," June 1999.

1384 <http://www.ietf.org/rfc/rfc2617.txt>

1385 **[RFC 4346]**

1386 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

1387 <http://www.ietf.org/rfc/rfc4346.txt>

1388 **[WS-Policy]**

1389 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

1390 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

1391 **[WS-PolicyAttachment]**

1392 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

1393 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-
1394 20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1395 **[WS-Security]**

1396 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1397 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1398 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

1399 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1400 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

1401 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

1402 **[RTTM]**

1403 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May
1404 1992.

1405 <http://www.rfc-editor.org/rfc/rfc1323.txt>

1406 **[SecurityPolicy]**

1407 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

1408 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

1409 **[SecureConversation]**

1410 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February
1411 2005.

1412 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

1413 **[Trust]**

1414 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

1415 <http://schemas.xmlsoap.org/ws/2005/02/trust>

1416 Appendix A. Schema

1417 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1418 Schema Part2] is located at:

1419 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

1420 The following copy is provided for reference.

```
1421 <?xml version="1.0" encoding="UTF-8"?>
1422 <!--
1423 OASIS takes no position regarding the validity or scope of any intellectual
1424 property or other rights that might be claimed to pertain to the
1425 implementation or use of the technology described in this document or the
1426 extent to which any license under such rights might or might not be available;
1427 neither does it represent that it has made any effort to identify any such
1428 rights. Information on OASIS's procedures with respect to rights in OASIS
1429 specifications can be found at the OASIS website. Copies of claims of rights
1430 made available for publication and any assurances of licenses to be made
1431 available, or the result of an attempt made to obtain a general license or
1432 permission for the use of such proprietary rights by implementors or users of
1433 this specification, can be obtained from the OASIS Executive Director.
1434 OASIS invites any interested party to bring to its attention any copyrights,
1435 patents or patent applications, or other proprietary rights which may cover
1436 technology that may be required to implement this specification. Please
1437 address the information to the OASIS Executive Director.
1438 Copyright &copy; OASIS Open 2002-2006. All Rights Reserved.
1439 This document and translations of it may be copied and furnished to others,
1440 and derivative works that comment on or otherwise explain it or assist in its
1441 implementation may be prepared, copied, published and distributed, in whole or
1442 in part, without restriction of any kind, provided that the above copyright
1443 notice and this paragraph are included on all such copies and derivative
1444 works. However, this document itself does not be modified in any way, such as
1445 by removing the copyright notice or references to OASIS, except as needed for
1446 the purpose of developing OASIS specifications, in which case the procedures
1447 for copyrights defined in the OASIS Intellectual Property Rights document must
1448 be followed, or as required to translate it into languages other than English.
1449 The limited permissions granted above are perpetual and will not be revoked by
1450 OASIS or its successors or assigns.
1451 This document and the information contained herein is provided on an "AS IS"
1452 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1453 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1454 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1455 FOR A PARTICULAR PURPOSE.
1456 -->
1457 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1458 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1459 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1460 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1461 elementFormDefault="qualified" attributeFormDefault="unqualified">
1462 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1463 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1464 <!-- Protocol Elements -->
1465 <xs:complexType name="SequenceType">
1466 <xs:sequence>
1467 <xs:element ref="wsrm:Identifier"/>
1468 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1469 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1470 maxOccurs="unbounded"/>
1471 </xs:sequence>
1472 <xs:anyAttribute namespace="##other" processContents="lax"/>
```

```

1473 </xs:complexType>
1474 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1475 <xs:element name="SequenceAcknowledgement">
1476   <xs:complexType>
1477     <xs:sequence>
1478       <xs:element ref="wsrm:Identifier"/>
1479       <xs:choice>
1480         <xs:sequence>
1481           <xs:choice>
1482             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1483               <xs:complexType>
1484                 <xs:sequence/>
1485                 <xs:attribute name="Upper" type="xs:unsignedLong"
1486 use="required"/>
1487                 <xs:attribute name="Lower" type="xs:unsignedLong"
1488 use="required"/>
1489                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1490               </xs:complexType>
1491             </xs:element>
1492             <xs:element name="None">
1493               <xs:complexType>
1494                 <xs:sequence/>
1495               </xs:complexType>
1496             </xs:element>
1497           </xs:choice>
1498           <xs:element name="Final" minOccurs="0">
1499             <xs:complexType>
1500               <xs:sequence/>
1501             </xs:complexType>
1502           </xs:element>
1503         </xs:sequence>
1504         <xs:element name="Nack" type="xs:unsignedLong"
1505 maxOccurs="unbounded"/>
1506       </xs:choice>
1507       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1508 maxOccurs="unbounded"/>
1509     </xs:sequence>
1510     <xs:anyAttribute namespace="##other" processContents="lax"/>
1511   </xs:complexType>
1512 </xs:element>
1513 <xs:complexType name="AckRequestedType">
1514   <xs:sequence>
1515     <xs:element ref="wsrm:Identifier"/>
1516     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1517 maxOccurs="unbounded"/>
1518   </xs:sequence>
1519   <xs:anyAttribute namespace="##other" processContents="lax"/>
1520 </xs:complexType>
1521 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1522 <xs:element name="Identifier">
1523   <xs:complexType>
1524     <xs:annotation>
1525       <xs:documentation>
1526         This type is for elements whose [children] is an anyURI and can have
1527 arbitrary attributes.
1528       </xs:documentation>
1529     </xs:annotation>
1530     <xs:simpleContent>
1531       <xs:extension base="xs:anyURI">
1532         <xs:anyAttribute namespace="##other" processContents="lax"/>
1533       </xs:extension>
1534     </xs:simpleContent>
1535   </xs:complexType>

```

```

1536 </xs:element>
1537 <xs:element name="Address">
1538   <xs:complexType>
1539     <xs:simpleContent>
1540       <xs:extension base="xs:anyURI">
1541         <xs:anyAttribute namespace="##other" processContents="lax"/>
1542       </xs:extension>
1543     </xs:simpleContent>
1544   </xs:complexType>
1545 </xs:element>
1546 <xs:simpleType name="MessageNumberType">
1547   <xs:restriction base="xs:unsignedLong">
1548     <xs:minInclusive value="1"/>
1549     <xs:maxInclusive value="9223372036854775807"/>
1550   </xs:restriction>
1551 </xs:simpleType>
1552 <!-- Fault Container and Codes -->
1553 <xs:simpleType name="FaultCodes">
1554   <xs:restriction base="xs:QName">
1555     <xs:enumeration value="wsrm:SequenceTerminated"/>
1556     <xs:enumeration value="wsrm:UnknownSequence"/>
1557     <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1558     <xs:enumeration value="wsrm:MessageNumberRollover"/>
1559     <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1560     <xs:enumeration value="wsrm:SequenceClosed"/>
1561     <xs:enumeration value="wsrm:WSRMRequired"/>
1562     <xs:enumeration value="wsrm:UnsupportedSelection"/>
1563   </xs:restriction>
1564 </xs:simpleType>
1565 <xs:complexType name="SequenceFaultType">
1566   <xs:sequence>
1567     <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1568     <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1569     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1570 maxOccurs="unbounded"/>
1571   </xs:sequence>
1572   <xs:anyAttribute namespace="##other" processContents="lax"/>
1573 </xs:complexType>
1574 <xs:complexType name="DetailType">
1575   <xs:sequence>
1576     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1577 maxOccurs="unbounded"/>
1578   </xs:sequence>
1579   <xs:anyAttribute namespace="##other" processContents="lax"/>
1580 </xs:complexType>
1581 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1582 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1583 <xs:element name="CreateSequenceResponse"
1584 type="wsrm:CreateSequenceResponseType"/>
1585 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1586 <xs:element name="CloseSequenceResponse"
1587 type="wsrm:CloseSequenceResponseType"/>
1588 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1589 <xs:element name="TerminateSequenceResponse"
1590 type="wsrm:TerminateSequenceResponseType"/>
1591 <xs:complexType name="CreateSequenceType">
1592   <xs:sequence>
1593     <xs:element ref="wsrm:AcksTo"/>
1594     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1595     <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1596     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1597 maxOccurs="unbounded"/>
1598   </xs:sequence>
1599 </xs:complexType>

```

```

1599         <xs:documentation>
1600             It is the authors intent that this extensibility be used to
1601 transfer a Security Token Reference as defined in WS-Security.
1602         </xs:documentation>
1603     </xs:annotation>
1604 </xs:any>
1605 </xs:sequence>
1606     <xs:anyAttribute namespace="##other" processContents="lax"/>
1607 </xs:complexType>
1608 <xs:complexType name="CreateSequenceResponseType">
1609     <xs:sequence>
1610         <xs:element ref="wsrm:Identifier"/>
1611         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1612         <xs:element name="IncompleteSequenceBehavior"
1613 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1614         <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1615         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1616 maxOccurs="unbounded"/>
1617     </xs:sequence>
1618     <xs:anyAttribute namespace="##other" processContents="lax"/>
1619 </xs:complexType>
1620 <xs:complexType name="CloseSequenceType">
1621     <xs:sequence>
1622         <xs:element ref="wsrm:Identifier"/>
1623         <xs:element ref="wsrm:MessageNumberType"/>
1624         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1625 maxOccurs="unbounded"/>
1626     </xs:sequence>
1627     <xs:anyAttribute namespace="##other" processContents="lax"/>
1628 </xs:complexType>
1629 <xs:complexType name="CloseSequenceResponseType">
1630     <xs:sequence>
1631         <xs:element ref="wsrm:Identifier"/>
1632         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1633 maxOccurs="unbounded"/>
1634     </xs:sequence>
1635     <xs:anyAttribute namespace="##other" processContents="lax"/>
1636 </xs:complexType>
1637 <xs:complexType name="TerminateSequenceType">
1638     <xs:sequence>
1639         <xs:element ref="wsrm:Identifier"/>
1640         <xs:element ref="wsrm:MessageNumberType" minOccurs="0"/>
1641         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1642 maxOccurs="unbounded"/>
1643     </xs:sequence>
1644     <xs:anyAttribute namespace="##other" processContents="lax"/>
1645 </xs:complexType>
1646 <xs:complexType name="TerminateSequenceResponseType">
1647     <xs:sequence>
1648         <xs:element ref="wsrm:Identifier"/>
1649         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1650 maxOccurs="unbounded"/>
1651     </xs:sequence>
1652     <xs:anyAttribute namespace="##other" processContents="lax"/>
1653 </xs:complexType>
1654 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1655 <xs:complexType name="OfferType">
1656     <xs:sequence>
1657         <xs:element ref="wsrm:Identifier"/>
1658         <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1659         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1660         <xs:element name="IncompleteSequenceBehavior"
1661 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>

```

```

1662     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1663 maxOccurs="unbounded"/>
1664   </xs:sequence>
1665   <xs:anyAttribute namespace="##other" processContents="lax"/>
1666 </xs:complexType>
1667 <xs:complexType name="AcceptType">
1668   <xs:sequence>
1669     <xs:element ref="wsrm:AcksTo"/>
1670     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1671 maxOccurs="unbounded"/>
1672   </xs:sequence>
1673   <xs:anyAttribute namespace="##other" processContents="lax"/>
1674 </xs:complexType>
1675 <xs:element name="Expires">
1676   <xs:complexType>
1677     <xs:simpleContent>
1678       <xs:extension base="xs:duration">
1679         <xs:anyAttribute namespace="##other" processContents="lax"/>
1680       </xs:extension>
1681     </xs:simpleContent>
1682   </xs:complexType>
1683 </xs:element>
1684 <xs:simpleType name="IncompleteSequenceBehaviorType">
1685   <xs:restriction base="xs:string">
1686     <xs:enumeration value="DiscardEntireSequence"/>
1687     <xs:enumeration value="DiscardFollowingFirstGap"/>
1688     <xs:enumeration value="NoDiscard"/>
1689   </xs:restriction>
1690 </xs:simpleType>
1691 <xs:element name="UsesSequenceSTR">
1692   <xs:complexType>
1693     <xs:sequence/>
1694     <xs:anyAttribute namespace="##other" processContents="lax"/>
1695   </xs:complexType>
1696 </xs:element>
1697 <xs:element name="UsesSequenceSSL">
1698   <xs:complexType>
1699     <xs:sequence/>
1700     <xs:anyAttribute namespace="##other" processContents="lax"/>
1701   </xs:complexType>
1702 </xs:element>
1703 <xs:element name="UnsupportedElement">
1704   <xs:simpleType>
1705     <xs:restriction base="xs:QName"/>
1706   </xs:simpleType>
1707 </xs:element>
1708 </xs:schema>

```

1709 Appendix B. WSDL

1710 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1711 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1712 present in exchanges with that endpoint.

1713 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1714 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1715 for a higher-level mechanism to indicate that WS-RM is engaged.

1716 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1717 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

1718 The following non-normative copy is provided for reference.

```
1719 <?xml version="1.0" encoding="utf-8"?>
1720 <!--
1721 OASIS takes no position regarding the validity or scope of any intellectual
1722 property or other rights that might be claimed to pertain to the
1723 implementation or use of the technology described in this document or the
1724 extent to which any license under such rights might or might not be available;
1725 neither does it represent that it has made any effort to identify any such
1726 rights. Information on OASIS's procedures with respect to rights in OASIS
1727 specifications can be found at the OASIS website. Copies of claims of rights
1728 made available for publication and any assurances of licenses to be made
1729 available, or the result of an attempt made to obtain a general license or
1730 permission for the use of such proprietary rights by implementors or users of
1731 this specification, can be obtained from the OASIS Executive Director.
1732 OASIS invites any interested party to bring to its attention any copyrights,
1733 patents or patent applications, or other proprietary rights which may cover
1734 technology that may be required to implement this specification. Please
1735 address the information to the OASIS Executive Director.
1736 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
1737 This document and translations of it may be copied and furnished to others,
1738 and derivative works that comment on or otherwise explain it or assist in its
1739 implementation may be prepared, copied, published and distributed, in whole or
1740 in part, without restriction of any kind, provided that the above copyright
1741 notice and this paragraph are included on all such copies and derivative
1742 works. However, this document itself does not be modified in any way, such as
1743 by removing the copyright notice or references to OASIS, except as needed for
1744 the purpose of developing OASIS specifications, in which case the procedures
1745 for copyrights defined in the OASIS Intellectual Property Rights document must
1746 be followed, or as required to translate it into languages other than English.
1747 The limited permissions granted above are perpetual and will not be revoked by
1748 OASIS or its successors or assigns.
1749 This document and the information contained herein is provided on an "AS IS"
1750 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1751 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1752 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1753 FOR A PARTICULAR PURPOSE.
1754 -->
1755 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1756 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1757 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
1758 open.org/ws-rx/wsrn/200608" xmlns:tns="http://docs.oasis-open.org/ws-
1759 rx/wsrn/200608/wsd" targetNamespace="http://docs.oasis-open.org/ws-
1760 rx/wsrn/200608/wsd">
```

```
1761 <wsdl:types>
```



```

1762     <xs:schema>
1763         <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1764         schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-
1765         200608.xsd"/>
1766     </xs:schema>
1767 </wsdl:types>

1768     <wsdl:message name="CreateSequence">
1769         <wsdl:part name="create" element="rm:CreateSequence"/>
1770     </wsdl:message>
1771     <wsdl:message name="CreateSequenceResponse">
1772         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1773     </wsdl:message>
1774     <wsdl:message name="CloseSequence">
1775         <wsdl:part name="close" element="rm:CloseSequence"/>
1776     </wsdl:message>
1777     <wsdl:message name="CloseSequenceResponse">
1778         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1779     </wsdl:message>
1780     <wsdl:message name="TerminateSequence">
1781         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1782     </wsdl:message>
1783     <wsdl:message name="TerminateSequenceResponse">
1784         <wsdl:part name="terminateResponse"
1785         element="rm:TerminateSequenceResponse"/>
1786     </wsdl:message>

1787     <wsdl:portType name="SequenceAbstractPortType">
1788         <wsdl:operation name="CreateSequence">
1789             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1790             open.org/ws-rx/wsrn/200608/CreateSequence"/>
1791             <wsdl:output message="tns:CreateSequenceResponse"
1792             wsaw:Action="http://docs.oasis-open.org/ws-
1793             rx/wsrn/200608/CreateSequenceResponse"/>
1794         </wsdl:operation>
1795         <wsdl:operation name="CloseSequence">
1796             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1797             open.org/ws-rx/wsrn/200608/CloseSequence"/>
1798             <wsdl:output message="tns:CloseSequenceResponse"
1799             wsaw:Action="http://docs.oasis-open.org/ws-
1800             rx/wsrn/200608/CloseSequenceResponse"/>
1801         </wsdl:operation>
1802         <wsdl:operation name="TerminateSequence">
1803             <wsdl:input message="tns:TerminateSequence"
1804             wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence"/>
1805             <wsdl:output message="tns:TerminateSequenceResponse"
1806             wsaw:Action="http://docs.oasis-open.org/ws-
1807             rx/wsrn/200608/TerminateSequenceResponse"/>
1808         </wsdl:operation>
1809     </wsdl:portType>

1810 </wsdl:definitions>

```

1811 Appendix C. Message Examples

1812 Appendix C.1 Create Sequence

1813 Create Sequence

```
1814 <?xml version="1.0" encoding="UTF-8"?>
1815 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1816 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1817 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1818   <S:Header>
1819     <wsa:MessageID>
1820       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1821     </wsa:MessageID>
1822     <wsa:To>http://example.com/serviceB/123</wsa:To>
1823     <wsa:Action>http://docs.oasis-open.org/ws-
1824 rx/wsmr/200608/CreateSequence</wsa:Action>
1825     <wsa:ReplyTo>
1826       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1827     </wsa:ReplyTo>
1828   </S:Header>
1829   <S:Body>
1830     <wsmr:CreateSequence>
1831       <wsmr:AcksTo>
1832         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1833       </wsmr:AcksTo>
1834     </wsmr:CreateSequence>
1835   </S:Body>
1836 </S:Envelope>
```

1837 Create Sequence Response

```
1838 <?xml version="1.0" encoding="UTF-8"?>
1839 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1840 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1841 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1842   <S:Header>
1843     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1844     <wsa:RelatesTo>
1845       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1846     </wsa:RelatesTo>
1847     <wsa:Action>
1848       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
1849     </wsa:Action>
1850   </S:Header>
1851   <S:Body>
1852     <wsmr:CreateSequenceResponse>
1853       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1854     </wsmr:CreateSequenceResponse>
1855   </S:Body>
1856 </S:Envelope>
```

1857 Appendix C.2 Initial Transmission

1858 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1859 figure. The three messages have the following headers; the third message is identified as the last
1860 message in the Sequence:

1861 Message 1

```

1862 <?xml version="1.0" encoding="UTF-8"?>
1863 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1864 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1865 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1866   <S:Header>
1867     <wsa:MessageID>
1868       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1869     </wsa:MessageID>
1870     <wsa:To>http://example.com/serviceB/123</wsa:To>
1871     <wsa:From>
1872       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1873     </wsa:From>
1874     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1875     <wsmr:Sequence>
1876       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1877       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1878     </wsmr:Sequence>
1879   </S:Header>
1880   <S:Body>
1881     <!-- Some Application Data -->
1882   </S:Body>
1883 </S:Envelope>

```

1884 Message 2

```

1885 <?xml version="1.0" encoding="UTF-8"?>
1886 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1887 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1888 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1889   <S:Header>
1890     <wsa:MessageID>
1891       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1892     </wsa:MessageID>
1893     <wsa:To>http://example.com/serviceB/123</wsa:To>
1894     <wsa:From>
1895       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1896     </wsa:From>
1897     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1898     <wsmr:Sequence>
1899       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1900       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1901     </wsmr:Sequence>
1902   </S:Header>
1903   <S:Body>
1904     <!-- Some Application Data -->
1905   </S:Body>
1906 </S:Envelope>

```

1907 Message 3

```

1908 <?xml version="1.0" encoding="UTF-8"?>
1909 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1910 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1911 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1912   <S:Header>
1913     <wsa:MessageID>
1914       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1915     </wsa:MessageID>
1916     <wsa:To>http://example.com/serviceB/123</wsa:To>
1917     <wsa:From>
1918       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1919     </wsa:From>
1920     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1921     <wsmr:Sequence>
1922       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>

```

```

1923     <wsrm:MessageNumber>3</wsrm:MessageNumber>
1924     </wsrm:Sequence>
1925     <wsrm:AckRequested>
1926         <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1927     </wsrm:AckRequested>
1928 </S:Header>
1929 <S:Body>
1930     <!-- Some Application Data -->
1931 </S:Body>
1932 </S:Envelope>

```

1933 **Appendix C.3 First Acknowledgement**

1934 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1935 responds with an Acknowledgement for messages 1 and 3:

```

1936 <?xml version="1.0" encoding="UTF-8"?>
1937 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1938 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
1939 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1940   <S:Header>
1941     <wsa:MessageID>
1942       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1943     </wsa:MessageID>
1944     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1945     <wsa:From>
1946       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1947     </wsa:From>
1948     <wsa:Action>
1949       http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
1950     </wsa:Action>
1951     <wsrm:SequenceAcknowledgement>
1952       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1953       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1954       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1955     </wsrm:SequenceAcknowledgement>
1956   </S:Header>
1957   <S:Body/>
1958 </S:Envelope>

```

1959 **Appendix C.4 Retransmission**

1960 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
1961 requests an Acknowledgement:

```

1962 <?xml version="1.0" encoding="UTF-8"?>
1963 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1964 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
1965 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1966   <S:Header>
1967     <wsa:MessageID>
1968       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1969     </wsa:MessageID>
1970     <wsa:To>http://example.com/serviceB/123</wsa:To>
1971     <wsa:From>
1972       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1973     </wsa:From>
1974     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1975     <wsrm:Sequence>
1976       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1977       <wsrm:MessageNumber>2</wsrm:MessageNumber>
1978     </wsrm:Sequence>

```

```

1979 <wsrm:AckRequested>
1980 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1981 </wsrm:AckRequested>
1982 </S:Header>
1983 <S:Body>
1984 <!-- Some Application Data -->
1985 </S:Body>
1986 </S:Envelope>

```

1987 Appendix C.5 Termination

1988 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
1989 be terminated:

```

1990 <?xml version="1.0" encoding="UTF-8"?>
1991 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1992 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
1993 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1994 <S:Header>
1995 <wsa:MessageID>
1996 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
1997 </wsa:MessageID>
1998 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1999 <wsa:From>
2000 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2001 </wsa:From>
2002 <wsa:Action>
2003 http://docs.oasis-open.org/ws-rx/wsrm/200608/SequenceAcknowledgement
2004 </wsa:Action>
2005 <wsrm:SequenceAcknowledgement>
2006 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2007 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2008 </wsrm:SequenceAcknowledgement>
2009 </S:Header>
2010 <S:Body/>
2011 </S:Envelope>

```

2012 Terminate Sequence

```

2013 <?xml version="1.0" encoding="UTF-8"?>
2014 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2015 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
2016 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2017 <S:Header>
2018 <wsa:MessageID>
2019 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2020 </wsa:MessageID>
2021 <wsa:To>http://example.com/serviceB/123</wsa:To>
2022 <wsa:Action>
2023 http://docs.oasis-open.org/ws-rx/wsrm/200608/TerminateSequence
2024 </wsa:Action>
2025 <wsa:From>
2026 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2027 </wsa:From>
2028 </S:Header>
2029 <S:Body>
2030 <wsrm:TerminateSequence>
2031 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2032 <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2033 </wsrm:TerminateSequence>
2034 </S:Body>
2035 </S:Envelope>

```

2036 Terminate Sequence Response

```
2037 <?xml version="1.0" encoding="UTF-8"?>
2038 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2039 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2040 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2041   <S:Header>
2042     <wsa:MessageID>
2043       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2044     </wsa:MessageID>
2045     <wsa:To>http://example.com/serviceA/789</wsa:To>
2046     <wsa:Action>
2047       http://docs.oasis-open.org/ws-rx/wsmr/200608/TerminateSequenceResponse
2048     </wsa:Action>
2049     <wsa:RelatesTo>
2050       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2051     </wsa:RelatesTo>
2052     <wsa:From>
2053       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2054     </wsa:From>
2055   </S:Header>
2056   <S:Body>
2057     <wsmr:TerminateSequenceResponse>
2058       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
2059     </wsmr:TerminateSequenceResponse>
2060   </S:Body>
2061 </S:Envelope>
```

2062 Appendix D. State Tables

2063 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2064 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2065 Legend:

2066 The first column of these tables contains the motivating event and has the following format:

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

2067 Where:

- 2068 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2069 described by the specification.
- 2070 ● [source]: indicates the source of the event; one of:
 - 2071 ● [msg] a Received message
 - 2072 ● [int]: an internal event such as the firing of a timer
 - 2073 ● [app]: the application
 - 2074 ● [unspec]: the source is unspecified

2075 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}

2076 Where:

- 2077 ● action to take: indicates that the state machine performs the following action. Actions surrounded
2078 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2079 "Transmit"
- 2080 ● [next state]: indicates the state to which the state machine will advance upon the performance of
2081 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2082 ● {ref} is a reference to the document section describing the behavior in this cell

2083 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2084 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2085 described in this specification and does not indicate normal protocol operation. Implementations MAY
2086 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2087 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2088 combinations.

2089 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
Close Sequence Response [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}

2090 Table 2 RM Destination Sequence State Transition Table

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

Events	Sequence States			
	None	Created	Closed	Terminating
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}	Generate Sequence Terminated Fault [Same] {4.2}
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}	Generate Sequence Terminated Fault [Same] {4.2}
<AckRequested> [msg] {3.8}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.8}	Xmit SeqAck+Final [Same] {3.9}	Generate Sequence Terminated Fault [Same] {4.2}
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
<CloseSequence autonomously> [int]		Xmit CloseSequence with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence with SeqAck+Final [Same] {3.5}	
CloseSequenceResponse [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}		No Action [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
TerminateSequence [msg] {3.6}	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}
<TerminateSequence autonomously> [int]		Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}	Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}	Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}
TerminateSequenceResponse [msg]	Generate Unknown Sequence Fault [Same] {4.3}			Terminate Sequence [None]
UnknownSequence Fault [msg] {4.3}		Terminate Sequence [None] {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}	Terminate Sequence [None] {4.3}
Invalid Acknowledgement Fault [msg] {4.4}	N/A			

Events	Sequence States			
	None	Created	Closed	Terminating
Expires exceeded [int] {3.9}	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}	

2091 **Appendix E. Acknowledgments**

2092 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2093 authors:

2094 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),
2095 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),
2096 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),
2097 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don
2098 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),
2099 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony
2100 Storey(IBM).

2101 The following individuals have provided invaluable input into the initial contribution:

2102 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen Brown
2103 (Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM), Paul
2104 Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott
2105 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal
2106 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter
2107 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish
2108 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2109 The following individuals were members of the committee during the development of this specification:

2110 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben Malek
2111 (Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd Burch
2112 (Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul Cotton
2113 (Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques Durand
2114 (Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2), Robert
2115 Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair Green(Choreology),
2116 Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu), Chamikara Jayalath
2117 (WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul Knight(Nortel), Dan
2118 Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt Lovett(IBM), Ashok Malhotra
2119 (Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff Mischkinsky(Oracle), Nilo Mitra
2120 (Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku Nishiyama(Hitachi), Dave Orchard
2121 (BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert Pilz(BEA), Martin Raeppe(SAP),
2122 Eric Rajkovic(Oracle), Stefan Rossmann(SAP), Tom Rutt(Fujitsu), Rich Salz(IBM), Shivajee
2123 Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von Riegen(SAP), Pete Wenzel(Sun), Steve
2124 Winkler(SAP), Ümit Yalçınalp(SAP), Nobuyuki 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

Rev	Date	By Whom	What
wd-08	2005-12-27	Gilbert Pilz	Moved schema and WSDL files to their own artifacts. Converted source document to 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 RDDL doc; added non-normative reference to RDDL 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

Rev	Date	By Whom	What
wd-11	2006-03-08	Doug Davis	Issue 100 applied
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

Rev	Date	By Whom	What
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
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

Rev	Date	By Whom	What
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.
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 (and PR013)
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
wd-16	2007-01-17	Doug Davis	PR026 applied
wd-16	2007-01-18	Doug Davis	PR021 applied
wd-16	2007-01-18	Doug Davis	PR022 applied
wd-16	2007-01-18	Doug Davis	Fixed a few typos (Doug, Gil)
wd-16	2007-01-18	Gilbert Pilz	PR007 applied
wd-16	2007-01-25	Doug Davis	PR039 applied

2126 **Appendix G. Notices**

2127 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2128 might be claimed to pertain to the implementation or use of the technology described in this document or
2129 the extent to which any license under such rights might or might not be available; neither does it represent
2130 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2131 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2132 available for publication and any assurances of licenses to be made available, or the result of an attempt
2133 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2134 users of this specification, can be obtained from the OASIS Executive Director.

2135 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2136 other proprietary rights which may cover technology that may be required to implement this specification.
2137 Please address the information to the OASIS Executive Director.

2138 Copyright (C) OASIS Open (2006). All Rights Reserved.

2139 This document and translations of it may be copied and furnished to others, and derivative works that
2140 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2141 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2142 this paragraph are included on all such copies and derivative works. However, this document itself may
2143 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2144 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2145 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2146 into languages other than English.

2147 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2148 or assigns.

2149 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2150 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2151 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2152 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.