



1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

3 Working Draft 16, November 20, 2006

4 Document identifier:

5 wsrn-1.1-spec-wd-16

6 Location:

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

8 Editors:

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

14 Contributors:

15 See the Acknowledgments (Appendix E).

16 Abstract:

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

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

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

31 Status:

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

42 Table of Contents

43	1 Introduction.....	4
44	1.1 Notational Conventions.....	4
45	1.2 Namespace.....	5
46	1.3 Compliance.....	5
47	2 Reliable Messaging Model.....	6
48	2.1 Glossary.....	6
49	2.2 Protocol Preconditions.....	7
50	2.3 Protocol Invariants.....	7
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	3.10 MakeConnection.....	22
63	3.11 MessagePending.....	24
64	4 Faults.....	26
65	4.1 SequenceFault Element.....	27
66	4.2 Sequence Terminated.....	28
67	4.3 Unknown Sequence.....	28
68	4.4 Invalid Acknowledgement.....	29
69	4.5 Message Number Rollover.....	29
70	4.6 Create Sequence Refused.....	30
71	4.7 Sequence Closed.....	30
72	4.8 WSRM Required.....	31
73	4.9 Unsupported Selection.....	31
74	5 Security Threats and Countermeasures.....	33
75	5.1 Threats and Countermeasures.....	33
76	5.1.1 Integrity Threats.....	33
77	5.1.1.1 Countermeasures.....	33
78	5.1.2 Resource Consumption Threats.....	34
79	5.1.2.1 Countermeasures.....	34

80	5.1.3 Sequence Spoofing Threats.....	34
81	5.1.3.1 Sequence Hijacking.....	34
82	5.1.3.2 Countermeasures.....	34
83	5.2 Security Solutions and Technologies.....	35
84	5.2.1 Transport Layer Security.....	35
85	5.2.1.1 Model.....	35
86	5.2.1.2 Countermeasure Implementation.....	36
87	5.2.2 SOAP Message Security.....	37
88	5.2.2.1 Model.....	37
89	5.2.2.2 Countermeasure Implementation.....	37
90	6 Securing Sequences.....	39
91	6.1 Securing Sequences Using WS-Security.....	39
92	6.2 Securing Sequences Using SSL/TLS.....	40
93	7 References.....	42
94	7.1 Normative.....	42
95	7.2 Non-Normative.....	43
96	Appendix A. Schema.....	45
97	Appendix B. WSDL.....	50
98	Appendix C. Message Examples.....	52
99	Appendix C.1 Create Sequence.....	52
100	Appendix C.2 Initial Transmission.....	52
101	Appendix C.3 First Acknowledgement.....	54
102	Appendix C.4 Retransmission.....	54
103	Appendix C.5 Termination.....	55
104	Appendix C.6 MakeConnection.....	56
105	Appendix D. State Tables.....	60
106	Appendix E. Acknowledgments.....	65
107	Appendix F. Revision History.....	66
108	Appendix G. Notices.....	72

109 1 Introduction

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

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

119 1.1 Notational Conventions

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

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

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

138 Elements and Attributes defined by this specification are referred to in the text of this document using
139 XPath 1.0 [XPATH 1.0] expressions. Extensibility points are referred to using an extended version of this
140 syntax:

- 141 • An element extensibility point is referred to using {any} in place of the element name. This
142 indicates that any element name can be used, from any namespace other than the wsrn:
143 namespace.
- 144 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
145 indicates that any attribute name can be used, from any namespace other than the wsrn:
146 namespace.

147 **1.2 Namespace**

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

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

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

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

154 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

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

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

158 **1.3 Conformance**

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

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

2 Reliable Messaging Model

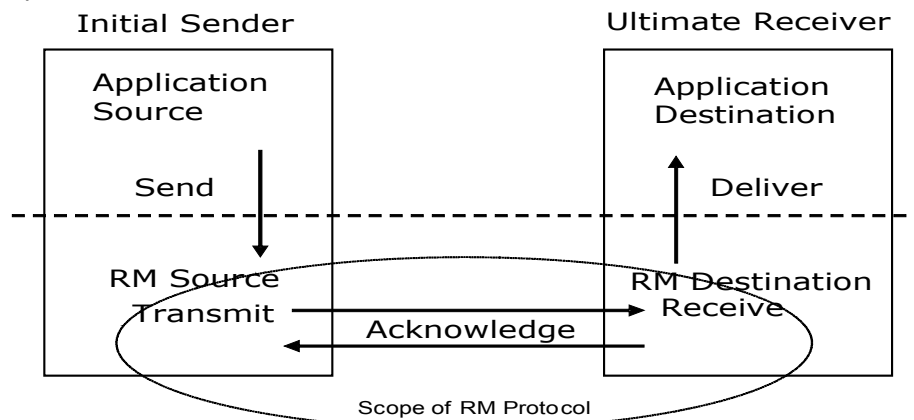
165

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

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

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

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



191 Figure 1: Reliable Messaging Model

2.1 Glossary

192

193 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

222 **2.2 Protocol Preconditions**

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

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

232 2.3 Protocol Invariants

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

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

244 2.4 Example Message Exchange

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

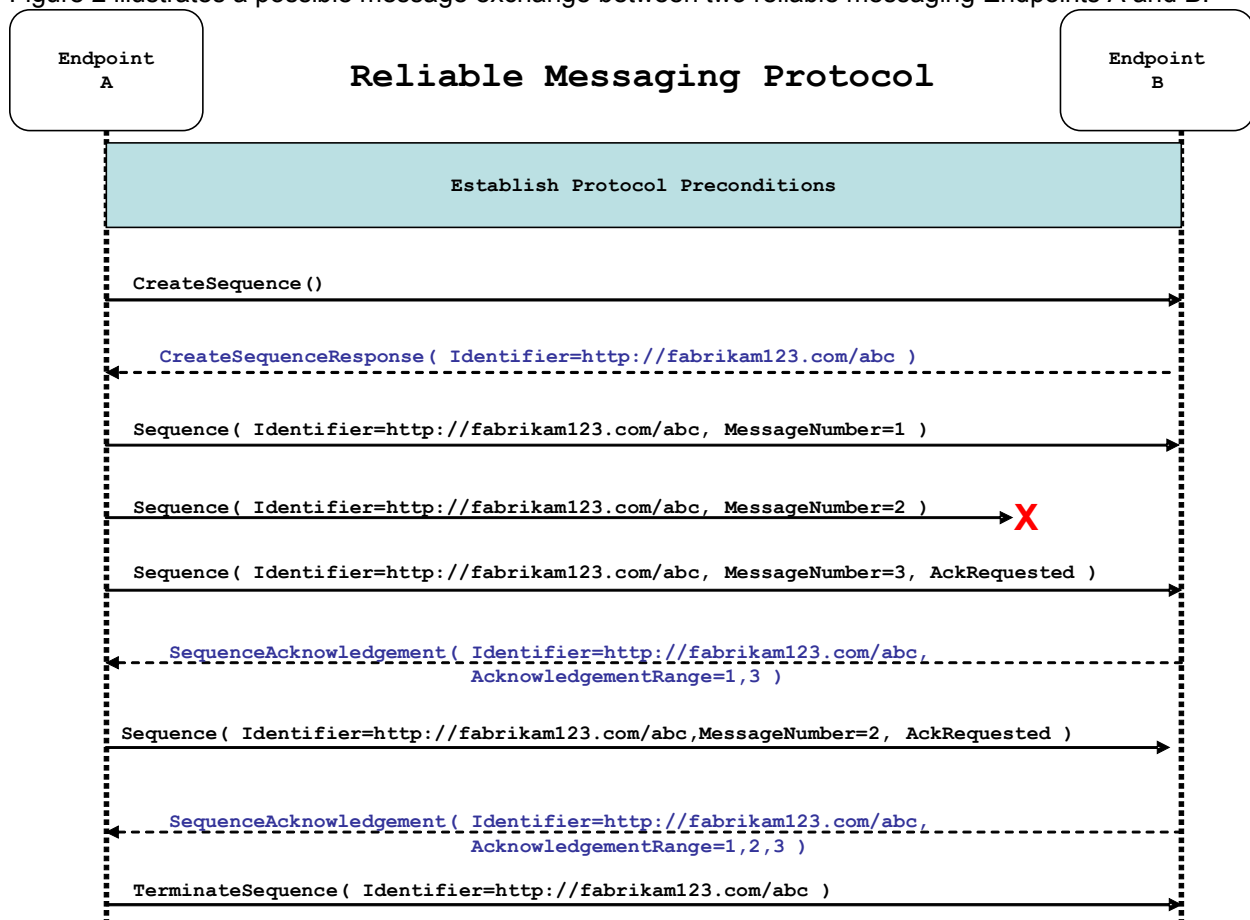


Figure 2: The WS-ReliableMessaging Protocol

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

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

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

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

285 **3 RM Protocol Elements**

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

288 **3.1 Considerations on the Use of Extensibility Points**

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

293 **3.2 Considerations on the Use of "Piggy-Backing"**

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

303 **3.3 Composition with WS-Addressing**

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

- 306 1. When an Endpoint generates a message that carries an RM protocol element, that is defined in
307 section 3 below, in the body of a SOAP envelope that Endpoint MUST include in that envelope a
308 `wsa:Action` SOAP header block whose value is an IRI that is a concatenation of the WS-RM
309 namespace URI, followed by a "/", followed by the value of the local name of the child element of
310 the SOAP body. For example, for a Sequence creation request message as described in section
311 3.4 below, the value of the `wsa:Action` IRI would be:

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

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

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

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

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

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

321 3.4 Sequence Creation

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

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

329 The following exemplar defines the `CreateSequence` syntax:

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

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

345 `/wsrm:CreateSequence`

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

350 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

360 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

368 `/wsmr:CreateSequence/wsmr:Offer`

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

371 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier`

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

374 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier/@{any}`

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

377 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint`

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

382 Implementations MUST NOT use an endpoint reference in the Endpoint element that would prevent the
383 sending of Sequence Lifecycle Message, Sequence Traffic Message, etc. For example, using the WS-
384 Addressing "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM
385 Destination to ever send Sequence Lifecycle Messages (e.g. `TerminateSequence`) to the RM Source
386 for the Offered Sequence. Implementations MAY use the WS-RM anonymous URI template and doing so
387 implies that messages will be retrieved using a mechanism such as the `MakeConnection` message (see
388 section 3.10).

389 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Expires`

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

393 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Expires/@{any}`

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

396 `/wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior`

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

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

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

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

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

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

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

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

413 /wsmr:CreateSequence/{any}

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

416 /wsmr:CreateSequence/@{any}

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

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

423 The following exemplar defines the `CreateSequenceResponse` syntax:

```
424 <wsmr:CreateSequenceResponse ...>  
425   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
426   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
427   <wsmr:IncompleteSequenceBehavior>  
428     wsmr:IncompleteSequenceBehaviorType  
429   </wsmr:IncompleteSequenceBehavior> ?  
430   <wsmr:Accept ...>  
431     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
432     ...  
433   </wsmr:Accept> ?  
434   ...  
435 </wsmr:CreateSequenceResponse>
```

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

437 /wsmr:CreateSequenceResponse

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

441 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

448 /wsmr:CreateSequenceResponse/wsmr:Expires

449 This element, if present, of type `xs:duration` accepts or refines the RM Source's requested duration for
450 the Sequence. It specifies the amount of time after which any resources associated with the Sequence

451 SHOULD be reclaimed thus causing the Sequence to be silently terminated. At the RM Destination this
452 duration is measured from a point proximate to Sequence creation and at the RM Source this duration is
453 measured from a point approximate to the successful processing of the `CreateSequenceResponse`. A
454 value of "PT0S" indicates that the Sequence will never expire. Absence of the element indicates an
455 implied value of "PT0S". The RM Destination MUST set the value of this element to be equal to or less
456 than the value requested by the RM Source in the corresponding `CreateSequence` message.

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

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

460 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

471 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

477 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

492 /wsmr:CreateSequenceResponse/{any}

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

495 /wsmr:CreateSequenceResponse/@{any}

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

498 **3.5 Closing A Sequence**

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

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

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

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

521 The following exemplar defines the `CloseSequence` syntax:

```
522 <wsmr:CloseSequence ...>  
523   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
524   ...  
525 </wsmr:CloseSequence>
```

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

527 /wsmr:CloseSequence

528 This element is sent by an RM Source to indicate that the RM Destination MUST NOT accept any new
529 messages for this Sequence.

530 /wsmr:CloseSequence/wsmr:Identifier

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

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

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

537 /wsmr:CloseSequence/{any}

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

540 /wsmr:CloseSequence@{any}

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

543 A `CloseSequenceResponse` is sent in the body of a response message by an RM Destination in
544 response to receipt of a `CloseSequence` request message. It indicates that the RM Destination has
545 closed the Sequence.

546 The following exemplar defines the `CloseSequenceResponse` syntax:

```
547 <wsmr:CloseSequenceResponse ...>  
548   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
549   ...  
550 </wsmr:CloseSequenceResponse>
```

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

552 /wsmr:CloseSequenceResponse

553 This element is sent in the body of a response message by an RM Destination in response to receipt of a
554 `CloseSequence` request message. It indicates that the RM Destination has closed the Sequence.

555 /wsmr:CloseSequenceResponse/wsmr:Identifier

556 The RM Destination MUST include this element in any `CloseSequenceResponse` message it sends. The
557 RM Destination MUST set the value of this element to the absolute URI (conformant with RFC3986) of the
558 Sequence that is being closed.

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

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

562 /wsmr:CloseSequenceResponse/{any}

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

565 /wsmr:CloseSequenceResponse@{any}

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

568 **3.6 Sequence Termination**

569 When the RM Source has completed its use of the Sequence it sends a `TerminateSequence` element,
570 in the body of a message, to the RM Destination to indicate that the Sequence is complete and that it will
571 not be sending any further messages related to the Sequence. The RM Destination can safely reclaim any
572 resources associated with the Sequence upon receipt of the `TerminateSequence` message. Under
573 normal usage the RM Source will complete its use of the Sequence when all of the messages in the

574 Sequence have been acknowledged. However, the RM Source is free to Terminate or Close a Sequence
575 at any time regardless of the acknowledgement state of the messages.

576 The following exemplar defines the TerminateSequence syntax:

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

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

582 /wsrm:TerminateSequence

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

588 /wsrm:TerminateSequence/wsrm:Identifier

589 The RM Source MUST include this element in any TerminateSequence message it sends. The RM
590 Source MUST set the value of this element to the absolute URI (conformant with RFC3986) of the
591 Sequence that is being terminated.

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

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

595 /wsrm:TerminateSequence/{any}

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

598 /wsrm:TerminateSequence/@{any}

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

601 A TerminateSequenceResponse is sent in the body of a response message by an RM Destination in
602 response to receipt of a TerminateSequence request message. It indicates that the RM Destination has
603 terminated the Sequence.

604 The following exemplar defines the TerminateSequenceResponse syntax:

```
605 <wsrm:TerminateSequenceResponse ...>  
606   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
607   ...  
608 </wsrm:TerminateSequenceResponse>
```

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

610 /wsrm:TerminateSequenceResponse

611 This element is sent in the body of a response message by an RM Destination in response to receipt of a
612 TerminateSequence request message. It indicates that the RM Destination has terminated the
613 Sequence. The RM Destination MUST NOT send this element as a header block.

614 /wsrm:TerminateSequenceResponse/wsrm:Identifier

615 The RM Destination MUST include this element in any `TerminateSequenceResponse` message it
616 sends. The RM Destination MUST set the value of this element to the absolute URI (conformant with
617 RFC3986) of the Sequence that is being terminated.

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

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

621 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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

627 On receipt of a `TerminateSequence` message an RM Destination MUST respond with a corresponding
628 `TerminateSequenceResponse` message or generate a fault `UnknownSequenceFault` if the
629 Sequence is not known.

630 **3.7 Sequences**

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

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

638 A following exemplar defines its syntax:

```
639 <wsrm:Sequence ...>  
640   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
641   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
642   ...  
643 </wsrm:Sequence>
```

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

645 `/wsrm:Sequence`

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

652 `/wsrm:Sequence/wsrm:Identifier`

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

656 /wsmr:Sequence/wsmr:Identifier/@{any}

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

659 /wsmr:Sequence/wsmr:MessageNumber

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

664 /wsmr:Sequence/{any}

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

667 /wsmr:Sequence/@{any}

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

670 The following example illustrates a Sequence header block.

```
671 <wsmr:Sequence>  
672   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
673   <wsmr:MessageNumber>10</wsmr:MessageNumber>  
674 </wsmr:Sequence>
```

675 **3.8 Request Acknowledgement**

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

678 The RM Source **MAY** request an Acknowledgement Message from the RM Destination at any time by
679 independently transmitting an `AckRequested` header block (i.e. as a header of a SOAP envelope with an
680 empty body). Alternately the RM Source **MAY** include an `AckRequested` header block in any message
681 targeted to the RM Destination. The RM Destination **SHOULD** process any `AckRequested` header
682 blocks that may be included in any message it receives. If a non-mustUnderstand fault occurs when
683 processing an `AckRequested` header block that was piggy-backed, a fault **MUST** be generated, but the
684 processing of the original message **MUST NOT** be affected.

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

690 The following exemplar defines its syntax:

```
691 <wsmr:AckRequested ...>  
692   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
693   ...  
694 </wsmr:AckRequested>
```

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

696 /wsmr:AckRequested

697 This element requests an Acknowledgement for the identified Sequence.

698 /wsmr:AckRequested/wsmr:Identifier
699 An RM Source that includes an `AckRequested` header block in a SOAP envelope MUST include this
700 element in that header block. The RM Source MUST set the value of this element to the absolute URI,
701 (conformant with RFC3986), that uniquely identifies the Sequence to which the request applies.

702 /wsmr:AckRequested/wsmr:Identifier/@{any}
703 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
704 element.

705 /wsmr:AckRequested/{any}
706 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
707 to be passed.

708 /wsmr:AckRequested/@{any}
709 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
710 element.

711 3.9 Sequence Acknowledgement

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

715 The RM Destination MAY Transmit the `SequenceAcknowledgement` header block independently (i.e. as
716 a header of a SOAP envelope with an empty body). Alternatively, an RM Destination MAY include a
717 `SequenceAcknowledgement` header block on any SOAP envelope targeted to the endpoint referenced
718 by the `AcksTo` EPR. The RM Source SHOULD process any `SequenceAcknowledgement` header
719 blocks that may be included in any message it receives. If a non-mustUnderstand fault occurs when
720 processing a `SequenceAcknowledgement` header that was piggy-backed, a fault MUST be generated,
721 but the processing of the original message MUST NOT be affected.

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

732 The following exemplar defines its syntax:

```
733 <wsmr:SequenceAcknowledgement ...>  
734   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
735   [ [ [ <wsmr:AcknowledgementRange ...  
736         Upper="wsmr:MessageNumberType"  
737         Lower="wsmr:MessageNumberType" /> +  
738         | <wsmr:None/> ]  
739         <wsmr:Final/> ? ]  
740         | <wsmr:Nack> wsmr:MessageNumberType </wsmr:Nack> + ]  
741   ]  
742   ...
```

743 `</wsrm:SequenceAcknowledgement>`

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

745 `/wsrm:SequenceAcknowledgement`

746 This element contains the Sequence Acknowledgement information.

747 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

756 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

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

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

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

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

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

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

771 `/wsrm:SequenceAcknowledgement/wsrm:None`

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

776 `/wsrm:SequenceAcknowledgement/wsrm:Final`

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

783 /wsmr:SequenceAcknowledgement/wsmr:Nack

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

793 /wsmr:SequenceAcknowledgement/{any}

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

796 /wsmr:SequenceAcknowledgement/@{any}

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

799 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
801 <wsmr:SequenceAcknowledgement>  
802   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
803   <wsmr:AcknowledgementRange Upper="10" Lower="1"/>  
804 </wsmr:SequenceAcknowledgement>
```

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

```
807 <wsmr:SequenceAcknowledgement>  
808   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
809   <wsmr:AcknowledgementRange Upper="2" Lower="1"/>  
810   <wsmr:AcknowledgementRange Upper="6" Lower="4"/>  
811   <wsmr:AcknowledgementRange Upper="10" Lower="8"/>  
812 </wsmr:SequenceAcknowledgement>
```

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

```
814 <wsmr:SequenceAcknowledgement>  
815   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
816   <wsmr:Nack>3</wsmr:Nack>  
817 </wsmr:SequenceAcknowledgement>
```

818 3.10 MakeConnection

819 When an Endpoint is not directly addressable (e.g. behind a firewall or not able to allow incoming
820 connections), an anonymous URI in the EPR address property can indicate such an Endpoint. The WS-
821 Addressing anonymous URI is one such anonymous URI. This specification defines a URI template (the
822 WS-RM anonymous URI) which may be used to uniquely identify anonymous Endpoints.

```
823 http://docs.oasis-open.org/ws-rx/wsmr/200608/anonymous?id={uuid}
```

824 This URI template in an EPR indicates a protocol-specific back-channel will be established through a
825 mechanism such as `MakeConnection`, defined below. When using this URI template, "{uuid}" MUST be
826 replaced by a UUID value as defined by RFC4122[UUID]. This UUID value uniquely distinguishes the

827 Endpoint. A sending Endpoint SHOULD Transmit messages at Endpoints identified with the URI template
828 using a protocol-specific back-channel, including but not limited to those established with a
829 `MakeConnection` message. Note, this URI is semantically similar to the WS-Addressing anonymous
830 URI if a protocol-specific back-channel is available.

831 The `MakeConnection` element is sent in the body of a one-way message that establishes a
832 contextualized back-channel for the transmission of messages according to matching criteria (defined
833 below). In the non-faulting case, if no matching message is available then no SOAP envelope will be
834 returned on the back-channel. A common usage will be a client RM Destination sending
835 `MakeConnection` to a server RM Source for the purpose of receiving asynchronous response
836 messages.

837 The following exemplar defines the `MakeConnection` syntax:

```
838 <wsrm:MakeConnection ...>  
839   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
840   <wsrm:Address ...> xs:anyURI </wsrm:Address> ?  
841   ...  
842 </wsrm:MakeConnection>
```

843 The following describes the content model of the `MakeConnection` element.

844 `/wsrm:MakeConnection`

845 This element allows the sender to create a transport-specific back-channel that can be used to return a
846 message that matches the selection criteria. Endpoints MUST NOT send this element as a header block.

847 `/wsrm:MakeConnection/wsrm:Identifier`

848 This element specifies the WS-RM Sequence Identifier that establishes the context for the transport-
849 specific back-channel. The Sequence Identifier should be compared with the Sequence Identifiers
850 associated with the messages held by the sending Endpoint, and if there is a matching message it will be
851 returned. If this element is omitted from the message then the `Address` MUST be included in the
852 message.

853 `/wsrm:MakeConnection/wsrm:Identifier/@{any}`

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

856 `/wsrm:MakeConnection/wsrm:Address`

857 This element specifies the URI (`wsa:Address`) of the initiating Endpoint. Endpoints MUST NOT return
858 messages on the transport-specific back-channel unless they have been addressed to this URI. This
859 `Address` property and a message's WS-Addressing destination property are considered identical when
860 they are exactly the same character-for-character. Note that URIs which are not identical in this sense
861 may in fact be functionally equivalent. Examples include URI references which differ only in case, or
862 which are in external entities which have different effective base URIs. If this element is omitted from the
863 message then the `Identifier` MUST be included in the message.

864 `/wsrm:MakeConnection/wsrm:Address/@{any}`

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

867 `/wsrm:MakeConnection/{any}`

868 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
869 to be passed. This allows fine-tuning of the messages to be returned, additional selection criteria included

870 here are logically ANDed with the `Address` and/or `Identifier`. If an extension is not supported by the
871 Endpoint then it should generate an `UnsupportedSelection` fault.

872 `/wsrm:MakeConnection/@{any}`

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

875 If both `Identifier` and `Address` are present, then the Endpoint processing the `MakeConnection`
876 message MUST insure that any SOAP Envelope flowing on the back-channel MUST be associated with
877 the given `Sequence` and MUST be addressed to the given URI.

878 The management of messages that are awaiting the establishment of a back-channel to their receiving
879 Endpoint is an implementation detail that is outside the scope of this specification. Note, however, that
880 these messages form a class of asynchronous messages that is not dissimilar from "ordinary"
881 asynchronous messages that are waiting for the establishment of a connection to their destination
882 Endpoints.

883 This specification places no constraint on the types of messages that can be returned on the transport-
884 specific back-channel. As in an asynchronous environment, it is up to the recipient of the
885 `MakeConnection` message to decide which messages are appropriate for transmission to any particular
886 Endpoint. However, the Endpoint processing the `MakeConnection` message MUST insure that the
887 messages match the selection criteria as specified by the child elements of the `MakeConnection`
888 element.

889 Since the message exchange pattern use by `MakeConnection` is untraditional, the following points need
890 to be reiterated for clarification:

- 891 ● The `MakeConnection` message is logically part of a one-way operation; there is no reply
892 message to the `MakeConnection` itself, and any response flowing on the transport back-channel
893 is a pending message.
- 894 ● Since there is no reply message to `MakeConnection`, the WS-Addressing specific rules in
895 section 3.4 "Formulating a Reply Message" are not used. Therefore, the value of any
896 `wsa:ReplyTo` element in the `MakeConnection` message has no effective impact since the WS-
897 Addressing [`reply endpoint`] property that is set by the presence of `wsa:ReplyTo` is not
898 used.
- 899 ● In the absence of any pending message, there will be no message transmitted on the transport
900 back-channel. E.g. In the HTTP case just an `HTTP 202 Accepted` will be returned without any
901 SOAP envelope in the HTTP response message.
- 902 ● When there is a message pending, it is sent on the transport back-channel, using the connection
903 that has been initiated by the `MakeConnection` request.

904 3.11 MessagePending

905 When `MakeConnection` is used, and a message is returned on the transport-specific back-channel, the
906 `MessagePending` header SHOULD be included on the returned message as an indicator whether there
907 are additional messages waiting to be retrieved using the same selection criteria that was specified in the
908 `MakeConnection` element.

909 The following exemplar defines the `MessagePending` syntax:

```
910 <wsrm:MessagePending pending="xs:boolean" ...>  
911   ...
```


912

`</wsrm:MessagePending>`

913 The following describes the content model of the `MessagePending` header block.

914 `/wsrm:MessagePending`

915 This element indicates whether additional messages are waiting to be retrieved.

916 `/wsrm:MessagePending@pending`

917 This attribute, when set to "true", indicates that there is at least one message waiting to be retrieved.

918 When this attribute is set to "false" it indicates there are currently no messages waiting to be retrieved.

919 `/wsrm:MessagePending/{any}`

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

922 `/wsrm:MessagePending/@{any}`

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

925 The absence of the `MessagePending` header has no implication as to whether there are additional
926 messages waiting to be retrieved.

927 4 Faults

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

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

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

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

941 The definitions of faults use the following properties:

942 [Code] The fault code.

943 [Subcode] The fault subcode.

944 [Reason] The English language reason element.

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

948 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
949 "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

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

```
951 <S:Envelope>
952   <S:Header>
953     <wsa:Action>
954       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault
955     </wsa:Action>
956     <!-- Headers elided for brevity. -->
957   </S:Header>
958   <S:Body>
959     <S:Fault>
960       <S:Code>
961         <S:Value> [Code] </S:Value>
962         <S:Subcode>
963           <S:Value> [Subcode] </S:Value>
964         </S:Subcode>
965       </S:Code>
966       <S:Reason>
967         <S:Text xml:lang="en"> [Reason] </S:Text>
968       </S:Reason>
969     <S:Detail>
```

```

970     [Detail]
971     ...
972     </S:Detail>
973     </S:Fault>
974     </S:Body>
975     </S:Envelope>

```

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

```

978 <S11:Envelope>
979   <S11:Header>
980     <wsrm:SequenceFault>
981       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
982       <wsrm:Detail> [Detail] </wsrm:Detail>
983       ...
984     </wsrm:SequenceFault>
985     <!-- Headers elided for brevity. -->
986   </S11:Header>
987   <S11:Body>
988     <S11:Fault>
989       <faultcode> [Code] </faultcode>
990       <faultstring> [Reason] </faultstring>
991     </S11:Fault>
992   </S11:Body>
993 </S11:Envelope>

```

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

```

996 <S11:Envelope>
997   <S11:Body>
998     <S11:Fault>
999       <faultcode> [Subcode] </faultcode>
1000       <faultstring> [Reason] </faultstring>
1001     </S11:Fault>
1002   </S11:Body>
1003 </S11:Envelope>

```

1004 4.1 SequenceFault Element

1005 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during
1006 the reliable messaging specific processing of a message belonging to a Sequence. WS-
1007 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP
1008 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in
1009 conjunction with the SOAP 1.2 binding.

1010 The following exemplar defines its syntax:

```

1011 <wsrm:SequenceFault ...>
1012   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1013   <wsrm:Detail> ... </wsrm:Detail> ?
1014   ...
1015 </wsrm:SequenceFault>

```

1016 The following describes the content model of the `SequenceFault` element.

1017 `/wsrm:SequenceFault`

1018 This is the element containing Sequence information for WS-ReliableMessaging

1019 /wsm:SequenceFault/wsm:FaultCode
 1020 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a
 1021 qualified name from the set of fault [Subcodes] defined below.

1022 /wsm:SequenceFault/wsm:Detail
 1023 This element, if present, carries application specific error information related to the fault being described.

1024 /wsm:SequenceFault/wsm:Detail/{any}
 1025 The application specific error information related to the fault being described.

1026 /wsm:SequenceFault/wsm:Detail/@{any}
 1027 The application specific error information related to the fault being described.

1028 /wsm:SequenceFault/{any}
 1029 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
 1030 to be passed.

1031 /wsm:SequenceFault/@{any}
 1032 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
 1033 element.

1034 4.2 Sequence Terminated

1035 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 1036 Endpoint of this decision.

1037 Properties:

1038 [Code] Sender or Receiver
 1039 [Subcode] wsm:SequenceTerminated
 1040 [Reason] The Sequence has been terminated due to an unrecoverable error.
 1041 [Detail]

1042 `<wsm:Identifier ...> xs:anyURI </wsm:Identifier>`

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

1043 4.3 Unknown Sequence

1044 Properties:

1045 [Code] Sender
 1046 [Subcode] wsm:UnknownSequence

1047 [Reason] The value of wsrn:Identifier is not a known Sequence identifier.

1048 [Detail]

1049 `<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.

1050 **4.4 Invalid Acknowledgement**

1051 An example of when this fault is generated is when a message is Received by the RM Source containing
1052 a SequenceAcknowledgement covering messages that have not been sent.

1053 [Code] Sender

1054 [Subcode] wsrn:InvalidAcknowledgement

1055 [Reason] The SequenceAcknowledgement violates the cumulative Acknowledgement invariant.

1056 [Detail]

1057 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

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

1058 **4.5 Message Number Rollover**

1059 If the condition listed below is reached, the RM Destination MUST generate this fault.

1060 Properties:

1061 [Code] Sender

1062 [Subcode] wsrn:MessageNumberRollover

1063 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1064 [Detail]

```
1065 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1066 <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.

1067 4.6 Create Sequence Refused

1068 Properties:

1069 [Code] Sender or Receiver

1070 [Subcode] wsrm:CreateSequenceRefused

1071 [Reason] The Create Sequence request has been refused by the RM Destination.

1072 [Detail]

```
1073 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.

1074 4.7 Sequence Closed

1075 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1076 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1077 is closed.

1078 Properties:

1079 [Code] Sender

1080 [Subcode] wsrm:SequenceClosed

1081 [Reason] The Sequence is closed and can not accept new messages.

1082 [Detail]

1083 `<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.

1084 **4.8 WSRM Required**

1085 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1086 message that did not use this protocol.

1087 Properties:

1088 [Code] Sender

1089 [Subcode] wsrm:WSRMRequired

1090 [Reason] The RM Destination requires the use of WSRM.

1091 [Detail]

1092 `xs:any`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	On receipt of a message that does not use this protocol and for which this protocol is required.	Unspecified.	Unspecified.

1093 **4.9 Unsupported Selection**

1094 The QName of the unsupported element(s) are included in the detail.

1095 Properties:

1096 [Code] Receiver

1097 [Subcode] wsrm:UnsupportedSelection

1098 [Reason] The extension element used in the message selection is not supported by the RM Source

1099 [Detail]

1100 `<wsrm:UnsupportedElement> xs:QName </wsrm:UnsupportedElement>+`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source or RM Destination.	In response to a <code>MakeConnection</code> message containing a selection criteria in the extensibility section of the message that is not supported	Unspecified.	Unspecified.

1101 **5 Security Threats and Countermeasures**

1102 This specification considers two sets of security requirements, those of the applications that use the WS-
1103 RM protocol and those of the protocol itself.

1104 This specification makes no assumptions about the security requirements of the applications that use WS-
1105 RM. However, once those requirements have been satisfied within a given operational context, the
1106 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1107 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1108 There are many other security concerns that one may need to consider when implementing or using this
1109 protocol. The material below should not be considered as a "check list". Implementers and users of this
1110 protocol are urged to perform a security analysis to determine their particular threat profile and the
1111 appropriate responses to those threats.

1112 Implementers are also advised that there is a core tension between security and reliable messaging that
1113 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1114 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1115 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1116 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1117 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1118 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1119 avoid and prevent this condition.

1120 **5.1 Threats and Countermeasures**

1121 The primary security requirement of this protocol is to protect the specified semantics and protocol
1122 invariants against various threats. The following sections describe several threats to the integrity and
1123 operation of this protocol and provide some general outlines of countermeasures to those threats.
1124 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1125 to all operational contexts.

1126 **5.1.1 Integrity Threats**

1127 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1128 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1129 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1130 to its intended message represents a threat to the WS-RM protocol.

1131 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM
1132 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1133 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be
1134 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1135 **5.1.1.1 Countermeasures**

1136 Integrity threats are generally countered via the use of digital signatures some level of the communication
1137 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1138 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers
1139 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which
1140 they occur, implementations MUST allow for signatures that cover only these headers.

1141 **5.1.2 Resource Consumption Threats**

1142 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1143 implement that RM Destination. These resources can include network connections, database tables,
1144 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1145 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1146 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1147 message Delivery and use this Sequence to send a stream of very large messages to that service,
1148 making sure to omit message number “1” from that stream.

1149 **5.1.2.1 Countermeasures**

1150 There are a number of countermeasures against the described resource consumption threats. The
1151 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1152 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1153 some cases, allows the identity of any attackers to be determined.

1154 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and
1155 authenticate the RM Source that issued the `CreateSequence` message.

1156 **5.1.3 Sequence Spoofing Threats**

1157 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1158 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1159 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1160 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the
1161 current `MessageNumber` for their target Sequence.

1162 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP
1163 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier
1164 to attack the Sequence. These attacks are “two-way” in that an attacker may choose to target the RM
1165 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends
1166 to the `AcksTo` EPR of an RM Source.

1167 **5.1.3.1 Sequence Hijacking**

1168 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject
1169 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1170 messages.

1171 Note that “sequence hijacking” should not be equated with “security session hijacking”. Although a
1172 Sequence may be bound to some form of a security session in order to counter the threats described in
1173 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1174 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1175 established by the security infrastructure to make such determinations. Failure to observe this rule
1176 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1177 the ability to authenticate its peers even though the necessary security processing has taken place.

1178 **5.1.3.2 Countermeasures**

1179 There are a number of countermeasures against sequence spoofing threats. The technique advocated by
1180 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1181 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1182 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1183 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1184 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1185 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1186 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1187 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1188 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1189 sequence peer it MUST be able to identify and authenticate the entity that sent the
1190 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1191 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1192 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1193 creation time.

1194 **5.2 Security Solutions and Technologies**

1195 The security threats described in the previous sections are neither new nor unique. The solutions that
1196 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1197 section maps the facilities provided by common web services security solutions against countermeasures
1198 described in the previous sections.

1199 Before continuing this discussion, however, some examination of the underlying requirements of the
1200 previously described countermeasures is necessary. Specifically it should be noted that the technique
1201 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1202 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check
1203 against this authenticated identity and determines if the RM Source is permitted to create Sequences with
1204 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,
1205 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of
1206 such facilities is considered to be beyond the scope of this specification.

1207 **5.2.1 Transport Layer Security**

1208 This section describes how the facilities provided by SSL/TLS [RFC 4346] can be used to implement the
1209 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1210 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1211 The description provided here is general in nature and is not intended to serve as a complete definition on
1212 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1213 choice of features as well as the manner in which they will be used. The mechanisms described in the
1214 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the
1215 requirements and constraints of the use of SSL/TLS.

1216 **5.2.1.1 Model**

1217 The basic model for using SSL/TLS is as follows:

- 1218 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1219 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1220 Destination.

- 1221 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1222 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1223 synchronous `CreateSequenceResponse` using the session established in (1).
- 1224 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1225 any and all messages or faults that refer to that Sequence.
- 1226 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1227 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1228 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1229 5.2.1.2 Countermeasure Implementation

1230 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1231 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1232 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1233 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1234 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1235 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1236 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1237 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1238 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1239 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1240 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1241 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1242 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1243 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1244 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1245 Acknowledgement) using BasicAuth.
- 1246 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1247 connection authenticates itself to the party accepting the connection using an X.509 certificate
1248 that is exchanged during the SSL/TLS handshake.

1249 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1250 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1251 Source is authorized to create a Sequence with the RM Destination.

1252 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1253 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1254 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1255 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1256 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1257 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1258 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1259 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1260 to protect that Sequence.

1261 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1262 countermeasures (such as associating specific authentication information with a Sequence) although such
1263 methods are not covered by this document.

1264 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1265 session) are outside the scope of this specification.

1266 **5.2.2 SOAP Message Security**

1267 The mechanisms described in WS-Security may be used in various ways to implement the
1268 countermeasures described in the previous sections. This specification advocates using the protocol
1269 described by WS-SecureConversation [[SecureConversation](#)] (optionally in conjunction with WS-Trust
1270 [[Trust](#)]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1271 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1272 The description provided here is general in nature and is not intended to serve as a complete definition on
1273 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1274 need to agree on the choice of features as well as the manner in which they will be used. The
1275 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1276 describe the requirements and constraints of the use of WS-SecureConversation.

1277 **5.2.2.1 Model**

1278 The basic model for using WS-SecureConversation is as follows:

- 1279 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This
1280 may involve the participation of third parties such as a security token service. The tokens
1281 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1282 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1283 context that will be used to protect the Sequence. This is done so that, in cases where the
1284 `CreateSequence` message is signed by more than one security context, the RM Source can
1285 indicate which security context should be used to protect the newly created Sequence.
- 1286 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1287 associated with the security context to sign (as defined by WS-Security) at least the body and any
1288 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

1289 **5.2.2.2 Countermeasure Implementation**

1290 Without relying upon any authentication information, the per-message signatures provide the necessary
1291 integrity qualities to counter the threats described in Section 5.1.1.

1292 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1293 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1294 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1295 create a Sequence with the RM Destination.

1296 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1297 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1298 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1299 context rather than on any authentication claims that may have been established during security context
1300 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1301 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1302 document.

1303 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer
1304 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1305 the association between a Sequence and its protecting security context cannot always be established
1306 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1307 `CreateSequenceResponse` messages may be signed by more than one security context.

1308 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1309 amending or renewing contexts) are outside the scope of this specification.

1310 6 Securing Sequences

1311 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared
1312 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1313 achieving this objective depending upon the underlying security infrastructure.

1314 6.1 Securing Sequences Using WS-Security

1315 One mechanism for protecting a Sequence is to include a security token using a
1316 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1317 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1318 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1319 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1320 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1321 there may be more than one token on a `CreateSequence` message or inferred from the communication
1322 context (e.g. transport protection).

1323 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1324 if the token being referenced supports such mechanism.

1325 The following exemplar defines the `CreateSequence` syntax when extended to include a
1326 `wsse:SecurityTokenReference`:

```
1327 <wsrm:CreateSequence ...>  
1328   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1329   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1330   <wsrm:Offer ...>  
1331     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1332     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1333     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1334     <wsrm:IncompleteSequenceBehavior>  
1335       wsrml:IncompleteSequenceBehaviorType  
1336     </wsrm:IncompleteSequenceBehavior> ?  
1337     ...  
1338   </wsrm:Offer> ?  
1339   ...  
1340   <wsse:SecurityTokenReference>  
1341     ...  
1342   </wsse:SecurityTokenReference> ?  
1343   ...  
1344 </wsrm:CreateSequence>
```

1345 The following describes the content model of the additional `CreateSequence` elements.

1346 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1347 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in
1348 section 3.4) to communicate an explicit reference to the security token, using a
1349 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and Destination
1350 MUST use to authorize messages for the created (and, if present, the offered) Sequence(s). All
1351 subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST
1352 demonstrate proof-of-possession of the secret associated with the token (e.g., by using or deriving from a
1353 private or secret key).

1354 When a RM Source transmits a `CreateSequence` that has been extended to include a
1355 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and
1356 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include

1357 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1358 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1359 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1360 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1361 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1362 in WS-Security still applies.

1363 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1364 <wsm:UsesSequenceSTR ... />
```

1365 The following describes the content model of the `UsesSequenceSTR` header block.

1366 `/wsm:UsesSequenceSTR`

1367 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1368 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1369 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1370 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1371 Sequence creation.

1372 The following is an example of a `CreateSequence` message using the

1373 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1374 <soap:Envelope ...>  
1375   <soap:Header>  
1376     ...  
1377     <wsm:UsesSequenceSTR soap:mustUnderstand='true' />  
1378     ...  
1379   </soap:Header>  
1380   <soap:Body>  
1381     <wsm:CreateSequence>  
1382       <wsm:AcksTo>  
1383         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1384       </wsm:AcksTo>  
1385       <wsse:SecurityTokenReference>  
1386         ...  
1387       </wsse:SecurityTokenReference>  
1388     </wsm:CreateSequence>  
1389   </soap:Body>  
1390 </soap:Envelope>
```

1391 6.2 Securing Sequences Using SSL/TLS

1392 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1393 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1394 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1395 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1396 SOAP header block within the `CreateSequence` message.

1397 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1398 <wsm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1399 The following describes the content model of the `UsesSequenceSSL` header block.

1400 `/wsm:UsesSequenceSSL`

1401 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1402 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was

1403 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a
1404 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand
1405 and correctly implement the functionality described in Section 5.2.1 or else generate a
1406 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1407 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1408 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1409 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1410 `CreateSequenceResponse` message.

1411 **7 References**

1412 **7.1 Normative**

1413 **[KEYWORDS]**

1414 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,
1415 March 1997

1416 <http://www.ietf.org/rfc/rfc2119.txt>

1417 **[WS-RM Policy]**

1418 OASIS WS-RX Technical Committee Draft, "[Web Services ReliableMessaging Policy Assertion\(WS-RM
1419 Policy\)](#)" October 2006

1420 <http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-wd-11.pdf>

1421 **[SOAP 1.1]**

1422 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

1423 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1424 **[SOAP 1.2]**

1425 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

1426 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

1427 **[URI]**

1428 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,
1429 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1430 <http://ietf.org/rfc/rfc3986>

1431 **[UUID]**

1432 P. Leach, M. Mealling, R. Salz, "[A Universally Unique IDentifier \(UUID\) URN Namespace](#)," RFC 4122,
1433 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1434 <http://www.ietf.org/rfc/rfc4122.txt>

1435 **[XML]**

1436 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#)", September 2006.

1437 <http://www.w3.org/TR/REC-xml/>

1438 **[XML-ns]**

1439 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

1440 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

1441 **[XML-Schema Part1]**

1442 W3C Recommendation, "[XML Schema Part 1: Structures](#)," October 2004.

1443 <http://www.w3.org/TR/xmlschema-1/>

1444 **[XML-Schema Part2]**

1445 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

1446 <http://www.w3.org/TR/xmlschema-2/>

1447 **[XPath 1.0]**

1448 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

1449 <http://www.w3.org/TR/xpath>

1450 **[WSDL 1.1]**

1451 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

1452 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1453 **[WS-Addressing]**

1454 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

1455 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

1456 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

1457 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1458 **7.2 Non-Normative**

1459 **[BSP 1.0]**

1460 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

1461 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

1462 **[RDDL 2.0]**

1463 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

1464 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

1465 **[RFC 2617]**

1466 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
1467 Authentication: Basic and Digest Access Authentication," June 1999.

1468 <http://www.ietf.org/rfc/rfc2617.txt>

1469 **[RFC 4346]**

1470 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

1471 <http://www.ietf.org/rfc/rfc4346.txt>

1472 **[WS-Policy]**

1473 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

1474 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

1475 **[WS-PolicyAttachment]**

1476 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

1477 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-
1478 20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1479 **[WS-Security]**

1480 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1481 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1482 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

1483 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1484 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

1485 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

1486 **[RTTM]**

1487 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May
1488 1992.

1489 <http://www.rfc-editor.org/rfc/rfc1323.txt>

1490 **[SecurityPolicy]**

1491 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

1492 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

1493 **[SecureConversation]**

1494 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February
1495 2005.

1496 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

1497 **[Trust]**

1498 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

1499 <http://schemas.xmlsoap.org/ws/2005/02/trust>

1500 Appendix A. Schema

1501 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1502 Schema Part2] is located at:

1503 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

1504 The following copy is provided for reference.

```
1505 <?xml version="1.0" encoding="UTF-8"?>
1506 <!--
1507 OASIS takes no position regarding the validity or scope of any intellectual
1508 property or other rights that might be claimed to pertain to the
1509 implementation or use of the technology described in this document or the
1510 extent to which any license under such rights might or might not be available;
1511 neither does it represent that it has made any effort to identify any such
1512 rights. Information on OASIS's procedures with respect to rights in OASIS
1513 specifications can be found at the OASIS website. Copies of claims of rights
1514 made available for publication and any assurances of licenses to be made
1515 available, or the result of an attempt made to obtain a general license or
1516 permission for the use of such proprietary rights by implementors or users of
1517 this specification, can be obtained from the OASIS Executive Director.
1518 OASIS invites any interested party to bring to its attention any copyrights,
1519 patents or patent applications, or other proprietary rights which may cover
1520 technology that may be required to implement this specification. Please
1521 address the information to the OASIS Executive Director.
1522 Copyright © OASIS Open 2002-2006. All Rights Reserved.
1523 This document and translations of it may be copied and furnished to others,
1524 and derivative works that comment on or otherwise explain it or assist in its
1525 implementation may be prepared, copied, published and distributed, in whole or
1526 in part, without restriction of any kind, provided that the above copyright
1527 notice and this paragraph are included on all such copies and derivative
1528 works. However, this document itself does not be modified in any way, such as
1529 by removing the copyright notice or references to OASIS, except as needed for
1530 the purpose of developing OASIS specifications, in which case the procedures
1531 for copyrights defined in the OASIS Intellectual Property Rights document must
1532 be followed, or as required to translate it into languages other than English.
1533 The limited permissions granted above are perpetual and will not be revoked by
1534 OASIS or its successors or assigns.
1535 This document and the information contained herein is provided on an "AS IS"
1536 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1537 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1538 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1539 FOR A PARTICULAR PURPOSE.
1540 -->
1541 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1542 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1543 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1544 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1545 elementFormDefault="qualified" attributeFormDefault="unqualified">
1546   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1547   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1548   <!-- Protocol Elements -->
1549   <xs:complexType name="SequenceType">
1550     <xs:sequence>
1551       <xs:element ref="wsrm:Identifier"/>
1552       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1553       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1554 maxOccurs="unbounded"/>
1555     </xs:sequence>
```

```

1556     <xs:anyAttribute namespace="##other" processContents="lax"/>
1557 </xs:complexType>
1558 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1559 <xs:element name="SequenceAcknowledgement">
1560   <xs:complexType>
1561     <xs:sequence>
1562       <xs:element ref="wsrm:Identifier"/>
1563       <xs:choice>
1564         <xs:sequence>
1565           <xs:choice>
1566             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1567               <xs:complexType>
1568                 <xs:sequence/>
1569                 <xs:attribute name="Upper" type="xs:unsignedLong"
1570 use="required"/>
1571                 <xs:attribute name="Lower" type="xs:unsignedLong"
1572 use="required"/>
1573               <xs:anyAttribute namespace="##other" processContents="lax"/>
1574             </xs:complexType>
1575           </xs:element>
1576           <xs:element name="None">
1577             <xs:complexType>
1578               <xs:sequence/>
1579             </xs:complexType>
1580           </xs:element>
1581         </xs:choice>
1582         <xs:element name="Final" minOccurs="0">
1583           <xs:complexType>
1584             <xs:sequence/>
1585           </xs:complexType>
1586         </xs:element>
1587       </xs:sequence>
1588       <xs:element name="Nack" type="xs:unsignedLong"
1589 maxOccurs="unbounded"/>
1590     </xs:choice>
1591     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1592 maxOccurs="unbounded"/>
1593   </xs:sequence>
1594   <xs:anyAttribute namespace="##other" processContents="lax"/>
1595 </xs:complexType>
1596 </xs:element>
1597 <xs:complexType name="AckRequestedType">
1598   <xs:sequence>
1599     <xs:element ref="wsrm:Identifier"/>
1600     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1601 maxOccurs="unbounded"/>
1602   </xs:sequence>
1603   <xs:anyAttribute namespace="##other" processContents="lax"/>
1604 </xs:complexType>
1605 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1606 <xs:complexType name="MessagePendingType">
1607   <xs:sequence>
1608     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1609 maxOccurs="unbounded"/>
1610   </xs:sequence>
1611   <xs:attribute name="pending" type="xs:boolean"/>
1612   <xs:anyAttribute namespace="##other" processContents="lax"/>
1613 </xs:complexType>
1614 <xs:element name="MessagePending" type="wsrm:MessagePendingType"/>
1615 <xs:element name="Identifier">
1616   <xs:complexType>
1617     <xs:annotation>
1618       <xs:documentation>

```

```

1619         This type is for elements whose [children] is an anyURI and can have
1620 arbitrary attributes.
1621         </xs:documentation>
1622         </xs:annotation>
1623         <xs:simpleContent>
1624             <xs:extension base="xs:anyURI">
1625                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1626             </xs:extension>
1627         </xs:simpleContent>
1628     </xs:complexType>
1629 </xs:element>
1630 <xs:element name="Address">
1631     <xs:complexType>
1632         <xs:simpleContent>
1633             <xs:extension base="xs:anyURI">
1634                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1635             </xs:extension>
1636         </xs:simpleContent>
1637     </xs:complexType>
1638 </xs:element>
1639 <xs:complexType name="MakeConnectionType">
1640     <xs:sequence>
1641         <xs:element ref="wsrm:Identifier" minOccurs="0" maxOccurs="1"/>
1642         <xs:element ref="wsrm:Address" minOccurs="0" maxOccurs="1"/>
1643         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1644 maxOccurs="unbounded"/>
1645     </xs:sequence>
1646     <xs:anyAttribute namespace="##other" processContents="lax"/>
1647 </xs:complexType>
1648 <xs:element name="MakeConnection" type="wsrm:MakeConnectionType"/>
1649 <xs:simpleType name="MessageNumberType">
1650     <xs:restriction base="xs:unsignedLong">
1651         <xs:minInclusive value="1"/>
1652         <xs:maxInclusive value="9223372036854775807"/>
1653     </xs:restriction>
1654 </xs:simpleType>
1655 <!-- Fault Container and Codes -->
1656 <xs:simpleType name="FaultCodes">
1657     <xs:restriction base="xs:QName">
1658         <xs:enumeration value="wsrm:SequenceTerminated"/>
1659         <xs:enumeration value="wsrm:UnknownSequence"/>
1660         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1661         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1662         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1663         <xs:enumeration value="wsrm:SequenceClosed"/>
1664         <xs:enumeration value="wsrm:WSRMRequired"/>
1665         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1666     </xs:restriction>
1667 </xs:simpleType>
1668 <xs:complexType name="SequenceFaultType">
1669     <xs:sequence>
1670         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1671         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1672         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1673 maxOccurs="unbounded"/>
1674     </xs:sequence>
1675     <xs:anyAttribute namespace="##other" processContents="lax"/>
1676 </xs:complexType>
1677 <xs:complexType name="DetailType">
1678     <xs:sequence>
1679         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1680 maxOccurs="unbounded"/>
1681     </xs:sequence>

```

```

1682     <xs:anyAttribute namespace="##other" processContents="lax"/>
1683 </xs:complexType>
1684 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1685 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1686 <xs:element name="CreateSequenceResponse"
1687 type="wsrm:CreateSequenceResponseType"/>
1688 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1689 <xs:element name="CloseSequenceResponse"
1690 type="wsrm:CloseSequenceResponseType"/>
1691 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1692 <xs:element name="TerminateSequenceResponse"
1693 type="wsrm:TerminateSequenceResponseType"/>
1694 <xs:complexType name="CreateSequenceType">
1695 <xs:sequence>
1696 <xs:element ref="wsrm:AcksTo"/>
1697 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1698 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1699 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1700 maxOccurs="unbounded">
1701 <xs:annotation>
1702 <xs:documentation>
1703     It is the authors intent that this extensibility be used to
1704 transfer a Security Token Reference as defined in WS-Security.
1705 </xs:documentation>
1706 </xs:annotation>
1707 </xs:any>
1708 </xs:sequence>
1709 <xs:anyAttribute namespace="##other" processContents="lax"/>
1710 </xs:complexType>
1711 <xs:complexType name="CreateSequenceResponseType">
1712 <xs:sequence>
1713 <xs:element ref="wsrm:Identifier"/>
1714 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1715 <xs:element name="IncompleteSequenceBehavior"
1716 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1717 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1718 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1719 maxOccurs="unbounded"/>
1720 </xs:sequence>
1721 <xs:anyAttribute namespace="##other" processContents="lax"/>
1722 </xs:complexType>
1723 <xs:complexType name="CloseSequenceType">
1724 <xs:sequence>
1725 <xs:element ref="wsrm:Identifier"/>
1726 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1727 maxOccurs="unbounded"/>
1728 </xs:sequence>
1729 <xs:anyAttribute namespace="##other" processContents="lax"/>
1730 </xs:complexType>
1731 <xs:complexType name="CloseSequenceResponseType">
1732 <xs:sequence>
1733 <xs:element ref="wsrm:Identifier"/>
1734 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1735 maxOccurs="unbounded"/>
1736 </xs:sequence>
1737 <xs:anyAttribute namespace="##other" processContents="lax"/>
1738 </xs:complexType>
1739 <xs:complexType name="TerminateSequenceType">
1740 <xs:sequence>
1741 <xs:element ref="wsrm:Identifier"/>
1742 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1743 maxOccurs="unbounded"/>
1744 </xs:sequence>

```



```

1745     <xs:anyAttribute namespace="##other" processContents="lax"/>
1746 </xs:complexType>
1747 <xs:complexType name="TerminateSequenceResponseType">
1748   <xs:sequence>
1749     <xs:element ref="wsrm:Identifier"/>
1750     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1751 maxOccurs="unbounded"/>
1752   </xs:sequence>
1753   <xs:anyAttribute namespace="##other" processContents="lax"/>
1754 </xs:complexType>
1755 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1756 <xs:complexType name="OfferType">
1757   <xs:sequence>
1758     <xs:element ref="wsrm:Identifier"/>
1759     <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1760     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1761     <xs:element name="IncompleteSequenceBehavior"
1762 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1763     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1764 maxOccurs="unbounded"/>
1765   </xs:sequence>
1766   <xs:anyAttribute namespace="##other" processContents="lax"/>
1767 </xs:complexType>
1768 <xs:complexType name="AcceptType">
1769   <xs:sequence>
1770     <xs:element ref="wsrm:AcksTo"/>
1771     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1772 maxOccurs="unbounded"/>
1773   </xs:sequence>
1774   <xs:anyAttribute namespace="##other" processContents="lax"/>
1775 </xs:complexType>
1776 <xs:element name="Expires">
1777   <xs:complexType>
1778     <xs:simpleContent>
1779       <xs:extension base="xs:duration">
1780         <xs:anyAttribute namespace="##other" processContents="lax"/>
1781       </xs:extension>
1782     </xs:simpleContent>
1783   </xs:complexType>
1784 </xs:element>
1785 <xs:simpleType name="IncompleteSequenceBehaviorType">
1786   <xs:restriction base="xs:string">
1787     <xs:enumeration value="DiscardEntireSequence"/>
1788     <xs:enumeration value="DiscardFollowingFirstGap"/>
1789     <xs:enumeration value="NoDiscard"/>
1790   </xs:restriction>
1791 </xs:simpleType>
1792 <xs:element name="UsesSequenceSTR">
1793   <xs:complexType>
1794     <xs:sequence/>
1795     <xs:anyAttribute namespace="##other" processContents="lax"/>
1796   </xs:complexType>
1797 </xs:element>
1798 <xs:element name="UsesSequenceSSL">
1799   <xs:complexType>
1800     <xs:sequence/>
1801     <xs:anyAttribute namespace="##other" processContents="lax"/>
1802   </xs:complexType>
1803 </xs:element>
1804 <xs:element name="UnsupportedElement">
1805   <xs:simpleType>
1806     <xs:restriction base="xs:QName"/>
1807   </xs:simpleType>

```

1808
1809

```
</xs:element>  
</xs:schema>
```

1810 Appendix B. WSDL

1811 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1812 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1813 present in exchanges with that endpoint.

1814 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1815 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1816 for a higher-level mechanism to indicate that WS-RM is engaged.

1817 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1818 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

1819 The following non-normative copy is provided for reference.

```
1820 <?xml version="1.0" encoding="utf-8"?>
1821 <!--
1822 OASIS takes no position regarding the validity or scope of any intellectual
1823 property or other rights that might be claimed to pertain to the
1824 implementation or use of the technology described in this document or the
1825 extent to which any license under such rights might or might not be available;
1826 neither does it represent that it has made any effort to identify any such
1827 rights. Information on OASIS's procedures with respect to rights in OASIS
1828 specifications can be found at the OASIS website. Copies of claims of rights
1829 made available for publication and any assurances of licenses to be made
1830 available, or the result of an attempt made to obtain a general license or
1831 permission for the use of such proprietary rights by implementors or users of
1832 this specification, can be obtained from the OASIS Executive Director.
1833 OASIS invites any interested party to bring to its attention any copyrights,
1834 patents or patent applications, or other proprietary rights which may cover
1835 technology that may be required to implement this specification. Please
1836 address the information to the OASIS Executive Director.
1837 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
1838 This document and translations of it may be copied and furnished to others,
1839 and derivative works that comment on or otherwise explain it or assist in its
1840 implementation may be prepared, copied, published and distributed, in whole or
1841 in part, without restriction of any kind, provided that the above copyright
1842 notice and this paragraph are included on all such copies and derivative
1843 works. However, this document itself does not be modified in any way, such as
1844 by removing the copyright notice or references to OASIS, except as needed for
1845 the purpose of developing OASIS specifications, in which case the procedures
1846 for copyrights defined in the OASIS Intellectual Property Rights document must
1847 be followed, or as required to translate it into languages other than English.
1848 The limited permissions granted above are perpetual and will not be revoked by
1849 OASIS or its successors or assigns.
1850 This document and the information contained herein is provided on an "AS IS"
1851 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1852 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1853 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1854 FOR A PARTICULAR PURPOSE.
1855 -->
1856 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1857 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1858 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
1859 open.org/ws-rx/wsrn/200608" xmlns:tns="http://docs.oasis-open.org/ws-
1860 rx/wsrn/200608/wsd" targetNamespace="http://docs.oasis-open.org/ws-
1861 rx/wsrn/200608/wsd">
1862 <wsdl:types>
```

```

1863     <xs:schema>
1864         <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1865 schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-
1866 200608.xsd"/>
1867     </xs:schema>
1868 </wsdl:types>

1869     <wsdl:message name="CreateSequence">
1870         <wsdl:part name="create" element="rm:CreateSequence"/>
1871     </wsdl:message>
1872     <wsdl:message name="CreateSequenceResponse">
1873         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1874     </wsdl:message>
1875     <wsdl:message name="CloseSequence">
1876         <wsdl:part name="close" element="rm:CloseSequence"/>
1877     </wsdl:message>
1878     <wsdl:message name="CloseSequenceResponse">
1879         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1880     </wsdl:message>
1881     <wsdl:message name="TerminateSequence">
1882         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1883     </wsdl:message>
1884     <wsdl:message name="TerminateSequenceResponse">
1885         <wsdl:part name="terminateResponse"
1886 element="rm:TerminateSequenceResponse"/>
1887     </wsdl:message>
1888     <wsdl:message name="MakeConnection">
1889         <wsdl:part name="makeConnection" element="rm:MakeConnection"/>
1890     </wsdl:message>

1891     <wsdl:portType name="SequenceAbstractPortType">
1892         <wsdl:operation name="CreateSequence">
1893             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1894 open.org/ws-rx/wsrn/200608/CreateSequence"/>
1895             <wsdl:output message="tns:CreateSequenceResponse"
1896 wsaw:Action="http://docs.oasis-open.org/ws-
1897 rx/wsrn/200608/CreateSequenceResponse"/>
1898         </wsdl:operation>
1899         <wsdl:operation name="CloseSequence">
1900             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1901 open.org/ws-rx/wsrn/200608/CloseSequence"/>
1902             <wsdl:output message="tns:CloseSequenceResponse"
1903 wsaw:Action="http://docs.oasis-open.org/ws-
1904 rx/wsrn/200608/CloseSequenceResponse"/>
1905         </wsdl:operation>
1906         <wsdl:operation name="TerminateSequence">
1907             <wsdl:input message="tns:TerminateSequence"
1908 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence"/>
1909             <wsdl:output message="tns:TerminateSequenceResponse"
1910 wsaw:Action="http://docs.oasis-open.org/ws-
1911 rx/wsrn/200608/TerminateSequenceResponse"/>
1912         </wsdl:operation>
1913         <wsdl:operation name="MakeConnection">
1914             <wsdl:input message="tns:MakeConnection" wsaw:Action="http://docs.oasis-
1915 open.org/ws-rx/wsrn/200608/MakeConnection"/>
1916             <!-- As described in section 3.10, the MakeConnection operation
1917 establishes a connection. If a matching message is available then
1918 the back-channel of the connection will be used to carry the
1919 message. In SOAP terms the returned message is not a response,
1920 so there is no WSDL output message. -->
1921         </wsdl:operation>
1922     </wsdl:portType>

```

```
</wsdl:definitions>
```

1924 Appendix C. Message Examples

1925 Appendix C.1 Create Sequence

1926 Create Sequence

```
1927 <?xml version="1.0" encoding="UTF-8"?>
1928 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1929 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1930 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1931   <S:Header>
1932     <wsa:MessageID>
1933       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1934     </wsa:MessageID>
1935     <wsa:To>http://example.com/serviceB/123</wsa:To>
1936     <wsa:Action>http://docs.oasis-open.org/ws-
1937 rx/wsmr/200608/CreateSequence</wsa:Action>
1938     <wsa:ReplyTo>
1939       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1940     </wsa:ReplyTo>
1941   </S:Header>
1942   <S:Body>
1943     <wsmr:CreateSequence>
1944       <wsmr:AcksTo>
1945         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1946       </wsmr:AcksTo>
1947     </wsmr:CreateSequence>
1948   </S:Body>
1949 </S:Envelope>
```

1950 Create Sequence Response

```
1951 <?xml version="1.0" encoding="UTF-8"?>
1952 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1953 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1954 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1955   <S:Header>
1956     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1957     <wsa:RelatesTo>
1958       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1959     </wsa:RelatesTo>
1960     <wsa:Action>
1961       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
1962     </wsa:Action>
1963   </S:Header>
1964   <S:Body>
1965     <wsmr:CreateSequenceResponse>
1966       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1967     </wsmr:CreateSequenceResponse>
1968   </S:Body>
1969 </S:Envelope>
```

1970 Appendix C.2 Initial Transmission

1971 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1972 figure. The three messages have the following headers; the third message is identified as the last
1973 message in the Sequence:

1974 **Message 1**

```
1975 <?xml version="1.0" encoding="UTF-8"?>
1976 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1977 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1978 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1979   <S:Header>
1980     <wsa:MessageID>
1981       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1982     </wsa:MessageID>
1983     <wsa:To>http://example.com/serviceB/123</wsa:To>
1984     <wsa:From>
1985       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1986     </wsa:From>
1987     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1988     <wsmr:Sequence>
1989       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1990       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1991     </wsmr:Sequence>
1992   </S:Header>
1993   <S:Body>
1994     <!-- Some Application Data -->
1995   </S:Body>
1996 </S:Envelope>
```

1997 **Message 2**

```
1998 <?xml version="1.0" encoding="UTF-8"?>
1999 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2000 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2001 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2002   <S:Header>
2003     <wsa:MessageID>
2004       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2005     </wsa:MessageID>
2006     <wsa:To>http://example.com/serviceB/123</wsa:To>
2007     <wsa:From>
2008       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2009     </wsa:From>
2010     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2011     <wsmr:Sequence>
2012       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
2013       <wsmr:MessageNumber>2</wsmr:MessageNumber>
2014     </wsmr:Sequence>
2015   </S:Header>
2016   <S:Body>
2017     <!-- Some Application Data -->
2018   </S:Body>
2019 </S:Envelope>
```

2020 **Message 3**

```
2021 <?xml version="1.0" encoding="UTF-8"?>
2022 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2023 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2024 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2025   <S:Header>
2026     <wsa:MessageID>
2027       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
2028     </wsa:MessageID>
2029     <wsa:To>http://example.com/serviceB/123</wsa:To>
2030     <wsa:From>
2031       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
```

```

2032 </wsa:From>
2033 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2034 <wsrm:Sequence>
2035 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2036 <wsrm:MessageNumber>3</wsrm:MessageNumber>
2037 </wsrm:Sequence>
2038 <wsrm:AckRequested>
2039 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2040 </wsrm:AckRequested>
2041 </S:Header>
2042 <S:Body>
2043 <!-- Some Application Data -->
2044 </S:Body>
2045 </S:Envelope>

```

2046 **Appendix C.3 First Acknowledgement**

2047 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
2048 responds with an Acknowledgement for messages 1 and 3:

```

2049 <?xml version="1.0" encoding="UTF-8"?>
2050 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2051 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2052 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2053 <S:Header>
2054 <wsa:MessageID>
2055 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
2056 </wsa:MessageID>
2057 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2058 <wsa:From>
2059 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2060 </wsa:From>
2061 <wsa:Action>
2062 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
2063 </wsa:Action>
2064 <wsrm:SequenceAcknowledgement>
2065 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2066 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2067 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2068 </wsrm:SequenceAcknowledgement>
2069 </S:Header>
2070 <S:Body/>
2071 </S:Envelope>

```

2072 **Appendix C.4 Retransmission**

2073 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2074 requests an Acknowledgement:

```

2075 <?xml version="1.0" encoding="UTF-8"?>
2076 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2077 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2078 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2079 <S:Header>
2080 <wsa:MessageID>
2081 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2082 </wsa:MessageID>
2083 <wsa:To>http://example.com/serviceB/123</wsa:To>
2084 <wsa:From>
2085 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2086 </wsa:From>

```



```

2087 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2088 <wsrm:Sequence>
2089 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2090 <wsrm:MessageNumber>2</wsrm:MessageNumber>
2091 </wsrm:Sequence>
2092 <wsrm:AckRequested>
2093 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2094 </wsrm:AckRequested>
2095 </S:Header>
2096 <S:Body>
2097 <!-- Some Application Data -->
2098 </S:Body>
2099 </S:Envelope>

```

2100 Appendix C.5 Termination

2101 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2102 be terminated:

```

2103 <?xml version="1.0" encoding="UTF-8"?>
2104 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2105 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2106 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2107 <S:Header>
2108 <wsa:MessageID>
2109 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2110 </wsa:MessageID>
2111 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2112 <wsa:From>
2113 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2114 </wsa:From>
2115 <wsa:Action>
2116 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
2117 </wsa:Action>
2118 <wsrm:SequenceAcknowledgement>
2119 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2120 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2121 </wsrm:SequenceAcknowledgement>
2122 </S:Header>
2123 <S:Body/>
2124 </S:Envelope>

```

2125 Terminate Sequence

```

2126 <?xml version="1.0" encoding="UTF-8"?>
2127 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2128 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2129 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2130 <S:Header>
2131 <wsa:MessageID>
2132 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2133 </wsa:MessageID>
2134 <wsa:To>http://example.com/serviceB/123</wsa:To>
2135 <wsa:Action>
2136 http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence
2137 </wsa:Action>
2138 <wsa:From>
2139 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2140 </wsa:From>
2141 </S:Header>
2142 <S:Body>
2143 <wsrm:TerminateSequence>

```

```

2144     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2145     </wsrm:TerminateSequence>
2146   </S:Body>
2147 </S:Envelope>

```

2148 Terminate Sequence Response

```

2149 <?xml version="1.0" encoding="UTF-8"?>
2150 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2151   xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2152   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2153   <S:Header>
2154     <wsa:MessageID>
2155       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2156     </wsa:MessageID>
2157     <wsa:To>http://example.com/serviceA/789</wsa:To>
2158     <wsa:Action>
2159       http://docs.oasis-open.org/ws-rx/wsmr/200608/TerminateSequenceResponse
2160     </wsa:Action>
2161     <wsa:RelatesTo>
2162       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2163     </wsa:RelatesTo>
2164     <wsa:From>
2165       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2166     </wsa:From>
2167   </S:Header>
2168   <S:Body>
2169     <wsrm:TerminateSequenceResponse>
2170       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2171     </wsrm:TerminateSequenceResponse>
2172   </S:Body>
2173 </S:Envelope>

```

2174 Appendix C.6 MakeConnection

2175 To illustrate how a `MakeConnection` message exchange can be used to deliver messages to an
 2176 Endpoint that is not addressable, consider the case of a pub/sub scenario in which the Endpoint to which
 2177 notifications are to be delivered (the "event consumer") is not addressable by the notification sending
 2178 Endpoint (the "event producer"). In this scenario the event consumer must initiate the connections in order
 2179 for the notifications to be delivered. One possible set of message exchanges (using HTTP) that
 2180 demonstrate how this can be achieved using `MakeConnection` is shown below.

2181 **Step 1** – During a "subscribe" operation, the event consumer's EPR specifies the RM anonymous URI
 2182 and the WS-RM Policy Assertion to indicate whether or not RM is required:

```

2183 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2184   xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2185   xmlns:wsmrp="http://docs.oasis-open.org/ws-rx/wsmrp/200608"
2186   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2187   <S:Header>
2188     <wsa:To> http://example.org/subscriptionService </wsa:To>
2189     <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813</wsa:MessageID>
2190     <wsa:ReplyTo>
2191       <wsa:To> http://client456.org/response </wsa:To>
2192     </wsa:ReplyTo>
2193   </S:Header>
2194   <S:Body>
2195     <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">
2196       <!-- subscription service specific data -->
2197     <targetEPR>

```

```

2198     <wsa:Address>http://docs.oasis-open.org/ws-
2199 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>
2200     <wsa:Metadata>
2201       <wsp:Policy wsu:Id="MyPolicy">
2202         <wsrmp:RMAssertion/>
2203       </wsp:Policy>
2204     </wsa:Metadata>
2205   </targetEPR>
2206 </sub:Subscribe>
2207 </S:Body>
2208 </S:Envelope>

```

2209 In this example the `subscribe` and `targetEPR` elements are simply examples of what a subscription
2210 request message might contain. Note: the `wsa:Address` element contains the RM anonymous URI
2211 indicating that the notification producer needs to queue the messages until they are requested using the
2212 `MakeConnection` message exchange. The EPR also contains the WS-RM Policy Assertion indicating
2213 the RM must be used when notifications related to this subscription are sent.

2214 **Step 2** – Once the subscription is established, the event consumer checks for a pending message:

```

2215 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2216 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2217 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2218   <S:Header>
2219     <wsa:Action>http://docs.oasis-open.org/ws-
2220 rx/wsrn/200608/MakeConnection</wsa:Action>
2221     <wsa:To> http://example.org/subscriptionService </wsa:To>
2222   </S:Header>
2223   <S:Body>
2224     <wsrm:MakeConnection>
2225       <wsrm:Address>http://docs.oasis-open.org/ws-
2226 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-
2227 446655440000</wsrm:Address>
2228     </wsrm:MakeConnection>
2229   </S:Body>
2230 </S:Envelope>

```

2231 **Step 3** – If there are messages waiting to be delivered then a message will be returned back to the event
2232 consumer. However, because WS-RM is being used to deliver the messages, the first message returned
2233 is a `CreateSequence`:

```

2234 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2235 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2236 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2237   <S:Header>
2238     <wsa:Action>http://docs.oasis-open.org/ws-
2239 rx/wsrn/200608/CreateSequence</wsa:Action>
2240     <wsa:To>http://docs.oasis-open.org/ws-
2241 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
2242     <wsa:ReplyTo> http://example.org/subscriptionService </wsa:ReplyTo>
2243     <wsa:MessageID> http://example.org/id-123-456 </wsa:MessageID>
2244   </S:Header>
2245   <S:Body>
2246     <wsrm:CreateSequence>
2247       <wsrm:AcksTo>
2248         <wsa:Address> http://example.org/subscriptionService </wsa:Address>
2249       </wsrm:AcksTo>
2250     </wsrm:CreateSequence>
2251   </S:Body>

```

2252

```
</S:Envelope>
```

2253 Notice from the perspective of how the RM Source on the event producer interacts with the RM
2254 Destination of those messages, nothing new is introduced by the use of the `MakeConnection`, the use
2255 of RM protocol is the same as the case where the event consumer is addressable.

2256 **Step 4** – The event consumer will respond with a `CreateSequenceResponse` message per normal WS-
2257 Addressing rules:

2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:Action>http://docs.oasis-open.org/ws-
rx/wsmr/200608/CreateSequenceResponse</wsa:Action>
    <wsa:To> http://example.org/subscriptionService </wsa:To>
    <wsa:RelatesTo> http://example.org/id-123-456 </wsa:RelatesTo>
  </S:Header>
  <S:Body>
    <wsmr:CreateSequenceResponse>
      <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>
    </wsmr:CreateSequenceResponse>
  </S:Body>
</S:Envelope>
```

2273 Note, this message is carried on an HTTP request directed to the `wsa:ReplyTo` EPR, and the HTTP
2274 response will be an HTTP 202.

2275 **Step 5** – The event consumer checks for another message pending:

2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:Action>http://docs.oasis-open.org/ws-
rx/wsmr/200608/MakeConnection</wsa:Action>
    <wsa:To> http://example.org/subscriptionService </wsa:To>
  </S:Header>
  <S:Body>
    <wsmr:MakeConnection>
      <wsmr:Address>http://docs.oasis-open.org/ws-
rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-
446655440000</wsmr:Address>
    </wsmr:MakeConnection>
  </S:Body>
</S:Envelope>
```

2292 Notice this is the same message as the one sent in step 2.

2293 **Step 6** – If there is a message pending for this destination then it is returned on the HTTP response:

2294
2295
2296
2297
2298
2299
2300

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:Action> http://example.org/eventType1 </wsa:Action>
    <wsa:To>http://docs.oasis-open.org/ws-
rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
```

```

2301     <wsrm:Sequence>
2302         <wsrm:Identifier> http://example.org/rmid-456 </wsrm:Identifier>
2303     </wsrm:Sequence>
2304     <wsrm:MessagePending pending="true"/>
2305 </S:Header>
2306 <S:Body>
2307     <!-- event specific data -->
2308 </S:Body>
2309 </S:Envelope>

```

2310 As noted in step 3, the use of the RM protocol does not change when using `MakeConnection`. The
 2311 format of the messages, the order of the messages sent and the timing of when to send it remains the
 2312 same.

2313 **Step 7** – At some later interval, or immediately due to the `MessagePending` header's "pending"
 2314 attribute being set to "true", the event consumer will poll again:

```

2315 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2316 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
2317 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2318     <S:Header>
2319         <wsa:Action>http://docs.oasis-open.org/ws-
2320 rx/wsrm/200608/MakeConnection</wsa:Action>
2321         <wsa:To> http://example.org/subscriptionService </wsa:To>
2322     </S:Header>
2323     <S:Body>
2324         <wsrm:MakeConnection>
2325             <wsrm:Address>http://docs.oasis-open.org/ws-
2326 rx/wsrm/200608/anonymous?id=550e8400-e29b-11d4-a716-
2327 446655440000</wsrm:Address>
2328         </wsrm:MakeConnection>
2329     </S:Body>
2330 </S:Envelope>

```

2331 Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to
 2332 the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event
 2333 producer to send not only application messages but RM protocol messages (e.g. `CloseSequence`,
 2334 `TerminateSequence` or even additional `CreateSequences`) as needed.

2335 **Step 8** – If at any point in time there are no messages pending, in response to a `MakeConnection` the
 2336 event producer returns an HTTP 202 back to the event consumer. The process then repeats (back to step
 2337 7) until the subscription ends.

2338 Appendix D. State Tables

2339 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2340 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2341 Legend:

2342 The first column of these tables contains the motivating event and has the following format:

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

2343 Where:

- 2344 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2345 described by the specification.
- 2346 ● [source]: indicates the source of the event; one of:
 - 2347 ● [msg] a Received message
 - 2348 ● [int]: an internal event such as the firing of a timer
 - 2349 ● [app]: the application
 - 2350 ● [unspec]: the source is unspecified

2351 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}

2352 Where:

- 2353 ● action to take: indicates that the state machine performs the following action. Actions surrounded
2354 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2355 "Transmit"
 - 2356 ● [next state]: indicates the state to which the state machine will advance upon the performance of
2357 the action. For ease of reading the next state "same" indicates that the state does not change.
 - 2358 ● {ref} is a reference to the document section describing the behavior in this cell
- 2359 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2360 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2361 described in this specification and does not indicate normal protocol operation. Implementations MAY
2362 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2363 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2364 combinations.

2365 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] {3.4}	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] {3.4}	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]
<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}
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] {3.4}	Generate Unknown Sequence Fault	Generate Unknown Sequence Fault	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
{4.7}	[Same] {4.3}	[Same] {4.3}				
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}
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 Acknowledgment [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}

2366 Table 2 RM Destination Sequence State Transition Table

Events	Sequence States		
	None	Created	Closed
CreateSequence (successful) [msg/int] {3.4}	Xmit Create Sequence Response [Created] {3.4}	N/A	N/A
CreateSequence (unsuccessful) [msg/int] {3.4}	Generate Create Sequence Refused Fault [None] {3.4}	N/A	N/A
Message (with message number within range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Accept Message; <Xmit SeqAck> [Same]	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}
Message (with message number outside of range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.7}{4.5}	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}
<AckRequested> [msg] {3.8}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.8}	Xmit SeqAck+Final [Same] {3.9}

Events	Sequence States		
	None	Created	Closed
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}
<CloseSequence autonomously> [int]	N/A	No Action [Closed]	N/A
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}
UnknownSequence Fault [msg] {4.3}		Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
SequenceTerminated Fault [msg] {4.2}		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
Invalid Acknowledgement Fault [msg] {4.4}	N/A		
Expires exceeded [int]	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}

2367 The following two tables apply only if the `MakeConnection` mechanism is utilized.

2368 Table 3 Sending Endpoint Message Transfer Engine

Event	None	Queued n=1	Queued, n>1
Message destined to anon Endpoint when channel unavailable [int] {3.10}	Queue message [Queued n=1]	Queue message [Queued n>1]	Queue message [Queued n>1]
MakeConnection [msg] {3.10}		Send message [none]	Xmit message with MessagePending [if n=2 then (Queued n=1) else (Queued n>1)]

2369 Table 4 Receiving Endpoint Message Transfer Engine

Event	None	Polling
Expectation of unreceived message [int, unspecified]	No Action [Polling]	No Action [Same]
Polling trigger [int, unspecified]		Xmit MakeConnection [Polling] (3.10)

2370 **Appendix E. Acknowledgments**

2371 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2372 authors:

2373 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),
2374 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),
2375 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),
2376 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don
2377 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),
2378 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony
2379 Storey(IBM).

2380 The following individuals have provided invaluable input into the initial contribution:

2381 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen
2382 Brown(Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM),
2383 Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott
2384 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal
2385 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter
2386 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish
2387 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2388 The following individuals were members of the committee during the development of this specification:

2389 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben
2390 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd
2391 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul
2392 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques
2393 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),
2394 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair
2395 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),
2396 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul
2397 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt
2398 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff
2399 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku
2400 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert
2401 Pilz(BEA), Martin Raeppele(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom
2402 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von
2403 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçinalp(SAP), Nobuyuki
2404 Yamamoto(Hitachi).

Appendix F. Revision History

Rev	Date	By Whom	What
wd-01	2005-07-07	Christopher Ferris	Initial version created based on submission by the authors.
ws-02	2005-07-21	Doug Davis	I011 (PT0S) added
wd-02	2005-08-16	Anish Karmarkar	Trivial editorial changes
ws-03	2005-09-15	Doug Davis	I019 and i028 (CloseSeq) added
wd-05	2005-09-26	Gilbert Pilz	i005 (Source resend of nacks messages when ack already received) added.
wd-05	2005-09-27	Doug Davis	i027 (InOrder delivery assurance spanning multiple sequences) added
wd-05	2005-09-27	Doug Davis	i020 (Semantics of "At most once" Delivery Assurance) added
wd-05	2005-09-27	Doug Davis	i034 (Fault while processing a piggy-backed RM header) added
wd-05	2005-09-27	Doug Davis	i033 (Processing model of NACKs) added
wd-05	2005-09-27	Doug Davis	i031 (AckRequested schema inconsistency) added
wd-05	2005-09-27	Doug Davis	i025 (SeqAck/None) added
wd-05	2005-09-27	Doug Davis	i029 (Remove dependency on WS-Security) added
wd-05	2005-09-27	Doug Davis	i039 (What does 'have a mU attribute' mean) added
wd-05	2005-09-27	Doug Davis	i040 (Change 'optiona'/'required' to 'OPTIONAL'/'REQUIRED') added
wd-05	2005-09-30	Anish Karmarkar	i017 (Change NS to http://docs.oasis-open.org/wsrn/200510/)
wd-05	2005-09-30	Anish Karmarkar	i045 (Include SecureConversation as a reference and move it to non-normative citation)
wd-05	2005-09-30	Anish Karmarkar	i046 (change the type of wsrn:FaultCode element)
wd-06	2005-11-02	Gilbert Pilz	Start wd-06 by changing title page from cd-01.
wd-06	2005-11-03	Gilbert Pilz	i047 (Reorder spec sections)
wd-07	2005-11-17	Gilbert Pilz	Start wd-07
wd-07	2005-11-28	Doug Davis	i071 – except for period in Appendix headings
wd-07	2005-11-28	Doug Davis	i10
wd-07	2005-11-28	Doug Davis	i030
wd-07	2005-11-28	Doug Davis	i037
wd-07	2005-11-28	Doug Davis	i038
wd-07	2005-11-28	Doug Davis	i041
wd-07	2005-11-28	Doug Davis	i043
wd-07	2005-11-28	Doug Davis	i044

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

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

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

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

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

2405 **Appendix G. Notices**

2406 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2407 might be claimed to pertain to the implementation or use of the technology described in this document or
2408 the extent to which any license under such rights might or might not be available; neither does it represent
2409 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2410 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2411 available for publication and any assurances of licenses to be made available, or the result of an attempt
2412 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2413 users of this specification, can be obtained from the OASIS Executive Director.

2414 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2415 other proprietary rights which may cover technology that may be required to implement this specification.
2416 Please address the information to the OASIS Executive Director.

2417 Copyright (C) OASIS Open (2006). All Rights Reserved.

2418 This document and translations of it may be copied and furnished to others, and derivative works that
2419 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2420 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2421 this paragraph are included on all such copies and derivative works. However, this document itself may
2422 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2423 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2424 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2425 into languages other than English.

2426 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2427 or assigns.

2428 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2429 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2430 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2431 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.