



1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

3 Working Draft 16, October 31, 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 Yalçinalp, 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.....	25
65	4.1 SequenceFault Element.....	26
66	4.2 Sequence Terminated.....	27
67	4.3 Unknown Sequence.....	27
68	4.4 Invalid Acknowledgement.....	28
69	4.5 Message Number Rollover.....	28
70	4.6 Create Sequence Refused.....	29
71	4.7 Sequence Closed.....	29
72	4.8 WSRM Required.....	30
73	4.9 Unsupported Selection.....	30
74	5 Security Threats and Countermeasures.....	32
75	5.1 Threats and Countermeasures.....	32
76	5.1.1 Integrity Threats.....	32
77	5.1.1.1 Countermeasures.....	32
78	5.1.2 Resource Consumption Threats.....	33
79	5.1.2.1 Countermeasures.....	33

80	5.1.3 Sequence Spoofing Threats.....	33
81	5.1.3.1 Sequence Hijacking.....	33
82	5.1.3.2 Countermeasures.....	33
83	5.2 Security Solutions and Technologies.....	34
84	5.2.1 Transport Layer Security.....	34
85	5.2.1.1 Model.....	34
86	5.2.1.2 Countermeasure Implementation.....	35
87	5.2.2 SOAP Message Security.....	36
88	5.2.2.1 Model.....	36
89	5.2.2.2 Countermeasure Implementation.....	36
90	6 Securing Sequences.....	38
91	6.1 Securing Sequences Using WS-Security.....	38
92	6.2 Securing Sequences Using SSL/TLS.....	39
93	7 References.....	41
94	7.1 Normative.....	41
95	7.2 Non-Normative.....	41
96	Appendix A. Schema.....	43
97	Appendix B. WSDL.....	48
98	Appendix C. Message Examples.....	50
99	Appendix C.1 Create Sequence.....	50
100	Appendix C.2 Initial Transmission.....	50
101	Appendix C.3 First Acknowledgement.....	52
102	Appendix C.4 Retransmission.....	52
103	Appendix C.5 Termination.....	53
104	Appendix C.6 MakeConnection.....	54
105	Appendix D. State Tables.....	58
106	Appendix E. Acknowledgments.....	63
107	Appendix F. Revision History.....	64
108	Appendix G. Notices.....	70

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 wsm:
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 wsm:
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 Compliance**

159 An implementation is not compliant 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 compliant with this
162 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

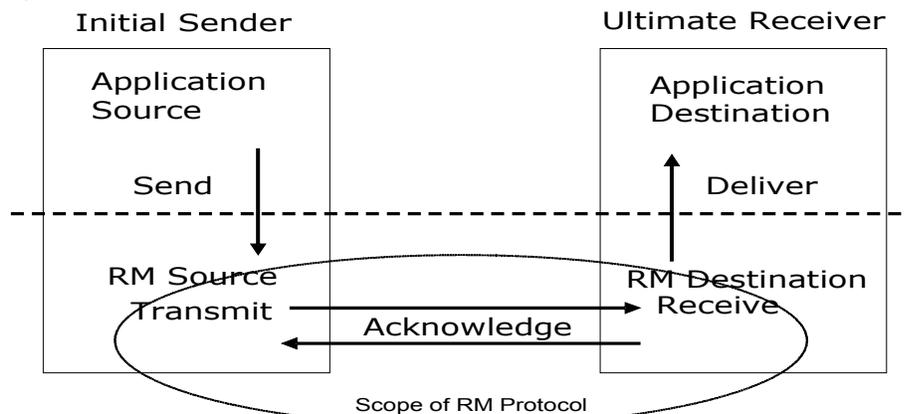
165

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

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

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

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

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

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

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

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

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

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

219 **2.2 Protocol Preconditions**

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

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

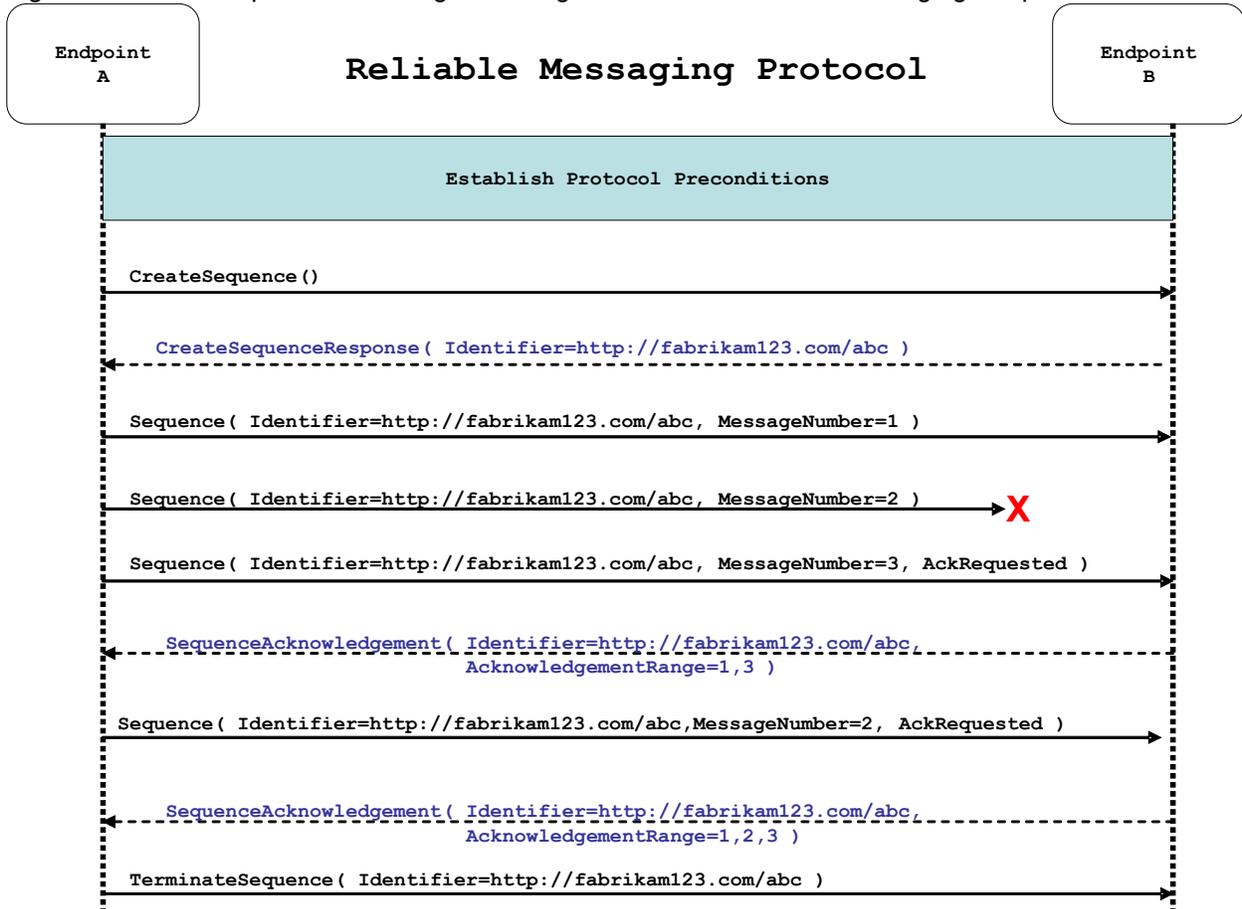
229 **2.3 Protocol Invariants**

230 During the lifetime of a `Sequence`, two invariants are **REQUIRED** for correctness:

- 231 • The RM Source MUST assign each message within a Sequence a message number (defined
- 232 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers
- 233 MUST be assigned in the same order in which messages are sent by the Application Source.
- 234 • Within every Acknowledgement Message it issues, the RM Destination MUST include one or more
- 235 `AcknowledgementRange` child elements that contain, in their collective ranges, the message
- 236 number of every message accepted by the RM Destination. The RM Destination MUST exclude, in
- 237 the `AcknowledgementRange` elements, the message numbers of any messages it has not
- 238 accepted.

239 2.4 Example Message Exchange

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



- 241 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
- 242 and establishing trust.
- 243 2. The RM Source requests creation of a new Sequence.
- 244 3. The RM Destination creates a new Sequence and returns its unique identifier.
- 245 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1.
- 246 In the figure above, the RM Source sends 3 messages in the Sequence.
- 247 5. The 2nd message in the Sequence is lost in transit.

- 248 6. The 3rd message is the last in this Sequence and the RM Source includes an `AckRequested`
249 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 250 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the
251 RM Source's `AckRequested` header.
- 252 8. The RM Source retransmits the unacknowledged message with `MessageNumber` 2. This is a new
253 message from the perspective of the underlying transport, but it has the same `Sequence Identifier`
254 and `MessageNumber` so the RM Destination can recognize it as a duplicate of the earlier message,
255 in case the original and retransmitted messages are both Received. The RM Source includes an
256 `AckRequested` header in the retransmitted message so the RM Destination will expedite an
257 acknowledgement.
- 258 9. The RM Destination Receives the second transmission of the message with `MessageNumber` 2
259 and acknowledges receipt of message numbers 1, 2, and 3.
- 260 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to the
261 RM Destination indicating that the Sequence is completed and reclaims any resources associated
262 with the Sequence.
- 263 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source will
264 not be sending any more messages. The RM Destination sends a `TerminateSequenceResponse`
265 message to the RM Source and reclaims any resources associated with the Sequence.
- 266 The RM Source will expect to Receive Acknowledgements from the RM Destination during the course of a
267 message exchange at occasions described in Section 3 below. Should an Acknowledgement not be
268 Received in a timely fashion, the RM Source MUST re-transmit the message since either the message or
269 the associated Acknowledgement might have been lost. Since the nature and dynamic characteristics of
270 the underlying transport and potential intermediaries are unknown in the general case, the timing of re-
271 transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been
272 demonstrated to cause transport or intermediary flooding which are counterproductive to the intention of
273 providing a reliable exchange of messages. Consequently, implementers are encouraged to utilize
274 adaptive mechanisms that dynamically adjust re-transmission time and the back-off intervals that are
275 appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP
276 transports, a mechanism similar to that described as RTTM in RFC 1323 [RTTM] SHOULD be
277 considered.
- 278 Now that the basic model has been outlined, the details of the elements used in this protocol are now
279 provided in Section 3.

280 **3 RM Protocol Elements**

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

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

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

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

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

293 **3.3 Composition with WS-Addressing**

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

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

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

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

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

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

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

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

311 **3.4 Sequence Creation**

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

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

319 The following exemplar defines the `CreateSequence` syntax:

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

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

335 `/wsmr:CreateSequence`

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

340 `/wsmr:CreateSequence/wsmr:AcksTo`

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

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

350 `/wsmr:CreateSequence/wsmr:Expires`

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

355 `/wsmr:CreateSequence/wsmr:Expires/@{any}`

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

358 `/wsmr:CreateSequence/wsmr:Offer`

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

361 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier`

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

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

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

367 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint

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

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

379 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires

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

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

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

386 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior

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

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

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

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

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

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

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

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

403 /wsmr:CreateSequence/{any}

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

406 /wsmr:CreateSequence/@{any}

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

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

413 The following exemplar defines the `CreateSequenceResponse` syntax:

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

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

427 /wsmr:CreateSequenceResponse

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

431 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

438 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

450 /wsmr:CreateSequenceResponse/wsmr:IncompleteSequenceBehavior

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

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

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

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

461 /wsmr:CreateSequenceResponse/wsmr:Accept

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

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

467 /wsmr:CreateSequenceResponse/wsmr:Accept/wsmr:AcksTo

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

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

476 /wsmr:CreateSequenceResponse/wsmr:Accept/{any}

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

479 /wsmr:CreateSequenceResponse/wsmr:Accept/@{any}

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

482 /wsmr:CreateSequenceResponse/{any}

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

485 /wsmr:CreateSequenceResponse/@{any}

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

488 3.5 Closing A Sequence

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

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

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

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

511 The following exemplar defines the `CloseSequence` syntax:

```
512 <wsrm:CloseSequence ...>  
513   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
514   ...  
515 </wsrm:CloseSequence>
```

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

517 `/wsrm:CloseSequence`

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

520 `/wsrm:CloseSequence/wsrm:Identifier`

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

524 `/wsrm:CloseSequence/wsrm:Identifier/@{any}`

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

527 `/wsrm:CloseSequence/{any}`

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

530 `/wsrm:CloseSequence@{any}`

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

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

536 The following exemplar defines the `CloseSequenceResponse` syntax:

```
537 <wsrm:CloseSequenceResponse ...>  
538   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
539   ...  
540 </wsrm:CloseSequenceResponse>
```

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

542 `/wsrm:CloseSequenceResponse`

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

545 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

549 `/wsrm:CloseSequenceResponse/wsrm:Identifier/@{any}`

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

552 `/wsrm:CloseSequenceResponse/{any}`

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

555 `/wsrm:CloseSequenceResponse@{any}`

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

558 **3.6 Sequence Termination**

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

566 The following exemplar defines the `TerminateSequence` syntax:

```
567 <wsrm:TerminateSequence ...>  
568   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
569   ...  
570 </wsrm:TerminateSequence>
```

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

572 `/wsrm:TerminateSequence`

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

578 /wsm:TerminateSequence/wsm:Identifier

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

582 /wsm:TerminateSequence/wsm:Identifier/@{any}

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

585 /wsm:TerminateSequence/{any}

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

588 /wsm:TerminateSequence/@{any}

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

591 A `TerminateSequenceResponse` is sent in the body of a response message by an RM Destination in
592 response to receipt of a `TerminateSequence` request message. It indicates that the RM Destination has
593 terminated the Sequence.

594 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
595 <wsm:TerminateSequenceResponse ...>  
596   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
597   ...  
598 </wsm:TerminateSequenceResponse>
```

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

600 /wsm:TerminateSequenceResponse

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

604 /wsm:TerminateSequenceResponse/wsm:Identifier

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

608 /wsm:TerminateSequenceResponse/wsm:Identifier/@{any}

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

611 /wsm:TerminateSequenceResponse/{any}

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

614 /wsmr:TerminateSequenceResponse/{any}

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

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

620 3.7 Sequences

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

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

628 A following exemplar defines its syntax:

```
629 <wsmr:Sequence ...>  
630   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
631   <wsmr:MessageNumber> wsmr:MessageNumberType </wsmr:MessageNumber>  
632   ...  
633 </wsmr:Sequence>
```

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

635 /wsmr:Sequence

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

642 /wsmr:Sequence/wsmr:Identifier

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

646 /wsmr:Sequence/wsmr:Identifier/{any}

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

649 /wsmr:Sequence/wsmr:MessageNumber

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

654 /wsmr:Sequence/{any}

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

657 /wsrm:Sequence/@{any}

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

660 The following example illustrates a Sequence header block.

```
661 <wsrm:Sequence>  
662   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
663   <wsrm:MessageNumber>10</wsrm:MessageNumber>  
664 </wsrm:Sequence>
```

665 3.8 Request Acknowledgement

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

668 The RM Source MAY request an Acknowledgement Message from the RM Destination at any time by
669 including an `AckRequested` header block in any message targeted to the RM Destination. An RM
670 Destination that Receives a message that contains an `AckRequested` header block MUST send a
671 message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference
672 (see Section 3.4) for a known Sequence or else generate an `UnknownSequence` fault. If a non-
673 mustUnderstand fault occurs when processing an RM header that was piggy-backed on another
674 message, a fault MUST be generated, but the processing of the original message MUST NOT be
675 affected. It is RECOMMENDED that the RM Destination return a `AcknowledgementRange` or `None`
676 element instead of a `Nack` element (see Section 3.9).

677 The following exemplar defines its syntax:

```
678 <wsrm:AckRequested ...>  
679   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
680   ...  
681 </wsrm:AckRequested>
```

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

683 /wsrm:AckRequested

684 This element requests an Acknowledgement for the identified Sequence.

685 /wsrm:AckRequested/wsrm:Identifier

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

689 /wsrm:AckRequested/wsrm:Identifier/@{any}

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

692 /wsrm:AckRequested/{any}

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

695 /wsrm:AckRequested/@{any}

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

698 **3.9 Sequence Acknowledgement**

699 The RM Destination informs the RM Source of successful message receipt using a
700 `SequenceAcknowledgement` header block. The RM Destination MAY Transmit the
701 `SequenceAcknowledgement` header block independently or it MAY include the
702 `SequenceAcknowledgement` header block on any message targeted to the `AcksTo` EPR.
703 Acknowledgements can be explicitly requested using the `AckRequested` directive (see Section 3.8). If a
704 non-mustUnderstand fault occurs when processing an RM header that was piggy-backed on another
705 message, a fault MUST be generated, but the processing of the original message MUST NOT be
706 affected.

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

709 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
710 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
711 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
712 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
713 on the protocol binding-specific channel. Such a channel is provided by the context of a Received
714 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
715 header block for that same Sequence identifier.

716 The following exemplar defines its syntax:

```
717 <wsrm:SequenceAcknowledgement ...>  
718   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
719   [ [ [ <wsrm:AcknowledgementRange ...  
720         Upper="wsrm:MessageNumberType"  
721         Lower="wsrm:MessageNumberType" /> +  
722         | <wsrm:None/> ]  
723         <wsrm:Final/> ? ]  
724         | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]  
725   ...  
726   ...  
727 </wsrm:SequenceAcknowledgement>
```

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

729 `/wsrm:SequenceAcknowledgement`

730 This element contains the Sequence Acknowledgement information.

731 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

740 /wsmr:SequenceAcknowledgement/wsmr:AcknowledgementRange
741 The RM Destination MAY include one or more instances of this element within a
742 SequenceAcknowledgement header block. It contains a range of Sequence message numbers
743 successfully accepted by the RM Destination. The ranges MUST NOT overlap. The RM Destination
744 MUST NOT include this element if a sibling Nack or None element is also present as a child of
745 SequenceAcknowledgement.

746 /wsmr:SequenceAcknowledgement/wsmr:AcknowledgementRange/@Upper
747 The RM Destination MUST set the value of this attribute equal to the message number of the highest
748 contiguous message in a Sequence range accepted by the RM Destination.

749 /wsmr:SequenceAcknowledgement/wsmr:AcknowledgementRange/@Lower
750 The RM Destination MUST set the value of this attribute equal to the message number of the lowest
751 contiguous message in a Sequence range accepted by the RM Destination.

752 /wsmr:SequenceAcknowledgement/wsmr:AcknowledgementRange/@{any}
753 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
754 element.

755 /wsmr:SequenceAcknowledgement/wsmr:None
756 The RM Destination MUST include this element within a SequenceAcknowledgement header block if
757 the RM Destination has not accepted any messages for the specified Sequence. The RM Destination
758 MUST NOT include this element if a sibling AcknowledgementRange or Nack element is also present
759 as a child of the SequenceAcknowledgement.

760 /wsmr:SequenceAcknowledgement/wsmr:Final
761 The RM Destination MAY include this element within a SequenceAcknowledgement header block. This
762 element indicates that the RM Destination is not receiving new messages for the specified Sequence. The
763 RM Source can be assured that the ranges of messages acknowledged by this
764 SequenceAcknowledgement header block will not change in the future. The RM Destination MUST
765 include this element when the Sequence is closed. The RM Destination MUST NOT include this element
766 when sending a Nack; it can only be used when sending AcknowledgementRange elements or a None.

767 /wsmr:SequenceAcknowledgement/wsmr:Nack
768 The RM Destination MAY include this element within a SequenceAcknowledgement header block. If
769 used, the RM Destination MUST set the value of this element to a MessageNumberType representing
770 the MessageNumber of an unreceived message in a Sequence. The RM Destination MUST NOT include
771 a Nack element if a sibling AcknowledgementRange or None element is also present as a child of
772 SequenceAcknowledgement. Upon the receipt of a Nack, an RM Source SHOULD retransmit the
773 message identified by the Nack. The RM Destination MUST NOT issue a SequenceAcknowledgement
774 containing a Nack for a message that it has previously acknowledged within a
775 AcknowledgementRange. The RM Source SHOULD ignore a SequenceAcknowledgement containing
776 a Nack for a message that has previously been acknowledged within a AcknowledgementRange.

777 /wsmr:SequenceAcknowledgement/{any}
778 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
779 to be passed.

780 /wsmr:SequenceAcknowledgement/@{any}

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

783 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
785 <wsrm:SequenceAcknowledgement>  
786   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
787   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
788 </wsrm:SequenceAcknowledgement>
```

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

```
791 <wsrm:SequenceAcknowledgement>  
792   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
793   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
794   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
795   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
796 </wsrm:SequenceAcknowledgement>
```

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

```
798 <wsrm:SequenceAcknowledgement>  
799   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
800   <wsrm:Nack>3</wsrm:Nack>  
801 </wsrm:SequenceAcknowledgement>
```

802 3.10 MakeConnection

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

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

808 This URI template in an EPR indicates a protocol-specific back-channel will be established through a
809 mechanism such as `MakeConnection`, defined below. When using this URI template, “{uuid}” MUST be
810 replaced by a UUID value as defined by RFC4122[UUID]. This UUID value uniquely distinguishes the
811 Endpoint. A sending Endpoint SHOULD Transmit messages at Endpoints identified with the URI template
812 using a protocol-specific back-channel, including but not limited to those established with a
813 `MakeConnection` message. Note, this URI is semantically similar to the WS-Addressing anonymous
814 URI if a protocol-specific back-channel is available.

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

821 The following exemplar defines the `MakeConnection` syntax:

```
822 <wsrm:MakeConnection ...>  
823   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
824   <wsrm:Address ...> xs:anyURI </wsrm:Address> ?  
825   ...  
826 </wsrm:MakeConnection>
```

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

828 /wsrm:MakeConnection

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

831 /wsrm:MakeConnection/wsrm:Identifier

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

837 /wsrm:MakeConnection/wsrm:Identifier/@{any}

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

840 /wsrm:MakeConnection/wsrm:Address

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

848 /wsrm:MakeConnection/wsrm:Address/@{any}

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

851 /wsrm:MakeConnection/{any}

852 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
853 to be passed. This allows fine-tuning of the messages to be returned, additional selection criteria included
854 here are logically ANDed with the *Address* and/or *Identifier*. If an extension is not supported by the
855 Endpoint then it should return a *UnsupportedSelection fault*.

856 /wsrm:MakeConnection/@{any}

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

859 If both *Identifier* and *Address* are present, then the Endpoint processing the *MakeConnection*
860 message MUST insure that any SOAP Envelope flowing on the backchannel MUST be associated with
861 the given Sequence and MUST be addressed to the given URI.

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

867 This specification places no constraint on the types of messages that can be returned on the transport-
868 specific back-channel. As in an asynchronous environment, it is up to the recipient of the
869 *MakeConnection* message to decide which messages are appropriate for transmission to any particular

870 Endpoint. However, the Endpoint processing the `MakeConnection` message MUST insure that the
871 messages match the selection criteria as specified by the child elements of the `MakeConnection`
872 element.

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

- 875 ● The `MakeConnection` message is logically part of a one-way operation; there is no reply
876 message to the `MakeConnection` itself, and any response flowing on the transport back-channel
877 is a pending message.
- 878 ● Since there is no reply message to `MakeConnection`, the WS-Addressing specific rules in
879 section 3.4 "Formulating a Reply Message" are not used. Therefore, the value of any
880 `wsa:ReplyTo` element in the `MakeConnection` message has no effective impact since the WS-
881 Addressing [`reply endpoint`] property that is set by the presence of `wsa:ReplyTo` is not
882 used.
- 883 ● In the absence of any pending message, there will be no message transmitted on the transport
884 back-channel. E.g. In the HTTP case just an `HTTP 202 Accepted` will be returned without any
885 SOAP envelope in the HTTP response message.
- 886 ● When there is a message pending, it is sent on the transport back-channel, using the connection
887 that has been initiated by the `MakeConnection` request.

888 3.11 MessagePending

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

893 The following exemplar defines the `MessagePending` syntax:

```
894 <wsrm:MessagePending pending="xs:boolean" ...>  
895   ...  
896 </wsrm:MessagePending>
```

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

898 `/wsrm:MessagePending`

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

900 `/wsrm:MessagePending@pending`

901 This attribute, when set to "true", indicates that there is at least one message waiting to be retrieved.
902 When this attribute is set to "false" it indicates there are currently no messages waiting to be retrieved.

903 `/wsrm:MessagePending/{any}`

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

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

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

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

911 4 Faults

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

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

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

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

925 The definitions of faults use the following properties:

926 [Code] The fault code.

927 [Subcode] The fault subcode.

928 [Reason] The English language reason element.

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

932 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
933 "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

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

```
935 <S:Envelope>  
936   <S:Header>  
937     <wsa:Action>  
938       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault  
939     </wsa:Action>  
940     <!-- Headers elided for brevity. -->  
941   </S:Header>  
942   <S:Body>  
943     <S:Fault>  
944       <S:Code>  
945         <S:Value> [Code] </S:Value>  
946         <S:Subcode>  
947           <S:Value> [Subcode] </S:Value>  
948         </S:Subcode>  
949       </S:Code>  
950       <S:Reason>  
951         <S:Text xml:lang="en"> [Reason] </S:Text>  
952       </S:Reason>  
953       <S:Detail>  
954         [Detail]
```

```

955     ...
956     </S:Detail>
957 </S:Fault>
958 </S:Body>
959 </S:Envelope>

```

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

```

962 <S11:Envelope>
963   <S11:Header>
964     <wsrm:SequenceFault>
965       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
966       <wsrm:Detail> [Detail] </wsrm:Detail>
967       ...
968     </wsrm:SequenceFault>
969     <!-- Headers elided for brevity. -->
970   </S11:Header>
971   <S11:Body>
972     <S11:Fault>
973       <faultcode> [Code] </faultcode>
974       <faultstring> [Reason] </faultstring>
975     </S11:Fault>
976   </S11:Body>
977 </S11:Envelope>

```

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

```

980 <S11:Envelope>
981   <S11:Body>
982     <S11:Fault>
983       <faultcode> [Subcode] </faultcode>
984       <faultstring> [Reason] </faultstring>
985     </S11:Fault>
986   </S11:Body>
987 </S11:Envelope>

```

988 4.1 SequenceFault Element

989 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during
990 the reliable messaging specific processing of a message belonging to a Sequence. WS-
991 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP
992 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in
993 conjunction with the SOAP 1.2 binding.

994 The following exemplar defines its syntax:

```

995 <wsrm:SequenceFault ...>
996   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
997   <wsrm:Detail> ... </wsrm:Detail> ?
998   ...
999 </wsrm:SequenceFault>

```

1000 The following describes the content model of the `SequenceFault` element.

1001 /wsrm:SequenceFault

1002 This is the element containing Sequence information for WS-ReliableMessaging

1003 /wsrm:SequenceFault/wsrm:FaultCode

1004 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a
1005 qualified name from the set of fault [Subcodes] defined below.

1006 `/wsrm:SequenceFault/wsrn:Detail`

1007 This element, if present, carries application specific error information related to the fault being described.

1008 `/wsrm:SequenceFault/wsrn:Detail/{any}`

1009 The application specific error information related to the fault being described.

1010 `/wsrm:SequenceFault/wsrn:Detail/@{any}`

1011 The application specific error information related to the fault being described.

1012 `/wsrm:SequenceFault/{any}`

1013 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
1014 to be passed.

1015 `/wsrm:SequenceFault/@{any}`

1016 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
1017 element.

1018 **4.2 Sequence Terminated**

1019 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
1020 Endpoint of this decision.

1021 Properties:

1022 [Code] Sender or Receiver

1023 [Subcode] `wsrn:SequenceTerminated`

1024 [Reason] The Sequence has been terminated due to an unrecoverable error.

1025 [Detail]

1026 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

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

1027 **4.3 Unknown Sequence**

1028 Properties:

1029 [Code] Sender

1030 [Subcode] `wsrn:UnknownSequence`

1031 [Reason] The value of `wsrn:Identifier` is not a known Sequence identifier.

1032 [Detail]

1033 `<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.

1034 4.4 Invalid Acknowledgement

1035 An example of when this fault is generated is when a message is Received by the RM Source containing
1036 a `SequenceAcknowledgement` covering messages that have not been sent.

1037 [Code] Sender

1038 [Subcode] `wsrm:InvalidAcknowledgement`

1039 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1040 [Detail]

1041 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

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

1042 4.5 Message Number Rollover

1043 If the condition listed below is reached, the RM Destination MUST generate this fault.

1044 Properties:

1045 [Code] Sender

1046 [Subcode] `wsrm:MessageNumberRollover`

1047 [Reason] The maximum value for `wsrm:MessageNumber` has been exceeded.

1048 [Detail]

1049 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`
 1050 `<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.

1051 4.6 Create Sequence Refused

1052 Properties:

1053 [Code] Sender

1054 [Subcode] wsrm:CreateSequenceRefused

1055 [Reason] The Create Sequence request has been refused by the RM Destination.

1056 [Detail]

1057 `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.

1058 4.7 Sequence Closed

1059 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1060 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
 1061 is closed.

1062 Properties:

1063 [Code] Sender

1064 [Subcode] wsrm:SequenceClosed

1065 [Reason] The Sequence is closed and can not accept new messages.

1066 [Detail]

1067 `<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.

1068 4.8 WSRM Required

1069 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1070 message that did not use this protocol.

1071 Properties:

1072 [Code] Sender

1073 [Subcode] wsrn:WSRMRequired

1074 [Reason] The RM Destination requires the use of WSRM.

1075 [Detail]

1076 `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.

1077 4.9 Unsupported Selection

1078 The QName of the unsupported element(s) are included in the detail.

1079 Properties:

1080 [Code] Receiver

1081 [Subcode] wsrn:UnsupportedSelection

1082 [Reason] The extension element used in the message selection is not supported by the RM Source

1083 [Detail]

1084 `<wsrn:UnsupportedElement> xs:QName </wsrn: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.

1085 **5 Security Threats and Countermeasures**

1086 This specification considers two sets of security requirements, those of the applications that use the WS-
1087 RM protocol and those of the protocol itself.

1088 This specification makes no assumptions about the security requirements of the applications that use WS-
1089 RM. However, once those requirements have been satisfied within a given operational context, the
1090 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1091 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1092 There are many other security concerns that one may need to consider when implementing or using this
1093 protocol. The material below should not be considered as a "check list". Implementers and users of this
1094 protocol are urged to perform a security analysis to determine their particular threat profile and the
1095 appropriate responses to those threats.

1096 Implementers are also advised that there is a core tension between security and reliable messaging that
1097 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1098 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1099 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1100 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1101 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1102 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1103 avoid and prevent this condition.

1104 **5.1 Threats and Countermeasures**

1105 The primary security requirement of this protocol is to protect the specified semantics and protocol
1106 invariants against various threats. The following sections describe several threats to the integrity and
1107 operation of this protocol and provide some general outlines of countermeasures to those threats.
1108 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1109 to all operational contexts.

1110 **5.1.1 Integrity Threats**

1111 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1112 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1113 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1114 to its intended message represents a threat to the WS-RM protocol.

1115 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM
1116 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1117 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be
1118 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1119 **5.1.1.1 Countermeasures**

1120 Integrity threats are generally countered via the use of digital signatures some level of the communication
1121 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1122 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers
1123 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which
1124 they occur, implementations MUST allow for signatures that cover only these headers.

1125 **5.1.2 Resource Consumption Threats**

1126 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1127 implement that RM Destination. These resources can include network connections, database tables,
1128 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1129 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1130 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1131 message Delivery and use this Sequence to send a stream of very large messages to that service,
1132 making sure to omit message number “1” from that stream.

1133 **5.1.2.1 Countermeasures**

1134 There are a number of countermeasures against the described resource consumption threats. The
1135 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1136 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1137 some cases, allows the identity of any attackers to be determined.

1138 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and
1139 authenticate the RM Source that issued the `CreateSequence` message.

1140 **5.1.3 Sequence Spoofing Threats**

1141 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1142 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1143 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1144 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the
1145 current `MessageNumber` for their target Sequence.

1146 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP
1147 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier
1148 to attack the Sequence. These attacks are “two-way” in that an attacker may choose to target the RM
1149 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends
1150 to the `AcksTo` EPR of an RM Source.

1151 **5.1.3.1 Sequence Hijacking**

1152 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject
1153 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1154 messages.

1155 Note that “sequence hijacking” should not be equated with “security session hijacking”. Although a
1156 Sequence may be bound to some form of a security session in order to counter the threats described in
1157 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1158 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1159 established by the security infrastructure to make such determinations. Failure to observe this rule
1160 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1161 the ability to authenticate its peers even though the necessary security processing has taken place.

1162 **5.1.3.2 Countermeasures**

1163 There are a number of countermeasures against sequence spoofing threats. The technique advocated by
1164 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1165 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1166 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1167 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1168 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1169 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1170 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1171 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1172 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1173 sequence peer it MUST be able to identify and authenticate the entity that sent the
1174 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1175 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1176 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1177 creation time.

1178 **5.2 Security Solutions and Technologies**

1179 The security threats described in the previous sections are neither new nor unique. The solutions that
1180 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1181 section maps the facilities provided by common web services security solutions against countermeasures
1182 described in the previous sections.

1183 Before continuing this discussion, however, some examination of the underlying requirements of the
1184 previously described countermeasures is necessary. Specifically it should be noted that the technique
1185 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1186 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check
1187 against this authenticated identity and determines if the RM Source is permitted to create Sequences with
1188 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,
1189 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of
1190 such facilities is considered to be beyond the scope of this specification.

1191 **5.2.1 Transport Layer Security**

1192 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1193 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1194 defined in Section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1195 The description provided here is general in nature and is not intended to serve as a complete definition on
1196 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1197 choice of features as well as the manner in which they will be used. The mechanisms described in the
1198 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1199 requirements and constraints of the use of SSL/TLS.

1200 **5.2.1.1 Model**

1201 The basic model for using SSL/TLS is as follows:

- 1202 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1203 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1204 Destination.

- 1205 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1206 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1207 synchronous `CreateSequenceResponse` using the session established in (1).
- 1208 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1209 any and all messages or faults that refer to that Sequence.
- 1210 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1211 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1212 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1213 5.2.1.2 Countermeasure Implementation

1214 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1215 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1216 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1217 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1218 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1219 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1220 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1221 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1222 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1223 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1224 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1225 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1226 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1227 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1228 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1229 Acknowledgement) using BasicAuth.
- 1230 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1231 connection authenticates itself to the party accepting the connection using an X.509 certificate
1232 that is exchanged during the SSL/TLS handshake.

1233 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1234 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1235 Source is authorized to create a Sequence with the RM Destination.

1236 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1237 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1238 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1239 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1240 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1241 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1242 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1243 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1244 to protect that Sequence.

1245 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1246 countermeasures (such as associating specific authentication information with a Sequence) although such
1247 methods are not covered by this document.

1248 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1249 session) are outside the scope of this specification.

1250 **5.2.2 SOAP Message Security**

1251 The mechanisms described in WS-Security may be used in various ways to implement the
1252 countermeasures described in the previous sections. This specification advocates using the protocol
1253 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust
1254 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1255 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1256 The description provided here is general in nature and is not intended to serve as a complete definition on
1257 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1258 need to agree on the choice of features as well as the manner in which they will be used. The
1259 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1260 describe the requirements and constraints of the use of WS-SecureConversation.

1261 **5.2.2.1 Model**

1262 The basic model for using WS-SecureConversation is as follows:

- 1263 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This
1264 may involve the participation of third parties such as a security token service. The tokens
1265 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1266 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1267 context that will be used to protect the Sequence. This is done so that, in cases where the
1268 `CreateSequence` message is signed by more than one security context, the RM Source can
1269 indicate which security context should be used to protect the newly created Sequence.
- 1270 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1271 associated with the security context to sign (as defined by WS-Security) at least the body and any
1272 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

1273 **5.2.2.2 Countermeasure Implementation**

1274 Without relying upon any authentication information, the per-message signatures provide the necessary
1275 integrity qualities to counter the threats described in Section 5.1.1.

1276 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1277 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1278 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1279 create a Sequence with the RM Destination.

1280 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1281 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1282 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1283 context rather than on any authentication claims that may have been established during security context
1284 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1285 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1286 document.

1287 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer
1288 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1289 the association between a Sequence and its protecting security context cannot always be established
1290 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1291 `CreateSequenceResponse` messages may be signed by more than one security context.

1292 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1293 amending or renewing contexts) are outside the scope of this specification.

1294 6 Securing Sequences

1295 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared
1296 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1297 achieving this objective depending upon the underlying security infrastructure.

1298 6.1 Securing Sequences Using WS-Security

1299 One mechanism for protecting a Sequence is to include a security token using a
1300 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1301 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1302 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1303 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1304 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1305 there may be more than one token on a `CreateSequence` message or inferred from the communication
1306 context (e.g. transport protection).

1307 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1308 if the token being referenced supports such mechanism.

1309 The following exemplar defines the `CreateSequence` syntax when extended to include a
1310 `wsse:SecurityTokenReference`:

```
1311 <wsrm:CreateSequence ...>  
1312   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1313   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1314   <wsrm:Offer ...>  
1315     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1316     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1317     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1318     <wsrm:IncompleteSequenceBehavior>  
1319       wsrml:IncompleteSequenceBehaviorType  
1320     </wsrm:IncompleteSequenceBehavior> ?  
1321     ...  
1322   </wsrm:Offer> ?  
1323   ...  
1324   <wsse:SecurityTokenReference>  
1325     ...  
1326   </wsse:SecurityTokenReference> ?  
1327   ...  
1328 </wsrm:CreateSequence>
```

1329 The following describes the content model of the additional `CreateSequence` elements.

1330 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1331 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in
1332 section 3.4) to communicate an explicit reference to the security token, using a
1333 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and Destination
1334 MUST use to authorize messages for the created (and, if present, the offered) Sequence(s). All
1335 subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST
1336 demonstrate proof-of-possession of the secret associated with the token (e.g., by using or deriving from a
1337 private or secret key).

1338 When a RM Source transmits a `CreateSequence` that has been extended to include a
1339 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and
1340 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1341 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This

1342 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1343 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1344 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1345 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1346 in WS-Security still applies.

1347 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1348 <wsrm:UsesSequenceSTR ... />
```

1349 The following describes the content model of the `UsesSequenceSTR` header block.

1350 `/wsrm:UsesSequenceSTR`

1351 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1352 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1353 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1354 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1355 Sequence creation.

1356 The following is an example of a `CreateSequence` message using the
1357 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1358 <soap:Envelope ...>  
1359   <soap:Header>  
1360     ...  
1361     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1362     ...  
1363   </soap:Header>  
1364   <soap:Body>  
1365     <wsrm:CreateSequence>  
1366       <wsrm:AcksTo>  
1367         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1368       </wsrm:AcksTo>  
1369       <wsse:SecurityTokenReference>  
1370         ...  
1371       </wsse:SecurityTokenReference>  
1372     </wsrm:CreateSequence>  
1373   </soap:Body>  
1374 </soap:Envelope>
```

1375 6.2 Securing Sequences Using SSL/TLS

1376 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1377 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1378 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1379 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1380 SOAP header block within the `CreateSequence` message.

1381 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1382 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1383 The following describes the content model of the `UsesSequenceSSL` header block.

1384 `/wsrm:UsesSequenceSSL`

1385 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1386 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was
1387 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a
1388 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand

1389 and correctly implement the functionality described in Section 5.2.1 or else generate a
1390 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1391 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1392 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1393 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1394 `CreateSequenceResponse` message.

1395 **7 References**

1396 **7.1 Normative**

1397 **[KEYWORDS]**

1398 S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University,
1399 March 1997

1400 <http://www.ietf.org/rfc/rfc2119.txt>

1401 **[SOAP 1.1]**

1402 W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

1403 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1404 **[SOAP 1.2]**

1405 W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June 2003.

1406 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

1407 **[URI]**

1408 T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 3986,
1409 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1410 <http://ietf.org/rfc/rfc3986>

1411 **[UUID]**

1412 P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN Namespace," RFC 4122,
1413 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1414 <http://www.ietf.org/rfc/rfc4122.txt>

1415 **[XML]**

1416 W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", September 2006.

1417 <http://www.w3.org/TR/REC-xml/>

1418 **[XML-ns]**

1419 W3C Recommendation, "Namespaces in XML," 14 January 1999.

1420 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

1421 **[XML-Schema Part1]**

1422 W3C Recommendation, "XML Schema Part 1: Structures," October 2004.

1423 <http://www.w3.org/TR/xmlschema-1/>

1424 **[XML-Schema Part2]**

1425 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

1426 <http://www.w3.org/TR/xmlschema-2/>

1427 **[XPath 1.0]**

1428 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

1429 <http://www.w3.org/TR/xpath>

1430 **[WSDL 1.1]**

1431 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

1432 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1433 **[WS-Addressing]**

1434 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

1435 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

1436 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

1437 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1438 **7.2 Non-Normative**

1439 **[BSP 1.0]**

1440 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

1441 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

1442 **[RDDL 2.0]**

1443 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

1444 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

1445 **[RFC 2617]**

1446 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
1447 Authentication: Basic and Digest Access Authentication," June 1999.

1448 <http://www.ietf.org/rfc/rfc2617.txt>

1449 **[RFC 4346]**

1450 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

1451 <http://www.ietf.org/rfc/rfc4346.txt>

1452 **[WS-Policy]**

1453 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

1454 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

1455 **[WS-PolicyAttachment]**

1456 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

1457 <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment->

1458 [20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1459 **[WS-Security]**

1460 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1461 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1462 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

- 1463 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1464 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.
1465 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 1466 **[RTTM]**
- 1467 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May
1468 1992.
1469 <http://www.rfc-editor.org/rfc/rfc1323.txt>
- 1470 **[SecurityPolicy]**
- 1471 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005
1472 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
- 1473 **[SecureConversation]**
- 1474 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February
1475 2005.
1476 <http://schemas.xmlsoap.org/ws/2004/04/sc/>
- 1477 **[Trust]**
- 1478 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.
1479 <http://schemas.xmlsoap.org/ws/2005/02/trust>

1480 Appendix A. Schema

1481 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1482 Schema Part2] is located at:

1483 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

1484 The following copy is provided for reference.

```
1485 <?xml version="1.0" encoding="UTF-8"?>
1486 <!--
1487 OASIS takes no position regarding the validity or scope of any intellectual
1488 property or other rights that might be claimed to pertain to the
1489 implementation or use of the technology described in this document or the
1490 extent to which any license under such rights might or might not be available;
1491 neither does it represent that it has made any effort to identify any such
1492 rights. Information on OASIS's procedures with respect to rights in OASIS
1493 specifications can be found at the OASIS website. Copies of claims of rights
1494 made available for publication and any assurances of licenses to be made
1495 available, or the result of an attempt made to obtain a general license or
1496 permission for the use of such proprietary rights by implementors or users of
1497 this specification, can be obtained from the OASIS Executive Director.
1498 OASIS invites any interested party to bring to its attention any copyrights,
1499 patents or patent applications, or other proprietary rights which may cover
1500 technology that may be required to implement this specification. Please
1501 address the information to the OASIS Executive Director.
1502 Copyright &copy; OASIS Open 2002-2006. All Rights Reserved.
1503 This document and translations of it may be copied and furnished to others,
1504 and derivative works that comment on or otherwise explain it or assist in its
1505 implementation may be prepared, copied, published and distributed, in whole or
1506 in part, without restriction of any kind, provided that the above copyright
1507 notice and this paragraph are included on all such copies and derivative
1508 works. However, this document itself does not be modified in any way, such as
1509 by removing the copyright notice or references to OASIS, except as needed for
1510 the purpose of developing OASIS specifications, in which case the procedures
1511 for copyrights defined in the OASIS Intellectual Property Rights document must
1512 be followed, or as required to translate it into languages other than English.
1513 The limited permissions granted above are perpetual and will not be revoked by
1514 OASIS or its successors or assigns.
1515 This document and the information contained herein is provided on an "AS IS"
1516 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1517 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1518 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1519 FOR A PARTICULAR PURPOSE.
1520 -->
1521 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1522 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1523 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1524 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1525 elementFormDefault="qualified" attributeFormDefault="unqualified">
1526 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1527 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1528 <!-- Protocol Elements -->
1529 <xs:complexType name="SequenceType">
1530 <xs:sequence>
1531 <xs:element ref="wsrm:Identifier"/>
1532 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1533 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1534 maxOccurs="unbounded"/>
1535 </xs:sequence>
1536 <xs:anyAttribute namespace="##other" processContents="lax"/>
```

```

1537 </xs:complexType>
1538 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1539 <xs:element name="SequenceAcknowledgement">
1540   <xs:complexType>
1541     <xs:sequence>
1542       <xs:element ref="wsrm:Identifier"/>
1543       <xs:choice>
1544         <xs:sequence>
1545           <xs:choice>
1546             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1547               <xs:complexType>
1548                 <xs:sequence/>
1549                 <xs:attribute name="Upper" type="xs:unsignedLong"
1550 use="required"/>
1551                 <xs:attribute name="Lower" type="xs:unsignedLong"
1552 use="required"/>
1553                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1554               </xs:complexType>
1555             </xs:element>
1556             <xs:element name="None">
1557               <xs:complexType>
1558                 <xs:sequence/>
1559               </xs:complexType>
1560             </xs:element>
1561           </xs:choice>
1562           <xs:element name="Final" minOccurs="0">
1563             <xs:complexType>
1564               <xs:sequence/>
1565             </xs:complexType>
1566           </xs:element>
1567         </xs:sequence>
1568         <xs:element name="Nack" type="xs:unsignedLong"
1569 maxOccurs="unbounded"/>
1570       </xs:choice>
1571       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1572 maxOccurs="unbounded"/>
1573     </xs:sequence>
1574     <xs:anyAttribute namespace="##other" processContents="lax"/>
1575   </xs:complexType>
1576 </xs:element>
1577 <xs:complexType name="AckRequestedType">
1578   <xs:sequence>
1579     <xs:element ref="wsrm:Identifier"/>
1580     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1581 maxOccurs="unbounded"/>
1582   </xs:sequence>
1583   <xs:anyAttribute namespace="##other" processContents="lax"/>
1584 </xs:complexType>
1585 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1586 <xs:complexType name="MessagePendingType">
1587   <xs:sequence>
1588     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1589 maxOccurs="unbounded"/>
1590   </xs:sequence>
1591   <xs:attribute name="pending" type="xs:boolean"/>
1592   <xs:anyAttribute namespace="##other" processContents="lax"/>
1593 </xs:complexType>
1594 <xs:element name="MessagePending" type="wsrm:MessagePendingType"/>
1595 <xs:element name="Identifier">
1596   <xs:complexType>
1597     <xs:annotation>
1598       <xs:documentation>
1599         This type is for elements whose [children] is an anyURI and can have

```

```

1600 arbitrary attributes.
1601     </xs:documentation>
1602     </xs:annotation>
1603     <xs:simpleContent>
1604         <xs:extension base="xs:anyURI">
1605             <xs:anyAttribute namespace="##other" processContents="lax"/>
1606         </xs:extension>
1607     </xs:simpleContent>
1608 </xs:complexType>
1609 </xs:element>
1610 <xs:element name="Address">
1611     <xs:complexType>
1612         <xs:simpleContent>
1613             <xs:extension base="xs:anyURI">
1614                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1615             </xs:extension>
1616         </xs:simpleContent>
1617     </xs:complexType>
1618 </xs:element>
1619 <xs:complexType name="MakeConnectionType">
1620     <xs:sequence>
1621         <xs:element ref="wsrm:Identifier" minOccurs="0" maxOccurs="1"/>
1622         <xs:element ref="wsrm:Address" minOccurs="0" maxOccurs="1"/>
1623         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1624 maxOccurs="unbounded"/>
1625     </xs:sequence>
1626     <xs:anyAttribute namespace="##other" processContents="lax"/>
1627 </xs:complexType>
1628 <xs:element name="MakeConnection" type="wsrm:MakeConnectionType"/>
1629 <xs:simpleType name="MessageNumberType">
1630     <xs:restriction base="xs:unsignedLong">
1631         <xs:minInclusive value="1"/>
1632         <xs:maxInclusive value="9223372036854775807"/>
1633     </xs:restriction>
1634 </xs:simpleType>
1635 <!-- Fault Container and Codes -->
1636 <xs:simpleType name="FaultCodes">
1637     <xs:restriction base="xs:QName">
1638         <xs:enumeration value="wsrm:SequenceTerminated"/>
1639         <xs:enumeration value="wsrm:UnknownSequence"/>
1640         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1641         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1642         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1643         <xs:enumeration value="wsrm:SequenceClosed"/>
1644         <xs:enumeration value="wsrm:WSRMRequired"/>
1645         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1646     </xs:restriction>
1647 </xs:simpleType>
1648 <xs:complexType name="SequenceFaultType">
1649     <xs:sequence>
1650         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1651         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1652         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1653 maxOccurs="unbounded"/>
1654     </xs:sequence>
1655     <xs:anyAttribute namespace="##other" processContents="lax"/>
1656 </xs:complexType>
1657 <xs:complexType name="DetailType">
1658     <xs:sequence>
1659         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1660 maxOccurs="unbounded"/>
1661     </xs:sequence>
1662     <xs:anyAttribute namespace="##other" processContents="lax"/>

```

```

1663     </xs:complexType>
1664     <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1665     <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1666     <xs:element name="CreateSequenceResponse"
1667 type="wsrm:CreateSequenceResponseType"/>
1668     <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1669     <xs:element name="CloseSequenceResponse"
1670 type="wsrm:CloseSequenceResponseType"/>
1671     <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1672     <xs:element name="TerminateSequenceResponse"
1673 type="wsrm:TerminateSequenceResponseType"/>
1674     <xs:complexType name="CreateSequenceType">
1675       <xs:sequence>
1676         <xs:element ref="wsrm:AcksTo"/>
1677         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1678         <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1679         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1680 maxOccurs="unbounded">
1681           <xs:annotation>
1682             <xs:documentation>
1683               It is the authors intent that this extensibility be used to
1684 transfer a Security Token Reference as defined in WS-Security.
1685             </xs:documentation>
1686           </xs:annotation>
1687         </xs:any>
1688       </xs:sequence>
1689       <xs:anyAttribute namespace="##other" processContents="lax"/>
1690     </xs:complexType>
1691     <xs:complexType name="CreateSequenceResponseType">
1692       <xs:sequence>
1693         <xs:element ref="wsrm:Identifier"/>
1694         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1695         <xs:element name="IncompleteSequenceBehavior"
1696 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1697         <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1698         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1699 maxOccurs="unbounded"/>
1700       </xs:sequence>
1701       <xs:anyAttribute namespace="##other" processContents="lax"/>
1702     </xs:complexType>
1703     <xs:complexType name="CloseSequenceType">
1704       <xs:sequence>
1705         <xs:element ref="wsrm:Identifier"/>
1706         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1707 maxOccurs="unbounded"/>
1708       </xs:sequence>
1709       <xs:anyAttribute namespace="##other" processContents="lax"/>
1710     </xs:complexType>
1711     <xs:complexType name="CloseSequenceResponseType">
1712       <xs:sequence>
1713         <xs:element ref="wsrm:Identifier"/>
1714         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1715 maxOccurs="unbounded"/>
1716       </xs:sequence>
1717       <xs:anyAttribute namespace="##other" processContents="lax"/>
1718     </xs:complexType>
1719     <xs:complexType name="TerminateSequenceType">
1720       <xs:sequence>
1721         <xs:element ref="wsrm:Identifier"/>
1722         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1723 maxOccurs="unbounded"/>
1724       </xs:sequence>
1725       <xs:anyAttribute namespace="##other" processContents="lax"/>

```

```

1726 </xs:complexType>
1727 <xs:complexType name="TerminateSequenceResponseType">
1728   <xs:sequence>
1729     <xs:element ref="wsrm:Identifier"/>
1730     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1731 maxOccurs="unbounded"/>
1732   </xs:sequence>
1733   <xs:anyAttribute namespace="##other" processContents="lax"/>
1734 </xs:complexType>
1735 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1736 <xs:complexType name="OfferType">
1737   <xs:sequence>
1738     <xs:element ref="wsrm:Identifier"/>
1739     <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1740     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1741     <xs:element name="IncompleteSequenceBehavior"
1742 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1743     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1744 maxOccurs="unbounded"/>
1745   </xs:sequence>
1746   <xs:anyAttribute namespace="##other" processContents="lax"/>
1747 </xs:complexType>
1748 <xs:complexType name="AcceptType">
1749   <xs:sequence>
1750     <xs:element ref="wsrm:AcksTo"/>
1751     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1752 maxOccurs="unbounded"/>
1753   </xs:sequence>
1754   <xs:anyAttribute namespace="##other" processContents="lax"/>
1755 </xs:complexType>
1756 <xs:element name="Expires">
1757   <xs:complexType>
1758     <xs:simpleContent>
1759       <xs:extension base="xs:duration">
1760         <xs:anyAttribute namespace="##other" processContents="lax"/>
1761       </xs:extension>
1762     </xs:simpleContent>
1763   </xs:complexType>
1764 </xs:element>
1765 <xs:simpleType name="IncompleteSequenceBehaviorType">
1766   <xs:restriction base="xs:string">
1767     <xs:enumeration value="DiscardEntireSequence"/>
1768     <xs:enumeration value="DiscardFollowingFirstGap"/>
1769     <xs:enumeration value="NoDiscard"/>
1770   </xs:restriction>
1771 </xs:simpleType>
1772 <xs:element name="UsesSequenceSTR">
1773   <xs:sequence/>
1774   <xs:anyAttribute namespace="##other" processContents="lax"/>
1775 </xs:element>
1776 <xs:element name="UsesSequenceSSL">
1777   <xs:sequence/>
1778   <xs:anyAttribute namespace="##other" processContents="lax"/>
1779 </xs:element>
1780 <xs:element name="UnsupportedElement">
1781   <xs:simpleType>
1782     <xs:restriction base="xs:QName"/>
1783   </xs:simpleType>
1784 </xs:element>
1785 </xs:schema>

```

1786 Appendix B. WSDL

1787 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1788 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

1789 The following non-normative copy is provided for reference.

```
1790 <?xml version="1.0" encoding="utf-8"?>
1791 <!--
1792 OASIS takes no position regarding the validity or scope of any intellectual
1793 property or other rights that might be claimed to pertain to the
1794 implementation or use of the technology described in this document or the
1795 extent to which any license under such rights might or might not be available;
1796 neither does it represent that it has made any effort to identify any such
1797 rights. Information on OASIS's procedures with respect to rights in OASIS
1798 specifications can be found at the OASIS website. Copies of claims of rights
1799 made available for publication and any assurances of licenses to be made
1800 available, or the result of an attempt made to obtain a general license or
1801 permission for the use of such proprietary rights by implementors or users of
1802 this specification, can be obtained from the OASIS Executive Director.
1803 OASIS invites any interested party to bring to its attention any copyrights,
1804 patents or patent applications, or other proprietary rights which may cover
1805 technology that may be required to implement this specification. Please
1806 address the information to the OASIS Executive Director.
1807 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
1808 This document and translations of it may be copied and furnished to others,
1809 and derivative works that comment on or otherwise explain it or assist in its
1810 implementation may be prepared, copied, published and distributed, in whole or
1811 in part, without restriction of any kind, provided that the above copyright
1812 notice and this paragraph are included on all such copies and derivative
1813 works. However, this document itself does not be modified in any way, such as
1814 by removing the copyright notice or references to OASIS, except as needed for
1815 the purpose of developing OASIS specifications, in which case the procedures
1816 for copyrights defined in the OASIS Intellectual Property Rights document must
1817 be followed, or as required to translate it into languages other than English.
1818 The limited permissions granted above are perpetual and will not be revoked by
1819 OASIS or its successors or assigns.
1820 This document and the information contained herein is provided on an "AS IS"
1821 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1822 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1823 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1824 FOR A PARTICULAR PURPOSE.
1825 -->
1826 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1827 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1828 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
1829 open.org/ws-rx/wsrn/200608" xmlns:tns="http://docs.oasis-open.org/ws-
1830 rx/wsrn/200608/wsd" targetNamespace="http://docs.oasis-open.org/ws-
1831 rx/wsrn/200608/wsd">
1832   <wsdl:types>
1833     <xs:schema
1834       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1835       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-
1836       200608.xsd"/>
1837     </xs:schema>
1838   </wsdl:types>
1839   <wsdl:message name="CreateSequence">
1840     <wsdl:part name="create" element="rm:CreateSequence"/>
```

```

1841     </wsdl:message>
1842     <wsdl:message name="CreateSequenceResponse">
1843         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1844     </wsdl:message>
1845     <wsdl:message name="CloseSequence">
1846         <wsdl:part name="close" element="rm:CloseSequence"/>
1847     </wsdl:message>
1848     <wsdl:message name="CloseSequenceResponse">
1849         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1850     </wsdl:message>
1851     <wsdl:message name="TerminateSequence">
1852         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1853     </wsdl:message>
1854     <wsdl:message name="TerminateSequenceResponse">
1855         <wsdl:part name="terminateResponse"
1856 element="rm:TerminateSequenceResponse"/>
1857     </wsdl:message>
1858     <wsdl:message name="MakeConnection">
1859         <wsdl:part name="makeConnection" element="rm:MakeConnection"/>
1860     </wsdl:message>

1861     <wsdl:portType name="SequenceAbstractPortType">
1862         <wsdl:operation name="CreateSequence">
1863             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1864 open.org/ws-rx/wsrn/200608/CreateSequence"/>
1865             <wsdl:output message="tns:CreateSequenceResponse"
1866 wsaw:Action="http://docs.oasis-open.org/ws-
1867 rx/wsrn/200608/CreateSequenceResponse"/>
1868         </wsdl:operation>
1869         <wsdl:operation name="CloseSequence">
1870             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1871 open.org/ws-rx/wsrn/200608/CloseSequence"/>
1872             <wsdl:output message="tns:CloseSequenceResponse"
1873 wsaw:Action="http://docs.oasis-open.org/ws-
1874 rx/wsrn/200608/CloseSequenceResponse"/>
1875         </wsdl:operation>
1876         <wsdl:operation name="TerminateSequence">
1877             <wsdl:input message="tns:TerminateSequence"
1878 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence"/>
1879             <wsdl:output message="tns:TerminateSequenceResponse"
1880 wsaw:Action="http://docs.oasis-open.org/ws-
1881 rx/wsrn/200608/TerminateSequenceResponse"/>
1882         </wsdl:operation>
1883         <wsdl:operation name="MakeConnection">
1884             <wsdl:input message="tns:MakeConnection" wsaw:Action="http://docs.oasis-
1885 open.org/ws-rx/wsrn/200608/MakeConnection"/>
1886             <!-- As described in section 3.10, the MakeConnection operation
1887 establishes a connection. If a matching message is available then
1888 the backchannel of the connection will be used to carry the
1889 message. In SOAP terms the returned message is not a response,
1890 so there is no WSDL output message. -->
1891         </wsdl:operation>
1892     </wsdl:portType>
1893 </wsdl:definitions>

```

1894 Appendix C. Message Examples

1895 Appendix C.1 Create Sequence

1896 Create Sequence

```
1897 <?xml version="1.0" encoding="UTF-8"?>
1898 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1899 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1900 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1901   <S:Header>
1902     <wsa:MessageID>
1903       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1904     </wsa:MessageID>
1905     <wsa:To>http://example.com/serviceB/123</wsa:To>
1906     <wsa:Action>http://docs.oasis-open.org/ws-
1907 rx/wsmr/200608/CreateSequence</wsa:Action>
1908     <wsa:ReplyTo>
1909       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1910     </wsa:ReplyTo>
1911   </S:Header>
1912   <S:Body>
1913     <wsmr:CreateSequence>
1914       <wsmr:AcksTo>
1915         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1916       </wsmr:AcksTo>
1917     </wsmr:CreateSequence>
1918   </S:Body>
1919 </S:Envelope>
```

1920 Create Sequence Response

```
1921 <?xml version="1.0" encoding="UTF-8"?>
1922 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1923 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1924 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1925   <S:Header>
1926     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1927     <wsa:RelatesTo>
1928       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1929     </wsa:RelatesTo>
1930     <wsa:Action>
1931       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
1932     </wsa:Action>
1933   </S:Header>
1934   <S:Body>
1935     <wsmr:CreateSequenceResponse>
1936       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1937     </wsmr:CreateSequenceResponse>
1938   </S:Body>
1939 </S:Envelope>
```

1940 Appendix C.2 Initial Transmission

1941 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1942 figure. The three messages have the following headers; the third message is identified as the last
1943 message in the Sequence:

1944 Message 1

```

1945 <?xml version="1.0" encoding="UTF-8"?>
1946 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1947 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1948 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1949   <S:Header>
1950     <wsa:MessageID>
1951       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1952     </wsa:MessageID>
1953     <wsa:To>http://example.com/serviceB/123</wsa:To>
1954     <wsa:From>
1955       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1956     </wsa:From>
1957     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1958     <wsmr:Sequence>
1959       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1960       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1961     </wsmr:Sequence>
1962   </S:Header>
1963   <S:Body>
1964     <!-- Some Application Data -->
1965   </S:Body>
1966 </S:Envelope>

```

1967 Message 2

```

1968 <?xml version="1.0" encoding="UTF-8"?>
1969 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1970 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1971 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1972   <S:Header>
1973     <wsa:MessageID>
1974       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1975     </wsa:MessageID>
1976     <wsa:To>http://example.com/serviceB/123</wsa:To>
1977     <wsa:From>
1978       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1979     </wsa:From>
1980     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1981     <wsmr:Sequence>
1982       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1983       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1984     </wsmr:Sequence>
1985   </S:Header>
1986   <S:Body>
1987     <!-- Some Application Data -->
1988   </S:Body>
1989 </S:Envelope>

```

1990 Message 3

```

1991 <?xml version="1.0" encoding="UTF-8"?>
1992 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1993 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1994 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1995   <S:Header>
1996     <wsa:MessageID>
1997       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1998     </wsa:MessageID>
1999     <wsa:To>http://example.com/serviceB/123</wsa:To>
2000     <wsa:From>
2001       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2002     </wsa:From>
2003     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2004     <wsmr:Sequence>
2005       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>

```

```

2006     <wsrm:MessageNumber>3</wsrm:MessageNumber>
2007     </wsrm:Sequence>
2008     <wsrm:AckRequested>
2009         <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2010     </wsrm:AckRequested>
2011 </S:Header>
2012 <S:Body>
2013     <!-- Some Application Data -->
2014 </S:Body>
2015 </S:Envelope>

```

2016 **Appendix C.3 First Acknowledgement**

2017 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
2018 responds with an Acknowledgement for messages 1 and 3:

```

2019 <?xml version="1.0" encoding="UTF-8"?>
2020 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2021 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
2022 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2023   <S:Header>
2024     <wsa:MessageID>
2025       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
2026     </wsa:MessageID>
2027     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2028     <wsa:From>
2029       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2030     </wsa:From>
2031     <wsa:Action>
2032       http://docs.oasis-open.org/ws-rx/wsrm/200608/SequenceAcknowledgement
2033     </wsa:Action>
2034     <wsrm:SequenceAcknowledgement>
2035       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2036       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2037       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2038     </wsrm:SequenceAcknowledgement>
2039   </S:Header>
2040   <S:Body/>
2041 </S:Envelope>

```

2042 **Appendix C.4 Retransmission**

2043 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2044 requests an Acknowledgement:

```

2045 <?xml version="1.0" encoding="UTF-8"?>
2046 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2047 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
2048 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2049   <S:Header>
2050     <wsa:MessageID>
2051       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2052     </wsa:MessageID>
2053     <wsa:To>http://example.com/serviceB/123</wsa:To>
2054     <wsa:From>
2055       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2056     </wsa:From>
2057     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2058     <wsrm:Sequence>
2059       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2060       <wsrm:MessageNumber>2</wsrm:MessageNumber>
2061     </wsrm:Sequence>

```

```

2062     <wsrm:AckRequested>
2063     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2064     </wsrm:AckRequested>
2065   </S:Header>
2066   <S:Body>
2067     <!-- Some Application Data -->
2068   </S:Body>
2069 </S:Envelope>

```

2070 Appendix C.5 Termination

2071 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
 2072 be terminated:

```

2073 <?xml version="1.0" encoding="UTF-8"?>
2074 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2075 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
2076 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2077   <S:Header>
2078     <wsa:MessageID>
2079       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2080     </wsa:MessageID>
2081     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2082     <wsa:From>
2083       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2084     </wsa:From>
2085     <wsa:Action>
2086       http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
2087     </wsa:Action>
2088     <wsrm:SequenceAcknowledgement>
2089       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2090       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2091     </wsrm:SequenceAcknowledgement>
2092   </S:Header>
2093   <S:Body/>
2094 </S:Envelope>

```

2095 Terminate Sequence

```

2096 <?xml version="1.0" encoding="UTF-8"?>
2097 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2098 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
2099 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2100   <S:Header>
2101     <wsa:MessageID>
2102       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2103     </wsa:MessageID>
2104     <wsa:To>http://example.com/serviceB/123</wsa:To>
2105     <wsa:Action>
2106       http://docs.oasis-open.org/ws-rx/wsr/200608/TerminateSequence
2107     </wsa:Action>
2108     <wsa:From>
2109       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2110     </wsa:From>
2111   </S:Header>
2112   <S:Body>
2113     <wsrm:TerminateSequence>
2114       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2115     </wsrm:TerminateSequence>
2116   </S:Body>
2117 </S:Envelope>

```

2118 Terminate Sequence Response

```

2119 <?xml version="1.0" encoding="UTF-8"?>

```

```

2120 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2121 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2122 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2123   <S:Header>
2124     <wsa:MessageID>
2125       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2126     </wsa:MessageID>
2127     <wsa:To>http://example.com/serviceA/789</wsa:To>
2128     <wsa:Action>
2129       http://docs.oasis-open.org/ws-rx/wsmr/200608/TerminateSequenceResponse
2130     </wsa:Action>
2131     <wsa:RelatesTo>
2132       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2133     </wsa:RelatesTo>
2134     <wsa:From>
2135       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2136     </wsa:From>
2137   </S:Header>
2138   <S:Body>
2139     <wsmr:TerminateSequenceResponse>
2140       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
2141     </wsmr:TerminateSequenceResponse>
2142   </S:Body>
2143 </S:Envelope>

```

2144 Appendix C.6 MakeConnection

2145 To illustrate how a `MakeConnection` message exchange can be used to deliver messages to an
2146 Endpoint that is not addressable, consider the case of a pub/sub scenario in which the Endpoint to which
2147 notifications are to be delivered (the "event consumer") is not addressable by the notification sending
2148 Endpoint (the "event producer"). In this scenario the event consumer must initiate the connections in order
2149 for the notifications to be delivered. One possible set of message exchanges (using HTTP) that
2150 demonstrate how this can be achieved using `MakeConnection` is shown below.

2151 **Step 1** – During a "subscribe" operation, the event consumer's EPR specifies the RM anonymous URI
2152 and the RM Policy Assertion to indicate whether or not RM is required:

```

2153 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2154 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2155 xmlns:wsmrp="http://docs.oasis-open.org/ws-rx/wsmrp/200608"
2156 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2157   <S:Header>
2158     <wsa:To> http://example.org/subscriptionService </wsa:To>
2159     <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813</wsa:MessageID>
2160     <wsa:ReplyTo>
2161       <wsa:To> http://client456.org/response </wsa:To>
2162     </wsa:ReplyTo>
2163   </S:Header>
2164   <S:Body>
2165     <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">
2166       <!-- subscription service specific data -->
2167       <targetEPR>
2168         <wsa:Address>http://docs.oasis-open.org/ws-
2169 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>
2170       <wsa:Metadata>
2171         <wsp:Policy wsu:Id="MyPolicy">
2172           <wsmrp:RMAssertion/>
2173         </wsp:Policy>
2174       </wsa:Metadata>
2175     </targetEPR>
2176   </sub:Subscribe>
2177 </S:Body>

```

2178 </S:Envelope>

2179 In this example the `subscribe` and `targetEPR` elements are simply examples of what a subscription
2180 request message might contain. Note: the `wsa:Address` element contains the RM anonymous URI
2181 indicating that the notification producer needs to queue the messages until they are requested using the
2182 `MakeConnection` message exchange. The EPR also contains the RM Policy Assertion indicating the RM
2183 must be used when notifications related to this subscription are sent.

2184 **Step 2** – Once the subscription is established, the event consumer checks for a pending message:

```
2185 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
2186 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"  
2187 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
2188   <S:Header>  
2189     <wsa:Action>http://docs.oasis-open.org/ws-  
2190 rx/wsmr/200608/MakeConnection</wsa:Action>  
2191     <wsa:To> http://example.org/subscriptionService </wsa:To>  
2192   </S:Header>  
2193   <S:Body>  
2194     <wsmr:MakeConnection>  
2195       <wsmr:Address>http://docs.oasis-open.org/ws-  
2196 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-  
2197 446655440000</wsmr:Address>  
2198     </wsmr:MakeConnection>  
2199   </S:Body>  
2200 </S:Envelope>
```

2201 **Step 3** – If there are messages waiting to be delivered then a message will be returned back to the event
2202 consumer. However, because WS-RM is being used to deliver the messages, the first message returned
2203 is a `CreateSequence`:

```
2204 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
2205 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"  
2206 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
2207   <S:Header>  
2208     <wsa:Action>http://docs.oasis-open.org/ws-  
2209 rx/wsmr/200608/CreateSequence</wsa:Action>  
2210     <wsa:To>http://docs.oasis-open.org/ws-  
2211 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>  
2212     <wsa:ReplyTo> http://example.org/subscriptionService </wsa:ReplyTo>  
2213     <wsa:MessageID> http://example.org/id-123-456 </wsa:MessageID>  
2214   </S:Header>  
2215   <S:Body>  
2216     <wsmr:CreateSequence>  
2217       <wsmr:AcksTo>  
2218         <wsa:Address> http://example.org/subscriptionService </wsa:Address>  
2219       </wsmr:AcksTo>  
2220     </wsmr:CreateSequence>  
2221   </S:Body>  
2222 </S:Envelope>
```

2223 Notice from the perspective of how the RM Source on the event producer interacts with the RM
2224 Destination of those messages, nothing new is introduced by the use of the `MakeConnection`, the use
2225 of RM protocol is the same as the case where the event consumer is addressable.

2226 **Step 4** – The event consumer will respond with a `CreateSequenceResponse` message per normal WS-
2227 Addressing rules:

```
2228 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
```

```

2229 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2230 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2231   <S:Header>
2232     <wsa:Action>http://docs.oasis-open.org/ws-
2233 rx/wsmr/200608/CreateSequenceResponse</wsa:Action>
2234     <wsa:To> http://example.org/subscriptionService </wsa:To>
2235     <wsa:RelatesTo> http://example.org/id-123-456 </wsa:RelatesTo>
2236   </S:Header>
2237   <S:Body>
2238     <wsmr:CreateSequenceResponse>
2239       <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>
2240     </wsmr:CreateSequenceResponse>
2241   </S:Body>
2242 </S:Envelope>

```

2243 Note, this message is carried on an HTTP request directed to the `wsa:ReplyTo` EPR, and the HTTP
2244 response will be an HTTP 202.

2245 **Step 5** – The event consumer checks for another message pending:

```

2246 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2247 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2248 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2249   <S:Header>
2250     <wsa:Action>http://docs.oasis-open.org/ws-
2251 rx/wsmr/200608/MakeConnection</wsa:Action>
2252     <wsa:To> http://example.org/subscriptionService </wsa:To>
2253   </S:Header>
2254   <S:Body>
2255     <wsmr:MakeConnection>
2256       <wsmr:Address>http://docs.oasis-open.org/ws-
2257 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-
2258 446655440000</wsmr:Address>
2259     </wsmr:MakeConnection>
2260   </S:Body>
2261 </S:Envelope>

```

2262 Notice this is the same message as the one sent in step 2.

2263 **Step 6** – If there is a message pending for this destination then it is returned on the HTTP response:

```

2264 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2265 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2266 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2267   <S:Header>
2268     <wsa:Action> http://example.org/eventType1 </wsa:Action>
2269     <wsa:To>http://docs.oasis-open.org/ws-
2270 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
2271     <wsmr:Sequence>
2272       <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>
2273     </wsmr:Sequence>
2274     <wsmr:MessagePending pending="true"/>
2275   </S:Header>
2276   <S:Body>
2277     <!-- event specific data -->
2278   </S:Body>
2279 </S:Envelope>

```

2280 As noted in step 3, the use of the RM protocol does not change when using `MakeConnection`. The
2281 format of the messages, the order of the messages sent and the timing of when to send it remains the
2282 same.

2283 **Step 7** – At some later interval, or immediately due to the `MessagePending` header's "pending"
2284 attribute being set to "true", the event consumer will poll again:

```
2285 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"  
2286 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200608"  
2287 xmlns:wsa="http://www.w3.org/2005/08/addressing">  
2288   <S:Header>  
2289     <wsa:Action>http://docs.oasis-open.org/ws-  
2290 rx/wsm/200608/MakeConnection</wsa:Action>  
2291     <wsa:To> http://example.org/subscriptionService </wsa:To>  
2292   </S:Header>  
2293   <S:Body>  
2294     <wsm:MakeConnection>  
2295       <wsm:Address>http://docs.oasis-open.org/ws-  
2296 rx/wsm/200608/anonymous?id=550e8400-e29b-11d4-a716-  
2297 446655440000</wsm:Address>  
2298     </wsm:MakeConnection>  
2299   </S:Body>  
2300 </S:Envelope>
```

2301 Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to
2302 the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event
2303 producer to send not only application messages but RM protocol messages (e.g. `CloseSequence`,
2304 `TerminateSequence` or even additional `CreateSequences`) as needed.

2305 **Step 8** – If at any point in time there are no messages pending, in response to a `MakeConnection` the
2306 event producer returns an HTTP 202 back to the event consumer. The process then repeats (back to step
2307 7) until the subscription ends.

2308 Appendix D. State Tables

2309 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2310 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2311 Legend:

2312 The first column of these tables contains the motivating event and has the following format:

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

2313 Where:

- 2314 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2315 described by the specification.
- 2316 ● [source]: indicates the source of the event; one of:
 - 2317 ● [msg] a Received message
 - 2318 ● [int]: an internal event such as the firing of a timer
 - 2319 ● [app]: the application
 - 2320 ● [unspec]: the source is unspecified

2321 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}

2322 Where:

- 2323 ● action to take: indicates that the state machine performs the following action. Actions surrounded
2324 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2325 "Transmit"
 - 2326 ● [next state]: indicates the state to which the state machine will advance upon the performance of
2327 the action. For ease of reading the next state "same" indicates that the state does not change.
 - 2328 ● {ref} is a reference to the document section describing the behavior in this cell
- 2329 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2330 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2331 described in this specification and does not indicate normal protocol operation. Implementations MAY
2332 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2333 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2334 combinations.

2335 Table 1 RM Source Sequence State Transition Table

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

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
Sequence Closed Fault [msg] {4.7}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]
Unknown Sequence Fault [msg] {4.3}			Terminate Sequence [None] {4.3}			
Sequence Terminated Fault [msg] {4.2}	N/A		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 Acknowledgement [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}			

2336 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}

Events	Sequence States		
	None	Created	Closed
<AckRequested> [msg] {3.8}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.8}	Xmit SeqAck+Final [Same] {3.9}
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Xmit 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}

2337 The following two tables apply only if the `MakeConnection` mechanism is utilized.

2338 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)]

2339 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)

2340 **Appendix E. Acknowledgments**

2341 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2342 authors:

2343 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),
2344 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),
2345 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),
2346 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don
2347 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),
2348 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony
2349 Storey(IBM).

2350 The following individuals have provided invaluable input into the initial contribution:

2351 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen Brown
2352 (Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM), Paul
2353 Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott
2354 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal
2355 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter
2356 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish
2357 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2358 The following individuals were members of the committee during the development of this specification:

2359 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben Malek
2360 (Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd Burch
2361 (Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul Cotton
2362 (Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques Durand
2363 (Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2), Robert
2364 Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair Green(Choreology),
2365 Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu), Chamikara Jayalath
2366 (WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul Knight(Nortel), Dan
2367 Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt Lovett(IBM), Ashok Malhotra
2368 (Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff Mischkinsky(Oracle), Nilo Mitra
2369 (Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku Nishiyama(Hitachi), Dave Orchard
2370 (BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert Pilz(BEA), Martin Raepple(SAP),
2371 Eric Rajkovic(Oracle), Stefan Rossmann(SAP), Tom Rutt(Fujitsu), Rich Salz(IBM), Shivajee
2372 Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von Riegen(SAP), Pete Wenzel(Sun), Steve
2373 Winkler(SAP), Ümit Yalçınalp(SAP), Nobuyuki Yamamoto(Hitachi).

Appendix F. Revision History

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

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

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

Rev	Date	By Whom	What
wd-11	2006-03-08	Doug Davis	Issue 100 applied
wd-12	2006-03-20	Doug Davis	Added space in "SOAP1.x" – PaulCotton
wd-12	2006-04-11	Doug Davis	Issue 007 applied
wd-12	2006-04-11	Doug Davis	Issue 090 applied
wd-12	2006-04-11	Doug Davis	Issue 098 applied
wd-12	2006-04-11	Doug Davis	Issue 099 applied
wd-12	2006-04-11	Doug Davis	Issue 101 applied
wd-12	2006-04-11	Doug Davis	Issue 103 applied
wd-12	2006-04-11	Doug Davis	Issue 104 applied
wd-12	2006-04-11	Doug Davis	Issue 105 applied
wd-12	2006-04-11	Doug Davis	Issue 107 applied
wd-12	2006-04-11	Doug Davis	Issue 109 applied
wd-12	2006-04-11	Doug Davis	Issue 110 applied
wd-12	2006-04-12	Doug Davis	Used "generated" instead of "issue" or "send" when talking about faults.
wd-12	2006-04-24	Gilbert Pilz	Update references to WS-Addressing to the Proposed Recommendations; update WS-RM namespace to "200604".
wd-13	2006-05-08	Gilbert Pilz	i093 part 1; more work needed
wd-13	2006-05-10	Doug Davis	Issue 096 applied
wd-13	2006-05-26	Gilbert Pilz	i093 part 2; reflects decisions from 2006-05-25 meeting
wd-13	2006-05-28	Gilbert Pilz	Issue 106 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 118 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 120 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 114 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 116 applied
wd-14	2006-06-05	Gilbert Pilz	Accept all changes; bump WD number
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Marc Goodner
wd-14	2006-06-07	Doug Davis	Change a couple of period/sp/sp to period/sp
wd-14	2006-06-07	Doug Davis	Added a space in "URI]of" – per Marc Goodner
wd-14	2006-06-07	Doug Davis	Issue 131 applied
wd-14	2006-06-07	Doug Davis	Issue 132 applied
wd-14	2006-06-07	Doug Davis	Issue 119 applied
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Doug Davis
wd-14	2006-06-07	Doug Davis	s/"none"/"full-uri"/ - per Marc Goodner
wd-14	2006-06-12	Doug Davis	Complete i106
wd-14	2006-06-12	Doug Davis	Issues 089 applied
wd-14	2006-06-12	Doug Davis	Fix for several RFC2119 keywords – per Anish
wd-15	2006-06-12	Doug Davis	Accept all changed, dump WD number
wd-15	2006-06-12	Doug Davis	Move WSDL after Schema

Rev	Date	By Whom	What
wd-15	2006-06-12	Doug Davis	Nits – remove tabs, extra [yyy]'s ...
wd-15	2006-06-14	Doug Davis	Remove extra "OPTIONAL"s – Matt Lovett
wd-15	2006-06-14	Doug Davis	Remove blank rows/columns from state table. Fix italics in state table
wd-15	2006-06-15	Doug Davis	Typo – section D was empty
wd-15	2006-06-16	Doug Davis	Issue 125 applied
wd-15	2006-06-16	Doug Davis	Issue 126 applied
wd-15	2006-06-16	Doug Davis	Issue 127 applied
wd-15	2006-06-16	Doug Davis	Issue 133 applied
wd-15	2006-06-16	Doug Davis	Issue 136 applied
wd-15	2006-06-16	Doug Davis	Issue 138 applied
wd-15	2006-06-16	Doug Davis	Issue 135 applied
wd-15	2006-06-20	Doug Davis	Added all TC members to the ack list
wd-15	2006-06-22	Doug Davis	Issue 129 applied
wd-15	2006-06-22	Doug Davis	Issue 130 applied
wd-15	2006-06-22	Doug Davis	Issue 137 applied
wd-15	2006-06-26	Doug Davis	Issue 111 applied
wd-15	2006-06-26	Doug Davis	Missed a part of issue 129
wd-15	2006-06-30	Doug Davis	Fixed a typo in schema
wd-15	2006-06-30	Doug Davis	Issue 141 applied
wd-15	2006-06-30	Doug Davis	Issue 142 applied
wd-15	2006-06-30	Doug Davis	Issue 148 applied
wd-15	2006-06-30	Doug Davis	Issue 149 applied
wd-15	2006-06-30	Doug Davis	Issue 150 applied
wd-15	2006-07-06	Doug Davis	Issue 121 applied
wd-15	2006-07-21	Doug Davis	Issue 139 applied
wd-15	2006-07-21	Doug Davis	Issue 144 applied
wd-15	2006-07-21	Doug Davis	Issue 147 applied
wd-15	2006-07-21	Doug Davis	Issues 122-124 applied
wd-15	2006-07-27	Doug Davis	Updated list of oasis TC members (i134)
wd-15	2006-07-27	Doug Davis	Issue 140 applied
wd-15	2006-07-27	Doug Davis	Issue 145 applied
wd-15	2006-07-27	Doug Davis	Issue 143 applied
wd-15	2006-07-28	Doug Davis	Lots of minor typos found by Matt L.
wd-15	2006-07-28	Doug Davis	Issue 113 applied
wd-15	2006-08-04	Doug Davis	Update old namespaces – found by PaulC
wd-15	2006-08-04	Doug Davis	Issue 150 applied
wd-15	2006-08-04	Doug Davis	Minor typos – found by PeterN
wd-15	2006-08-04	Doug Davis	Verify all [refs]
wd-15	2006-08-04	Doug Davis	Change namespace to 2006/08
wd-15	2006-08-04	Doug Davis	Issue 148 applied

Rev	Date	By Whom	What
wd-15	2006-08-07	Doug Davis	Add some new glossary terms – per GilP
cd-04	2006-08-10	Gilbert Pilz	Formatting changes for better HTML rendering.
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.
wd-16	2006-10-25	Doug Davis	Accept all changes, update to wd16
wd-16	2006-10-26	Doug Davis	PR002 applied
wd-16	2006-10-26	Doug Davis	PR003 applied
wd-16	2006-10-26	Doug Davis	PR004 applied
wd-16	2006-10-27	Doug Davis	PR005 applied
wd-16	2006-10-27	Doug Davis	PR006 applied
wd-16	2006-10-27	Doug Davis	PR024 applied

2375 **Appendix G. Notices**

2376 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2377 might be claimed to pertain to the implementation or use of the technology described in this document or
2378 the extent to which any license under such rights might or might not be available; neither does it represent
2379 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
2380 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
2381 available for publication and any assurances of licenses to be made available, or the result of an attempt
2382 made to obtain a general license or permission for the use of such proprietary rights by implementors or
2383 users of this specification, can be obtained from the OASIS Executive Director.

2384 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
2385 other proprietary rights which may cover technology that may be required to implement this specification.
2386 Please address the information to the OASIS Executive Director.

2387 Copyright (C) OASIS Open (2006). All Rights Reserved.

2388 This document and translations of it may be copied and furnished to others, and derivative works that
2389 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
2390 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
2391 this paragraph are included on all such copies and derivative works. However, this document itself may
2392 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
2393 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
2394 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
2395 into languages other than English.

2396 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
2397 or assigns.

2398 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2399 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2400 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
2401 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.