



# 1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

3 Working Draft 16, November 20, 2006

## 4 Document identifier:

5 wsrn-1.1-spec-wd-16

## 6 Location:

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

## 8 Editors:

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

## 14 Contributors:

15 See the Acknowledgments (Appendix E).

## 16 Abstract:

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

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

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

## 31 Status:

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

## 42 Table of Contents

43	1 Introduction.....	4
44	1.1 Notational Conventions.....	4
45	1.2 Namespace.....	5
46	1.3 Compliance.....	5
47	2 Reliable Messaging Model.....	6
48	2.1 Glossary.....	6
49	2.2 Protocol Preconditions.....	7
50	2.3 Protocol Invariants.....	7
51	2.4 Example Message Exchange.....	8
52	3 RM Protocol Elements.....	10
53	3.1 Considerations on the Use of Extensibility Points.....	10
54	3.2 Considerations on the Use of "Piggy-Backing".....	10
55	3.3 Composition with WS-Addressing.....	10
56	3.4 Sequence Creation.....	10
57	3.5 Closing A Sequence.....	15
58	3.6 Sequence Termination.....	16
59	3.7 Sequences.....	18
60	3.8 Request Acknowledgement.....	19
61	3.9 Sequence Acknowledgement.....	20
62	3.10 MakeConnection.....	22
63	3.11 MessagePending.....	24
64	4 Faults.....	26
65	4.1 SequenceFault Element.....	27
66	4.2 Sequence Terminated.....	28
67	4.3 Unknown Sequence.....	28
68	4.4 Invalid Acknowledgement.....	29
69	4.5 Message Number Rollover.....	29
70	4.6 Create Sequence Refused.....	30
71	4.7 Sequence Closed.....	30
72	4.8 WSRM Required.....	31
73	4.9 Unsupported Selection.....	31
74	5 Security Threats and Countermeasures.....	33
75	5.1 Threats and Countermeasures.....	33
76	5.1.1 Integrity Threats.....	33
77	5.1.1.1 Countermeasures.....	33
78	5.1.2 Resource Consumption Threats.....	34
79	5.1.2.1 Countermeasures.....	34

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

# 109 **1 Introduction**

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

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

## 119 **1.1 Notational Conventions**

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

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

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

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

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

147 **1.2 Namespace**

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

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

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

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

154 Table 1

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

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

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

158 **1.3 Conformance**

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

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

## 2 Reliable Messaging Model

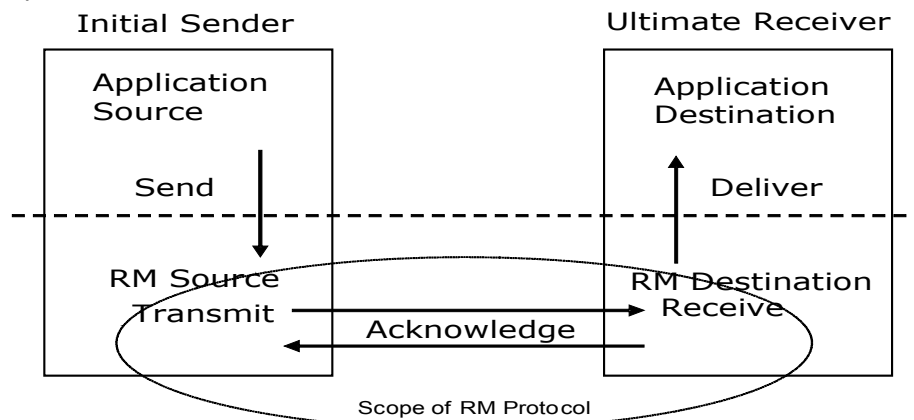
165

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

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

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

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



191 Figure 1: Reliable Messaging Model

### 2.1 Glossary

192

193 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 222 **2.2 Protocol Preconditions**

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

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

## 232 2.3 Protocol Invariants

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

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

## 244 2.4 Example Message Exchange

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

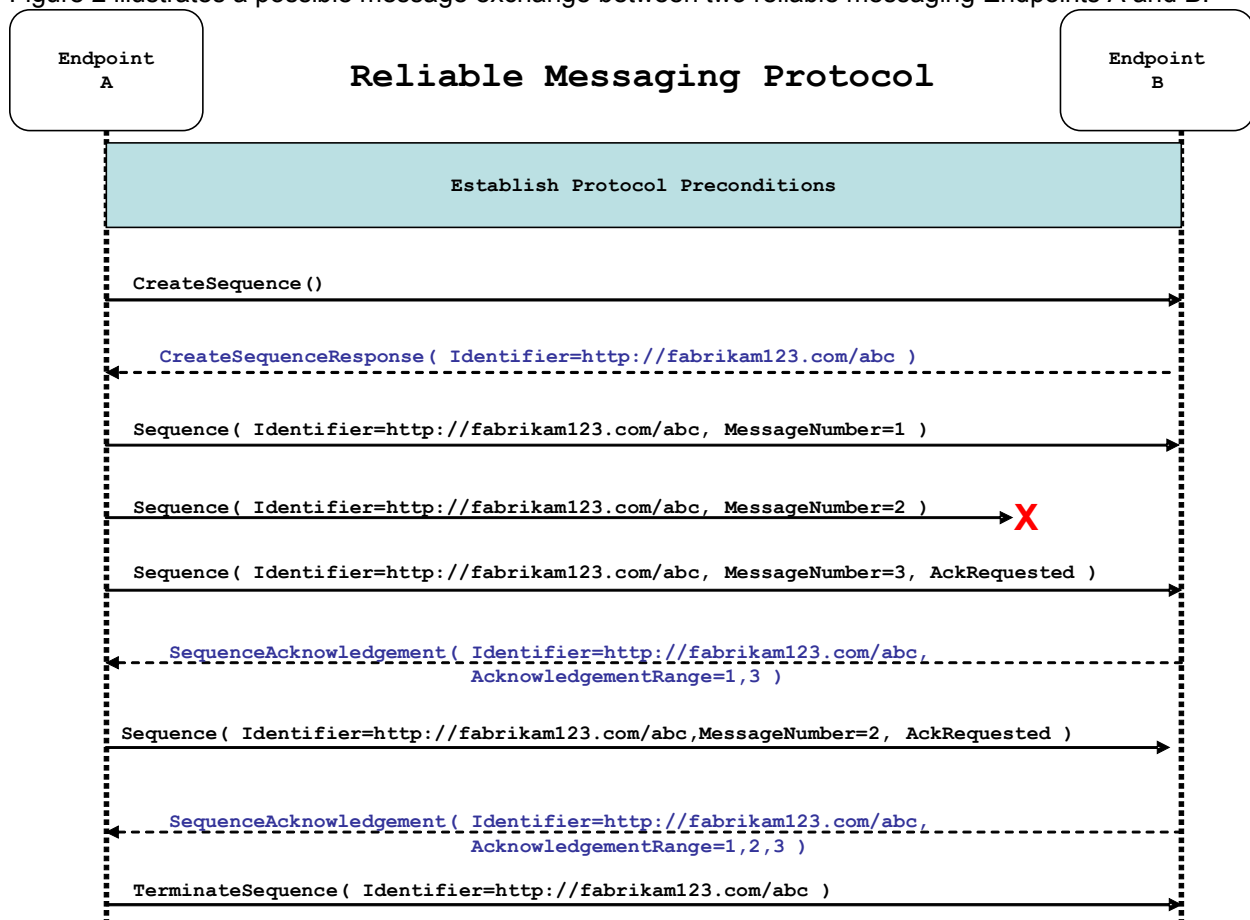


Figure 2: The WS-ReliableMessaging Protocol

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



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

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

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

## 285 3 RM Protocol Elements

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

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

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

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

294 Some RM [Protocol Header Blocks](#) may be added to messages that are targeted to the same Endpoint  
295 to which those headers are to be sent (a concept often referred to as "piggy-backing"), thus saving the  
296 overhead of an additional message exchange. Reference parameters MUST be considered when  
297 determining whether two EPRs are targeted to the same Endpoint. [The determination of if and when a  
298 Header Block will be piggy-backed onto another message is made by the entity \(RMS or RMD\) that is  
299 sending the header.](#) See the sections that define each RM header block to know which ones may be  
300 considered for piggy-backing.

### 301 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

### 319 3.4 Sequence Creation

320 The RM Source MUST request creation of an outbound Sequence by sending a `CreateSequence`  
321 element in the body of a message to the RM Destination which in turn responds either with a message  
322 containing `CreateSequenceResponse` or a `CreateSequenceRefused` fault. The RM Source MAY

323 include an offer to create an inbound Sequence within the `CreateSequence` message. This offer is  
324 either accepted or rejected by the RM Destination in the `CreateSequenceResponse` message.

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

327 The following exemplar defines the `CreateSequence` syntax:

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

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

343 `/wsrm:CreateSequence`

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

348 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

358 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

366 `/wsrm:CreateSequence/wsrm:Offer`

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

369 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier

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

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

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

375 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint

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

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

387 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires

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

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

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

394 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior

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

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

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

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

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

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

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

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

411 /wsmr:CreateSequence/{any}

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

414 /wsmr:CreateSequence/@{any}

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

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

421 The following exemplar defines the `CreateSequenceResponse` syntax:

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

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

435 /wsmr:CreateSequenceResponse

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

439 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

446 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

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

458 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

469 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

475 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

490 `/wsrm:CreateSequenceResponse/{any}`

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

493 /wsmr:CreateSequenceResponse/@{any}

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

### 496 **3.5 Closing A Sequence**

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

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

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

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

519 The following exemplar defines the `CloseSequence` syntax:

```
520 <wsmr:CloseSequence ...>  
521   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
522   ...  
523 </wsmr:CloseSequence>
```

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

525 /wsmr:CloseSequence

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

528 /wsmr:CloseSequence/wsmr:Identifier

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

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

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

535 /wsmr:CloseSequence/{any}

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

538 /wsmr:CloseSequence@{any}

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

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

544 The following exemplar defines the `CloseSequenceResponse` syntax:

```
545 <wsmr:CloseSequenceResponse ...>  
546   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
547   ...  
548 </wsmr:CloseSequenceResponse>
```

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

550 /wsmr:CloseSequenceResponse

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

553 /wsmr:CloseSequenceResponse/wsmr:Identifier

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

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

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

560 /wsmr:CloseSequenceResponse/{any}

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

563 /wsmr:CloseSequenceResponse@{any}

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

## 566 3.6 Sequence Termination

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

574 The following exemplar defines the `TerminateSequence` syntax:



```

575 <wsrm:TerminateSequence ...>
576   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
577   ...
578 </wsrm:TerminateSequence>

```

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

580 `/wsrm:TerminateSequence`

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

586 `/wsrm:TerminateSequence/wsrm:Identifier`

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

590 `/wsrm:TerminateSequence/wsrm:Identifier/@{any}`

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

593 `/wsrm:TerminateSequence/{any}`

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

596 `/wsrm:TerminateSequence/@{any}`

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

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

602 The following exemplar defines the `TerminateSequenceResponse` syntax:

```

603 <wsrm:TerminateSequenceResponse ...>
604   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
605   ...
606 </wsrm:TerminateSequenceResponse>

```

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

608 `/wsrm:TerminateSequenceResponse`

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

612 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

619 `/wsmr:TerminateSequenceResponse/{any}`

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

622 `/wsmr:TerminateSequenceResponse/@{any}`

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

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

### 628 **3.7 Sequences**

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

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

636 A following exemplar defines its syntax:

```
637 <wsmr:Sequence ...>  
638   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
639   <wsmr:MessageNumber> wsmr:MessageNumberType </wsmr:MessageNumber>  
640   ...  
641 </wsmr:Sequence>
```

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

643 `/wsmr:Sequence`

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

650 `/wsmr:Sequence/wsmr:Identifier`

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

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

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

657 `/wsmr:Sequence/wsmr:MessageNumber`

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

662 `/wsrm:Sequence/{any}`

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

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

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

662 The following example illustrates a Sequence header block.

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

### 662 3.8 Request Acknowledgement

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

662 The RM Source MAY request an Acknowledgement Message from the RM Destination at any time by  
663 independently transmitting an `AckRequested` header block (i.e. as a header of a SOAP envelope with an  
664 empty body). Alternately the RM Source independently or it MAY include an `AckRequested` header block  
665 in any message targeted to the RM Destination. The RM Destination MUST detect and process any  
666 `AckRequested` header blocks that are piggy-backed on another message. If a non-mustUnderstand fault  
667 occurs when processing an `AckRequested` header block that was piggy-backed, a fault MUST be  
668 generated, but the processing of the original message MUST NOT be affected.

662 An RM Destination that Receives a message that contains an `AckRequested` header block MUST send  
663 a message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference  
664 (see Section 3.4) for a known Sequence or else generate an `UnknownSequence` fault. ~~If a non-~~  
665 ~~mustUnderstand fault occurs when processing an RM header that was piggy-backed on another~~  
666 ~~message, a fault MUST be generated, but the processing of the original message MUST NOT be~~  
667 ~~affected.~~ It is RECOMMENDED that the RM Destination return a `AcknowledgementRange` or `None`  
668 element instead of a `Nack` element (see Section 3.9).

662 The following exemplar defines its syntax:

```
662 <wsrm:AckRequested ...>  
662   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
662   ...  
662 </wsrm:AckRequested>
```

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

664 `/wsrm:AckRequested`

665 This element requests an Acknowledgement for the identified Sequence.

666 `/wsrm:AckRequested/wsrm:Identifier`

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

670 `/wsmr:AckRequested/wsmr:Identifier/@{any}`

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

673 `/wsmr:AckRequested/{any}`

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

676 `/wsmr:AckRequested/@{any}`

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

### 679 **3.9 Sequence Acknowledgement**

680 The RM Destination informs the RM Source of successful message receipt using a  
681 `SequenceAcknowledgement` header block. [Acknowledgements can be explicitly requested using the](#)  
682 [AckRequested directive \(see Section 3.8\).](#)

683 The RM Destination MAY Transmit the `SequenceAcknowledgement` header block independently (*i.e. as*  
684 *a header of a SOAP envelope with an empty body*). [Alternatively, an RM Destination MAY include a](#)  
685 [SequenceAcknowledgement header block on any SOAP envelope targeted to the endpoint referenced](#)  
686 [by the AcksTo EPR, or it MAY include the SequenceAcknowledgement header block on any message](#)  
687 [targeted to the AcksTo EPR. The RM Source MUST detect and process any](#)  
688 [SequenceAcknowledgement header blocks that are piggy-backed on another](#)  
689 [message. Acknowledgements can be explicitly requested using the AckRequested directive \(see Section](#)  
690 [3.8\).](#) If a non-mustUnderstand fault occurs when processing an [SequenceAcknowledgement RM](#)  
691 [header that was piggy-backed on another message](#), a fault MUST be generated, but the processing of the  
692 original message MUST NOT be affected.

693 ~~A RM Destination MAY include a SequenceAcknowledgement header block on any SOAP envelope~~  
694 ~~targeted to the endpoint referenced by the AcksTo EPR.~~

695 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the  
696 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing  
697 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any  
698 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted  
699 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received  
700 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`  
701 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`  
702 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination  
703 SHOULD respond on the protocol binding-specific back-channel provided by the Received message  
704 containing the `AckRequested` header block.

705 The following exemplar defines its syntax:

```
706 <wsmr:SequenceAcknowledgement ...>  
707   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
708   [ [ [ <wsmr:AcknowledgementRange ...  
709     Upper="wsmr:MessageNumberType"
```

```

710         Lower="wsrm:MessageType"/> +
711         | <wsrm:None/> ]
712         <wsrm:Final/> ? ]
713         | <wsrm:Nack> wsrm:MessageType </wsrm:Nack> + ]
714
715         ...
716     </wsrm:SequenceAcknowledgement>

```

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

718 `/wsrm:SequenceAcknowledgement`

719 This element contains the Sequence Acknowledgement information.

720 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

729 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

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

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

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

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

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

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

744 `/wsrm:SequenceAcknowledgement/wsrm:None`

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

749 `/wsrm:SequenceAcknowledgement/wsrm:Final`

750 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. This  
751 element indicates that the RM Destination is not receiving new messages for the specified Sequence. The

752 RM Source can be assured that the ranges of messages acknowledged by this  
753 SequenceAcknowledgement header block will not change in the future. The RM Destination MUST  
754 include this element when the Sequence is closed. The RM Destination MUST NOT include this element  
755 when sending a Nack; it can only be used when sending AcknowledgementRange elements or a None.

756 /wsmr:SequenceAcknowledgement/wsmr:Nack

757 The RM Destination MAY include this element within a SequenceAcknowledgement header block. If  
758 used, the RM Destination MUST set the value of this element to a MessageNumberType representing  
759 the MessageNumber of an unreceived message in a Sequence. The RM Destination MUST NOT include  
760 a Nack element if a sibling AcknowledgementRange or None element is also present as a child of  
761 SequenceAcknowledgement. Upon the receipt of a Nack, an RM Source SHOULD retransmit the  
762 message identified by the Nack. The RM Destination MUST NOT issue a SequenceAcknowledgement  
763 containing a Nack for a message that it has previously acknowledged within a  
764 AcknowledgementRange. The RM Source SHOULD ignore a SequenceAcknowledgement containing  
765 a Nack for a message that has previously been acknowledged within a AcknowledgementRange.

766 /wsmr:SequenceAcknowledgement/{any}

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

769 /wsmr:SequenceAcknowledgement/@{any}

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

772 The following examples illustrate SequenceAcknowledgement elements:

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

```
774 <wsmr:SequenceAcknowledgement>  
775   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
776   <wsmr:AcknowledgementRange Upper="10" Lower="1"/>  
777 </wsmr:SequenceAcknowledgement>
```

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

```
780 <wsmr:SequenceAcknowledgement>  
781   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
782   <wsmr:AcknowledgementRange Upper="2" Lower="1"/>  
783   <wsmr:AcknowledgementRange Upper="6" Lower="4"/>  
784   <wsmr:AcknowledgementRange Upper="10" Lower="8"/>  
785 </wsmr:SequenceAcknowledgement>
```

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

```
787 <wsmr:SequenceAcknowledgement>  
788   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
789   <wsmr:Nack>3</wsmr:Nack>  
790 </wsmr:SequenceAcknowledgement>
```

### 791 3.10 MakeConnection

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

796

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

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

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

810 The following exemplar defines the `MakeConnection` syntax:

```
811 <wsrm:MakeConnection ...>  
812   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
813   <wsrm:Address ...> xs:anyURI </wsrm:Address> ?  
814   ...  
815 </wsrm:MakeConnection>
```

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

817 `/wsrm:MakeConnection`

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

820 `/wsrm:MakeConnection/wsrm:Identifier`

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

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

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

829 `/wsrm:MakeConnection/wsrm:Address`

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

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

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



840 /wsm:MakeConnection/{any}

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

845 /wsm:MakeConnection/@{any}

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

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

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

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

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

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

### 877 **3.11 MessagePending**

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



882 The following exemplar defines the `MessagePending` syntax:

```
883 <wsrm:MessagePending pending="xs:boolean" ...>  
884   ...  
885 </wsrm:MessagePending>
```

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

887 `/wsrm:MessagePending`

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

889 `/wsrm:MessagePending@pending`

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

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

892 `/wsrm:MessagePending/{any}`

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

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

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

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

## 900 4 Faults

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

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

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

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

914 The definitions of faults use the following properties:

915 [Code] The fault code.

916 [Subcode] The fault subcode.

917 [Reason] The English language reason element.

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

921 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or  
922 "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

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

```
924 <S:Envelope>
925   <S:Header>
926     <wsa:Action>
927       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault
928     </wsa:Action>
929     <!-- Headers elided for brevity. -->
930   </S:Header>
931   <S:Body>
932     <S:Fault>
933       <S:Code>
934         <S:Value> [Code] </S:Value>
935         <S:Subcode>
936           <S:Value> [Subcode] </S:Value>
937         </S:Subcode>
938       </S:Code>
939       <S:Reason>
940         <S:Text xml:lang="en"> [Reason] </S:Text>
941       </S:Reason>
942     <S:Detail>
```

```

943     [Detail]
944     ...
945     </S:Detail>
946     </S:Fault>
947     </S:Body>
948     </S:Envelope>

```

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

```

951 <S11:Envelope>
952   <S11:Header>
953     <wsrm:SequenceFault>
954       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
955       <wsrm:Detail> [Detail] </wsrm:Detail>
956       ...
957     </wsrm:SequenceFault>
958     <!-- Headers elided for brevity. -->
959   </S11:Header>
960   <S11:Body>
961     <S11:Fault>
962       <faultcode> [Code] </faultcode>
963       <faultstring> [Reason] </faultstring>
964     </S11:Fault>
965   </S11:Body>
966 </S11:Envelope>

```

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

```

969 <S11:Envelope>
970   <S11:Body>
971     <S11:Fault>
972       <faultcode> [Subcode] </faultcode>
973       <faultstring> [Reason] </faultstring>
974     </S11:Fault>
975   </S11:Body>
976 </S11:Envelope>

```

## 977 4.1 SequenceFault Element

978 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during  
979 the reliable messaging specific processing of a message belonging to a Sequence. WS-  
980 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP  
981 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in  
982 conjunction with the SOAP 1.2 binding.

983 The following exemplar defines its syntax:

```

984 <wsrm:SequenceFault ...>
985   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
986   <wsrm:Detail> ... </wsrm:Detail> ?
987   ...
988 </wsrm:SequenceFault>

```

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

990 `/wsrm:SequenceFault`

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

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

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

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

999 /wsm:SequenceFault/wsm:Detail/@{any}  
 1000 The application specific error information related to the fault being described.

1001 /wsm:SequenceFault/{any}  
 1002 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,  
 1003 to be passed.

1004 /wsm:SequenceFault/@{any}  
 1005 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the  
 1006 element.

1007 **4.2 Sequence Terminated**

1008 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding  
 1009 Endpoint of this decision.

1010 Properties:

1011 [Code] Sender or Receiver  
 1012 [Subcode] wsm:SequenceTerminated  
 1013 [Reason] The Sequence has been terminated due to an unrecoverable error.  
 1014 [Detail]

1015 `<wsm:Identifier ...> xs:anyURI </wsm:Identifier>`

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

1016 **4.3 Unknown Sequence**

1017 Properties:

1018 [Code] Sender  
 1019 [Subcode] wsm:UnknownSequence

1020 [Reason] The value of wsrn:Identifier is not a known Sequence identifier.

1021 [Detail]

1022 `<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.

#### 1023 **4.4 Invalid Acknowledgement**

1024 An example of when this fault is generated is when a message is Received by the RM Source containing  
1025 a SequenceAcknowledgement covering messages that have not been sent.

1026 [Code] Sender

1027 [Subcode] wsrn:InvalidAcknowledgement

1028 [Reason] The SequenceAcknowledgement violates the cumulative Acknowledgement invariant.

1029 [Detail]

1030 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

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

#### 1031 **4.5 Message Number Rollover**

1032 If the condition listed below is reached, the RM Destination MUST generate this fault.

1033 Properties:

1034 [Code] Sender

1035 [Subcode] wsrn:MessageNumberRollover

1036 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1037 [Detail]

```
1038 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1039 <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.

## 1040 4.6 Create Sequence Refused

1041 Properties:

1042 [Code] Sender or Receiver

1043 [Subcode] wsrm:CreateSequenceRefused

1044 [Reason] The Create Sequence request has been refused by the RM Destination.

1045 [Detail]

```
1046 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.

## 1047 4.7 Sequence Closed

1048 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1049 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that  
1050 is closed.

1051 Properties:

1052 [Code] Sender

1053 [Subcode] wsrm:SequenceClosed

1054 [Reason] The Sequence is closed and can not accept new messages.

1055 [Detail]

1056 `<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.

## 1057 4.8 WSRM Required

1058 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming  
1059 message that did not use this protocol.

1060 Properties:

1061 [Code] Sender

1062 [Subcode] wsrm:WSRMRequired

1063 [Reason] The RM Destination requires the use of WSRM.

1064 [Detail]

1065 `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.

## 1066 4.9 Unsupported Selection

1067 The QName of the unsupported element(s) are included in the detail.

1068 Properties:

1069 [Code] Receiver

1070 [Subcode] wsrm:UnsupportedSelection

1071 [Reason] The extension element used in the message selection is not supported by the RM Source

1072 [Detail]

1073 `<wsrm:UnsupportedElement> xs:QName </wsrm:UnsupportedElement>+`

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



## 1074 **5 Security Threats and Countermeasures**

1075 This specification considers two sets of security requirements, those of the applications that use the WS-  
1076 RM protocol and those of the protocol itself.

1077 This specification makes no assumptions about the security requirements of the applications that use WS-  
1078 RM. However, once those requirements have been satisfied within a given operational context, the  
1079 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;  
1080 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1081 There are many other security concerns that one may need to consider when implementing or using this  
1082 protocol. The material below should not be considered as a "check list". Implementers and users of this  
1083 protocol are urged to perform a security analysis to determine their particular threat profile and the  
1084 appropriate responses to those threats.

1085 Implementers are also advised that there is a core tension between security and reliable messaging that  
1086 can be problematic if not addressed by implementations; one aspect of security is to prevent message  
1087 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.  
1088 Consequently, if the security sub-system processes a message but a failure occurs before the reliable  
1089 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system  
1090 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-  
1091 system will likely continue to expect and even solicit the missing message(s). Care should be taken to  
1092 avoid and prevent this condition.

### 1093 **5.1 Threats and Countermeasures**

1094 The primary security requirement of this protocol is to protect the specified semantics and protocol  
1095 invariants against various threats. The following sections describe several threats to the integrity and  
1096 operation of this protocol and provide some general outlines of countermeasures to those threats.  
1097 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable  
1098 to all operational contexts.

#### 1099 **5.1.1 Integrity Threats**

1100 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic  
1101 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or  
1102 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block  
1103 to its intended message represents a threat to the WS-RM protocol.

1104 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM  
1105 Source and RM Destination then they have undermined the implementation's ability to guarantee the first  
1106 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be  
1107 Delivered to the Application Destination in the same order that they were sent by the Application Source.

##### 1108 **5.1.1.1 Countermeasures**

1109 Integrity threats are generally countered via the use of digital signatures some level of the communication  
1110 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include  
1111 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers  
1112 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which  
1113 they occur, implementations MUST allow for signatures that cover only these headers.

## 1114 **5.1.2 Resource Consumption Threats**

1115 The creation of a Sequence with an RM Destination consumes various resources on the systems used to  
1116 implement that RM Destination. These resources can include network connections, database tables,  
1117 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM  
1118 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM  
1119 Destination. Another attack is to create a Sequence for a service that is known to require in-order  
1120 message Delivery and use this Sequence to send a stream of very large messages to that service,  
1121 making sure to omit message number “1” from that stream.

### 1122 **5.1.2.1 Countermeasures**

1123 There are a number of countermeasures against the described resource consumption threats. The  
1124 technique advocated by this specification is for the RM Destination to restrict the ability to create a  
1125 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in  
1126 some cases, allows the identity of any attackers to be determined.

1127 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and  
1128 authenticate the RM Source that issued the `CreateSequence` message.

## 1129 **5.1.3 Sequence Spoofing Threats**

1130 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a  
1131 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a  
1132 fake `TerminateSequence` message that references the target Sequence and sends this message to the  
1133 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the  
1134 current `MessageNumber` for their target Sequence.

1135 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP  
1136 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier  
1137 to attack the Sequence. These attacks are “two-way” in that an attacker may choose to target the RM  
1138 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends  
1139 to the `AcksTo` EPR of an RM Source.

### 1140 **5.1.3.1 Sequence Hijacking**

1141 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject  
1142 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those  
1143 messages.

1144 Note that “sequence hijacking” should not be equated with “security session hijacking”. Although a  
1145 Sequence may be bound to some form of a security session in order to counter the threats described in  
1146 this section, applications MUST NOT rely on WS-RM-related information to make determinations about  
1147 the identity of the entity that created a message; applications SHOULD rely only upon information that is  
1148 established by the security infrastructure to make such determinations. Failure to observe this rule  
1149 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of  
1150 the ability to authenticate its peers even though the necessary security processing has taken place.

### 1151 **5.1.3.2 Countermeasures**

1152 There are a number of countermeasures against sequence spoofing threats. The technique advocated by  
1153 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1154 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that  
1155 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence  
1156 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that  
1157 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.  
1158 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a  
1159 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1160 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and  
1161 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its  
1162 sequence peer it MUST be able to identify and authenticate the entity that sent the  
1163 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a  
1164 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that  
1165 message and, if necessary, correlate this identity with the sequence peer identity established at sequence  
1166 creation time.

## 1167 **5.2 Security Solutions and Technologies**

1168 The security threats described in the previous sections are neither new nor unique. The solutions that  
1169 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This  
1170 section maps the facilities provided by common web services security solutions against countermeasures  
1171 described in the previous sections.

1172 Before continuing this discussion, however, some examination of the underlying requirements of the  
1173 previously described countermeasures is necessary. Specifically it should be noted that the technique  
1174 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates  
1175 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check  
1176 against this authenticated identity and determines if the RM Source is permitted to create Sequences with  
1177 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,  
1178 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of  
1179 such facilities is considered to be beyond the scope of this specification.

### 1180 **5.2.1 Transport Layer Security**

1181 This section describes how the facilities provided by SSL/TLS [RFC 4346] can be used to implement the  
1182 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints  
1183 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1184 The description provided here is general in nature and is not intended to serve as a complete definition on  
1185 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the  
1186 choice of features as well as the manner in which they will be used. The mechanisms described in the  
1187 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the  
1188 requirements and constraints of the use of SSL/TLS.

#### 1189 **5.2.1.1 Model**

1190 The basic model for using SSL/TLS is as follows:

- 1191 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1192 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM  
1193 Destination.

- 1194 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an  
1195 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a  
1196 synchronous `CreateSequenceResponse` using the session established in (1).
- 1197 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit  
1198 any and all messages or faults that refer to that Sequence.
- 1199 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established  
1200 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous  
1201 exchanges, the RM Destination uses the SSL/TLS session established in (1).

### 1202 5.2.1.2 Countermeasure Implementation

1203 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the  
1204 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the  
1205 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If  
1206 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and  
1207 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1208 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification  
1209 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods  
1210 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS  
1211 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1212 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP  
1213 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the  
1214 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party  
1215 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself  
1216 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.  
1217 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an  
1218 Acknowledgement) using BasicAuth.
- 1219 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the  
1220 connection authenticates itself to the party accepting the connection using an X.509 certificate  
1221 that is exchanged during the SSL/TLS handshake.

1222 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself  
1223 using one the above mechanisms. The authenticated identity can then be used to determine if the RM  
1224 Source is authorized to create a Sequence with the RM Destination.

1225 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1226 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the  
1227 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than  
1228 on authentication information. For example, an RM Destination can determine that a Sequence Traffic  
1229 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS  
1230 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a  
1231 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a  
1232 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used  
1233 to protect that Sequence.

1234 This specification does not preclude the use of other methods of using SSL/TLS to implement the  
1235 countermeasures (such as associating specific authentication information with a Sequence) although such  
1236 methods are not covered by this document.

1237 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS  
1238 session) are outside the scope of this specification.

## 1239 **5.2.2 SOAP Message Security**

1240 The mechanisms described in WS-Security may be used in various ways to implement the  
1241 countermeasures described in the previous sections. This specification advocates using the protocol  
1242 described by WS-SecureConversation [[SecureConversation](#)] (optionally in conjunction with WS-Trust  
1243 [[Trust](#)]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component  
1244 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1245 The description provided here is general in nature and is not intended to serve as a complete definition on  
1246 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations  
1247 need to agree on the choice of features as well as the manner in which they will be used. The  
1248 mechanisms described in the Web Services Security Policy Language MAY be used by services to  
1249 describe the requirements and constraints of the use of WS-SecureConversation.

### 1250 **5.2.2.1 Model**

1251 The basic model for using WS-SecureConversation is as follows:

- 1252 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This  
1253 may involve the participation of third parties such as a security token service. The tokens  
1254 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1255 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security  
1256 context that will be used to protect the Sequence. This is done so that, in cases where the  
1257 `CreateSequence` message is signed by more than one security context, the RM Source can  
1258 indicate which security context should be used to protect the newly created Sequence.
- 1259 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)  
1260 associated with the security context to sign (as defined by WS-Security) at least the body and any  
1261 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

### 1262 **5.2.2.2 Countermeasure Implementation**

1263 Without relying upon any authentication information, the per-message signatures provide the necessary  
1264 integrity qualities to counter the threats described in Section 5.1.1.

1265 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of  
1266 authentication claims must be provided by the RM Source to the RM Destination during the establishment  
1267 of the Security Context. These claims can then be used to determine if the RM Source is authorized to  
1268 create a Sequence with the RM Destination.

1269 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1270 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the  
1271 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security  
1272 context rather than on any authentication claims that may have been established during security context  
1273 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures  
1274 (such as associating specific authentication claims to a Sequence) are possible but not covered by this  
1275 document.

1276 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer  
1277 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1278 the association between a Sequence and its protecting security context cannot always be established  
1279 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and  
1280 `CreateSequenceResponse` messages may be signed by more than one security context.

1281 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as  
1282 amending or renewing contexts) are outside the scope of this specification.



## 1283 6 Securing Sequences

1284 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared  
1285 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of  
1286 achieving this objective depending upon the underlying security infrastructure.

### 1287 6.1 Securing Sequences Using WS-Security

1288 One mechanism for protecting a Sequence is to include a security token using a  
1289 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-  
1290 SecureConversation) in the `CreateSequence` element. This establishes an association between the  
1291 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source  
1292 and Destination MUST use the security token as the basis for authorization of all subsequent interactions  
1293 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as  
1294 there may be more than one token on a `CreateSequence` message or inferred from the communication  
1295 context (e.g. transport protection).

1296 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,  
1297 if the token being referenced supports such mechanism.

1298 The following exemplar defines the `CreateSequence` syntax when extended to include a  
1299 `wsse:SecurityTokenReference`:

```
1300 <wsrm:CreateSequence ...>  
1301   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1302   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1303   <wsrm:Offer ...>  
1304     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1305     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1306     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1307     <wsrm:IncompleteSequenceBehavior>  
1308       wsrml:IncompleteSequenceBehaviorType  
1309     </wsrm:IncompleteSequenceBehavior> ?  
1310     ...  
1311   </wsrm:Offer> ?  
1312   ...  
1313   <wsse:SecurityTokenReference>  
1314     ...  
1315   </wsse:SecurityTokenReference> ?  
1316   ...  
1317 </wsrm:CreateSequence>
```

1318 The following describes the content model of the additional `CreateSequence` elements.

1319 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1320 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in  
1321 section 3.4) to communicate an explicit reference to the security token, using a  
1322 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and Destination  
1323 MUST use to authorize messages for the created (and, if present, the offered) Sequence(s). All  
1324 subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST  
1325 demonstrate proof-of-possession of the secret associated with the token (e.g., by using or deriving from a  
1326 private or secret key).

1327 When a RM Source transmits a `CreateSequence` that has been extended to include a  
1328 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and  
1329 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include

1330 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This  
1331 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source  
1332 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands  
1333 and conforms with the requirements listed above. Note that an RM Destination understanding this header  
1334 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined  
1335 in WS-Security still applies.

1336 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1337 <wsm:UsesSequenceSTR ... />
```

1338 The following describes the content model of the `UsesSequenceSTR` header block.

1339 `/wsm:UsesSequenceSTR`

1340 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the  
1341 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value  
1342 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension  
1343 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested  
1344 Sequence creation.

1345 The following is an example of a `CreateSequence` message using the

1346 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1347 <soap:Envelope ...>  
1348   <soap:Header>  
1349     ...  
1350     <wsm:UsesSequenceSTR soap:mustUnderstand='true' />  
1351     ...  
1352   </soap:Header>  
1353   <soap:Body>  
1354     <wsm:CreateSequence>  
1355       <wsm:AcksTo>  
1356         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1357       </wsm:AcksTo>  
1358       <wsse:SecurityTokenReference>  
1359         ...  
1360       </wsse:SecurityTokenReference>  
1361     </wsm:CreateSequence>  
1362   </soap:Body>  
1363 </soap:Envelope>
```

## 1364 6.2 Securing Sequences Using SSL/TLS

1365 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).  
1366 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying  
1367 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a  
1368 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a  
1369 SOAP header block within the `CreateSequence` message.

1370 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1371 <wsm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1372 The following describes the content model of the `UsesSequenceSSL` header block.

1373 `/wsm:UsesSequenceSSL`

1374 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to  
1375 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was



1376 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a  
1377 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand  
1378 and correctly implement the functionality described in Section 5.2.1 or else generate a  
1379 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1380 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related  
1381 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from  
1382 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the  
1383 `CreateSequenceResponse` message.

## 1384 **7 References**

### 1385 **7.1 Normative**

#### 1386 **[KEYWORDS]**

1387 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,  
1388 March 1997

1389 <http://www.ietf.org/rfc/rfc2119.txt>

#### 1390 **[WS-RM Policy]**

1391 OASIS WS-RX Technical Committee Draft, "[Web Services ReliableMessaging Policy Assertion\( WS-RM  
1392 Policy\)](#)" October 2006

1393 <http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-wd-11.pdf>

#### 1394 **[SOAP 1.1]**

1395 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

1396 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

#### 1397 **[SOAP 1.2]**

1398 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

1399 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

#### 1400 **[URI]**

1401 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,  
1402 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1403 <http://ietf.org/rfc/rfc3986>

#### 1404 **[UUID]**

1405 P. Leach, M. Mealling, R. Salz, "[A Universally Unique IDentifier \(UUID\) URN Namespace](#)," RFC 4122,  
1406 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1407 <http://www.ietf.org/rfc/rfc4122.txt>

#### 1408 **[XML]**

1409 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#)", September 2006.

1410 <http://www.w3.org/TR/REC-xml/>

#### 1411 **[XML-ns]**

1412 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

1413 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

#### 1414 **[XML-Schema Part1]**

1415 W3C Recommendation, "[XML Schema Part 1: Structures](#)," October 2004.

1416 <http://www.w3.org/TR/xmlschema-1/>

1417 **[XML-Schema Part2]**

1418 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

1419 <http://www.w3.org/TR/xmlschema-2/>

1420 **[XPath 1.0]**

1421 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

1422 <http://www.w3.org/TR/xpath>

1423 **[WSDL 1.1]**

1424 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

1425 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1426 **[WS-Addressing]**

1427 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

1428 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

1429 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

1430 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1431 **7.2 Non-Normative**

1432 **[BSP 1.0]**

1433 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

1434 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

1435 **[RDDL 2.0]**

1436 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

1437 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

1438 **[RFC 2617]**

1439 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP  
1440 Authentication: Basic and Digest Access Authentication," June 1999.

1441 <http://www.ietf.org/rfc/rfc2617.txt>

1442 **[RFC 4346]**

1443 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

1444 <http://www.ietf.org/rfc/rfc4346.txt>

1445 **[WS-Policy]**

1446 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

1447 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

1448 **[WS-PolicyAttachment]**

1449 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

1450 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-  
1451 20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1452 **[WS-Security]**

1453 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:  
1454 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1455 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

1456 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:  
1457 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

1458 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

1459 **[RTTM]**

1460 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May  
1461 1992.

1462 <http://www.rfc-editor.org/rfc/rfc1323.txt>

1463 **[SecurityPolicy]**

1464 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

1465 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

1466 **[SecureConversation]**

1467 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February  
1468 2005.

1469 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

1470 **[Trust]**

1471 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

1472 <http://schemas.xmlsoap.org/ws/2005/02/trust>

## 1473 Appendix A. Schema

1474 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-  
1475 Schema Part2] is located at:

1476 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

1477 The following copy is provided for reference.

```
1478 <?xml version="1.0" encoding="UTF-8"?>
1479 <!--
1480 OASIS takes no position regarding the validity or scope of any intellectual
1481 property or other rights that might be claimed to pertain to the
1482 implementation or use of the technology described in this document or the
1483 extent to which any license under such rights might or might not be available;
1484 neither does it represent that it has made any effort to identify any such
1485 rights. Information on OASIS's procedures with respect to rights in OASIS
1486 specifications can be found at the OASIS website. Copies of claims of rights
1487 made available for publication and any assurances of licenses to be made
1488 available, or the result of an attempt made to obtain a general license or
1489 permission for the use of such proprietary rights by implementors or users of
1490 this specification, can be obtained from the OASIS Executive Director.
1491 OASIS invites any interested party to bring to its attention any copyrights,
1492 patents or patent applications, or other proprietary rights which may cover
1493 technology that may be required to implement this specification. Please
1494 address the information to the OASIS Executive Director.
1495 Copyright © OASIS Open 2002-2006. All Rights Reserved.
1496 This document and translations of it may be copied and furnished to others,
1497 and derivative works that comment on or otherwise explain it or assist in its
1498 implementation may be prepared, copied, published and distributed, in whole or
1499 in part, without restriction of any kind, provided that the above copyright
1500 notice and this paragraph are included on all such copies and derivative
1501 works. However, this document itself does not be modified in any way, such as
1502 by removing the copyright notice or references to OASIS, except as needed for
1503 the purpose of developing OASIS specifications, in which case the procedures
1504 for copyrights defined in the OASIS Intellectual Property Rights document must
1505 be followed, or as required to translate it into languages other than English.
1506 The limited permissions granted above are perpetual and will not be revoked by
1507 OASIS or its successors or assigns.
1508 This document and the information contained herein is provided on an "AS IS"
1509 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1510 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1511 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1512 FOR A PARTICULAR PURPOSE.
1513 -->
1514 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1515 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1516 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1517 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1518 elementFormDefault="qualified" attributeFormDefault="unqualified">
1519   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1520     schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1521   <!-- Protocol Elements -->
1522   <xs:complexType name="SequenceType">
1523     <xs:sequence>
1524       <xs:element ref="wsrm:Identifier"/>
1525       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1526       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1527     maxOccurs="unbounded"/>
1528     </xs:sequence>
```

```

1529     <xs:anyAttribute namespace="##other" processContents="lax"/>
1530 </xs:complexType>
1531 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1532 <xs:element name="SequenceAcknowledgement">
1533     <xs:complexType>
1534         <xs:sequence>
1535             <xs:element ref="wsrm:Identifier"/>
1536             <xs:choice>
1537                 <xs:sequence>
1538                     <xs:choice>
1539                         <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1540                             <xs:complexType>
1541                                 <xs:sequence/>
1542                                 <xs:attribute name="Upper" type="xs:unsignedLong"
1543 use="required"/>
1544                                 <xs:attribute name="Lower" type="xs:unsignedLong"
1545 use="required"/>
1546                             <xs:anyAttribute namespace="##other" processContents="lax"/>
1547                             </xs:complexType>
1548                         </xs:element>
1549                         <xs:element name="None">
1550                             <xs:complexType>
1551                                 <xs:sequence/>
1552                             </xs:complexType>
1553                         </xs:element>
1554                     </xs:choice>
1555                     <xs:element name="Final" minOccurs="0">
1556                         <xs:complexType>
1557                             <xs:sequence/>
1558                         </xs:complexType>
1559                     </xs:element>
1560                 </xs:sequence>
1561                 <xs:element name="Nack" type="xs:unsignedLong"
1562 maxOccurs="unbounded"/>
1563             </xs:choice>
1564             <xs:any namespace="##other" processContents="lax" minOccurs="0"
1565 maxOccurs="unbounded"/>
1566         </xs:sequence>
1567         <xs:anyAttribute namespace="##other" processContents="lax"/>
1568     </xs:complexType>
1569 </xs:element>
1570 <xs:complexType name="AckRequestedType">
1571     <xs:sequence>
1572         <xs:element ref="wsrm:Identifier"/>
1573         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1574 maxOccurs="unbounded"/>
1575     </xs:sequence>
1576     <xs:anyAttribute namespace="##other" processContents="lax"/>
1577 </xs:complexType>
1578 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1579 <xs:complexType name="MessagePendingType">
1580     <xs:sequence>
1581         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1582 maxOccurs="unbounded"/>
1583     </xs:sequence>
1584     <xs:attribute name="pending" type="xs:boolean"/>
1585     <xs:anyAttribute namespace="##other" processContents="lax"/>
1586 </xs:complexType>
1587 <xs:element name="MessagePending" type="wsrm:MessagePendingType"/>
1588 <xs:element name="Identifier">
1589     <xs:complexType>
1590         <xs:annotation>
1591             <xs:documentation>

```

```

1592         This type is for elements whose [children] is an anyURI and can have
1593 arbitrary attributes.
1594         </xs:documentation>
1595     </xs:annotation>
1596     <xs:simpleContent>
1597         <xs:extension base="xs:anyURI">
1598             <xs:anyAttribute namespace="##other" processContents="lax"/>
1599         </xs:extension>
1600     </xs:simpleContent>
1601 </xs:complexType>
1602 </xs:element>
1603 <xs:element name="Address">
1604     <xs:complexType>
1605         <xs:simpleContent>
1606             <xs:extension base="xs:anyURI">
1607                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1608             </xs:extension>
1609         </xs:simpleContent>
1610     </xs:complexType>
1611 </xs:element>
1612 <xs:complexType name="MakeConnectionType">
1613     <xs:sequence>
1614         <xs:element ref="wsrm:Identifier" minOccurs="0" maxOccurs="1"/>
1615         <xs:element ref="wsrm:Address" minOccurs="0" maxOccurs="1"/>
1616         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1617 maxOccurs="unbounded"/>
1618     </xs:sequence>
1619     <xs:anyAttribute namespace="##other" processContents="lax"/>
1620 </xs:complexType>
1621 <xs:element name="MakeConnection" type="wsrm:MakeConnectionType"/>
1622 <xs:simpleType name="MessageNumberType">
1623     <xs:restriction base="xs:unsignedLong">
1624         <xs:minInclusive value="1"/>
1625         <xs:maxInclusive value="9223372036854775807"/>
1626     </xs:restriction>
1627 </xs:simpleType>
1628 <!-- Fault Container and Codes -->
1629 <xs:simpleType name="FaultCodes">
1630     <xs:restriction base="xs:QName">
1631         <xs:enumeration value="wsrm:SequenceTerminated"/>
1632         <xs:enumeration value="wsrm:UnknownSequence"/>
1633         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1634         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1635         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1636         <xs:enumeration value="wsrm:SequenceClosed"/>
1637         <xs:enumeration value="wsrm:WSRMRequired"/>
1638         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1639     </xs:restriction>
1640 </xs:simpleType>
1641 <xs:complexType name="SequenceFaultType">
1642     <xs:sequence>
1643         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1644         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1645         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1646 maxOccurs="unbounded"/>
1647     </xs:sequence>
1648     <xs:anyAttribute namespace="##other" processContents="lax"/>
1649 </xs:complexType>
1650 <xs:complexType name="DetailType">
1651     <xs:sequence>
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:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1658 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1659 <xs:element name="CreateSequenceResponse"
1660 type="wsrm:CreateSequenceResponseType"/>
1661 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1662 <xs:element name="CloseSequenceResponse"
1663 type="wsrm:CloseSequenceResponseType"/>
1664 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1665 <xs:element name="TerminateSequenceResponse"
1666 type="wsrm:TerminateSequenceResponseType"/>
1667 <xs:complexType name="CreateSequenceType">
1668 <xs:sequence>
1669 <xs:element ref="wsrm:AcksTo"/>
1670 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1671 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1672 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1673 maxOccurs="unbounded">
1674 <xs:annotation>
1675 <xs:documentation>
1676 It is the authors intent that this extensibility be used to
1677 transfer a Security Token Reference as defined in WS-Security.
1678 </xs:documentation>
1679 </xs:annotation>
1680 </xs:any>
1681 </xs:sequence>
1682 <xs:anyAttribute namespace="##other" processContents="lax"/>
1683 </xs:complexType>
1684 <xs:complexType name="CreateSequenceResponseType">
1685 <xs:sequence>
1686 <xs:element ref="wsrm:Identifier"/>
1687 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1688 <xs:element name="IncompleteSequenceBehavior"
1689 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1690 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1691 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1692 maxOccurs="unbounded"/>
1693 </xs:sequence>
1694 <xs:anyAttribute namespace="##other" processContents="lax"/>
1695 </xs:complexType>
1696 <xs:complexType name="CloseSequenceType">
1697 <xs:sequence>
1698 <xs:element ref="wsrm:Identifier"/>
1699 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1700 maxOccurs="unbounded"/>
1701 </xs:sequence>
1702 <xs:anyAttribute namespace="##other" processContents="lax"/>
1703 </xs:complexType>
1704 <xs:complexType name="CloseSequenceResponseType">
1705 <xs:sequence>
1706 <xs:element ref="wsrm:Identifier"/>
1707 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1708 maxOccurs="unbounded"/>
1709 </xs:sequence>
1710 <xs:anyAttribute namespace="##other" processContents="lax"/>
1711 </xs:complexType>
1712 <xs:complexType name="TerminateSequenceType">
1713 <xs:sequence>
1714 <xs:element ref="wsrm:Identifier"/>
1715 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1716 maxOccurs="unbounded"/>
1717 </xs:sequence>

```



```

1718     <xs:anyAttribute namespace="##other" processContents="lax"/>
1719 </xs:complexType>
1720 <xs:complexType name="TerminateSequenceResponseType">
1721   <xs:sequence>
1722     <xs:element ref="wsrm:Identifier"/>
1723     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1724 maxOccurs="unbounded"/>
1725   </xs:sequence>
1726   <xs:anyAttribute namespace="##other" processContents="lax"/>
1727 </xs:complexType>
1728 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1729 <xs:complexType name="OfferType">
1730   <xs:sequence>
1731     <xs:element ref="wsrm:Identifier"/>
1732     <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1733     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1734     <xs:element name="IncompleteSequenceBehavior"
1735 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1736     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1737 maxOccurs="unbounded"/>
1738   </xs:sequence>
1739   <xs:anyAttribute namespace="##other" processContents="lax"/>
1740 </xs:complexType>
1741 <xs:complexType name="AcceptType">
1742   <xs:sequence>
1743     <xs:element ref="wsrm:AcksTo"/>
1744     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1745 maxOccurs="unbounded"/>
1746   </xs:sequence>
1747   <xs:anyAttribute namespace="##other" processContents="lax"/>
1748 </xs:complexType>
1749 <xs:element name="Expires">
1750   <xs:complexType>
1751     <xs:simpleContent>
1752       <xs:extension base="xs:duration">
1753         <xs:anyAttribute namespace="##other" processContents="lax"/>
1754       </xs:extension>
1755     </xs:simpleContent>
1756   </xs:complexType>
1757 </xs:element>
1758 <xs:simpleType name="IncompleteSequenceBehaviorType">
1759   <xs:restriction base="xs:string">
1760     <xs:enumeration value="DiscardEntireSequence"/>
1761     <xs:enumeration value="DiscardFollowingFirstGap"/>
1762     <xs:enumeration value="NoDiscard"/>
1763   </xs:restriction>
1764 </xs:simpleType>
1765 <xs:element name="UsesSequenceSTR">
1766   <xs:complexType>
1767     <xs:sequence/>
1768     <xs:anyAttribute namespace="##other" processContents="lax"/>
1769   </xs:complexType>
1770 </xs:element>
1771 <xs:element name="UsesSequenceSSL">
1772   <xs:complexType>
1773     <xs:sequence/>
1774     <xs:anyAttribute namespace="##other" processContents="lax"/>
1775   </xs:complexType>
1776 </xs:element>
1777 <xs:element name="UnsupportedElement">
1778   <xs:simpleType>
1779     <xs:restriction base="xs:QName"/>
1780   </xs:simpleType>

```

1781  
1782

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

## 1783 Appendix B. WSDL

1784 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where  
1785 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be  
1786 present in exchanges with that endpoint.

1787 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not  
1788 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]  
1789 for a higher-level mechanism to indicate that WS-RM is engaged.

1790 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1791 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

1792 The following non-normative copy is provided for reference.

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

```

1836     <xs:schema>
1837         <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
1838 schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-
1839 200608.xsd"/>
1840     </xs:schema>
1841 </wsdl:types>

1842     <wsdl:message name="CreateSequence">
1843         <wsdl:part name="create" element="rm:CreateSequence"/>
1844     </wsdl:message>
1845     <wsdl:message name="CreateSequenceResponse">
1846         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1847     </wsdl:message>
1848     <wsdl:message name="CloseSequence">
1849         <wsdl:part name="close" element="rm:CloseSequence"/>
1850     </wsdl:message>
1851     <wsdl:message name="CloseSequenceResponse">
1852         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1853     </wsdl:message>
1854     <wsdl:message name="TerminateSequence">
1855         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1856     </wsdl:message>
1857     <wsdl:message name="TerminateSequenceResponse">
1858         <wsdl:part name="terminateResponse"
1859 element="rm:TerminateSequenceResponse"/>
1860     </wsdl:message>
1861     <wsdl:message name="MakeConnection">
1862         <wsdl:part name="makeConnection" element="rm:MakeConnection"/>
1863     </wsdl:message>

1864     <wsdl:portType name="SequenceAbstractPortType">
1865         <wsdl:operation name="CreateSequence">
1866             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1867 open.org/ws-rx/wsrn/200608/CreateSequence"/>
1868             <wsdl:output message="tns:CreateSequenceResponse"
1869 wsaw:Action="http://docs.oasis-open.org/ws-
1870 rx/wsrn/200608/CreateSequenceResponse"/>
1871         </wsdl:operation>
1872         <wsdl:operation name="CloseSequence">
1873             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1874 open.org/ws-rx/wsrn/200608/CloseSequence"/>
1875             <wsdl:output message="tns:CloseSequenceResponse"
1876 wsaw:Action="http://docs.oasis-open.org/ws-
1877 rx/wsrn/200608/CloseSequenceResponse"/>
1878         </wsdl:operation>
1879         <wsdl:operation name="TerminateSequence">
1880             <wsdl:input message="tns:TerminateSequence"
1881 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence"/>
1882             <wsdl:output message="tns:TerminateSequenceResponse"
1883 wsaw:Action="http://docs.oasis-open.org/ws-
1884 rx/wsrn/200608/TerminateSequenceResponse"/>
1885         </wsdl:operation>
1886         <wsdl:operation name="MakeConnection">
1887             <wsdl:input message="tns:MakeConnection" wsaw:Action="http://docs.oasis-
1888 open.org/ws-rx/wsrn/200608/MakeConnection"/>
1889             <!-- As described in section 3.10, the MakeConnection operation
1890 establishes a connection. If a matching message is available then
1891 the back-channel of the connection will be used to carry the
1892 message. In SOAP terms the returned message is not a response,
1893 so there is no WSDL output message. -->
1894         </wsdl:operation>
1895     </wsdl:portType>

```

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

## 1897 Appendix C. Message Examples

### 1898 Appendix C.1 Create Sequence

#### 1899 Create Sequence

```
1900 <?xml version="1.0" encoding="UTF-8"?>
1901 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1902 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1903 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1904   <S:Header>
1905     <wsa:MessageID>
1906       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1907     </wsa:MessageID>
1908     <wsa:To>http://example.com/serviceB/123</wsa:To>
1909     <wsa:Action>http://docs.oasis-open.org/ws-
1910 rx/wsmr/200608/CreateSequence</wsa:Action>
1911     <wsa:ReplyTo>
1912       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1913     </wsa:ReplyTo>
1914   </S:Header>
1915   <S:Body>
1916     <wsmr:CreateSequence>
1917       <wsmr:AcksTo>
1918         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1919       </wsmr:AcksTo>
1920     </wsmr:CreateSequence>
1921   </S:Body>
1922 </S:Envelope>
```

#### 1923 Create Sequence Response

```
1924 <?xml version="1.0" encoding="UTF-8"?>
1925 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1926 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1927 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1928   <S:Header>
1929     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1930     <wsa:RelatesTo>
1931       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1932     </wsa:RelatesTo>
1933     <wsa:Action>
1934       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
1935     </wsa:Action>
1936   </S:Header>
1937   <S:Body>
1938     <wsmr:CreateSequenceResponse>
1939       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1940     </wsmr:CreateSequenceResponse>
1941   </S:Body>
1942 </S:Envelope>
```

### 1943 Appendix C.2 Initial Transmission

1944 The following example WS-ReliableMessaging headers illustrate the message exchange in the above  
1945 figure. The three messages have the following headers; the third message is identified as the last  
1946 message in the Sequence:

1947 **Message 1**

```
1948 <?xml version="1.0" encoding="UTF-8"?>
1949 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1950 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1951 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1952   <S:Header>
1953     <wsa:MessageID>
1954       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1955     </wsa:MessageID>
1956     <wsa:To>http://example.com/serviceB/123</wsa:To>
1957     <wsa:From>
1958       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1959     </wsa:From>
1960     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1961     <wsmr:Sequence>
1962       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1963       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1964     </wsmr:Sequence>
1965   </S:Header>
1966   <S:Body>
1967     <!-- Some Application Data -->
1968   </S:Body>
1969 </S:Envelope>
```

1970 **Message 2**

```
1971 <?xml version="1.0" encoding="UTF-8"?>
1972 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1973 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1974 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1975   <S:Header>
1976     <wsa:MessageID>
1977       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1978     </wsa:MessageID>
1979     <wsa:To>http://example.com/serviceB/123</wsa:To>
1980     <wsa:From>
1981       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1982     </wsa:From>
1983     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1984     <wsmr:Sequence>
1985       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1986       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1987     </wsmr:Sequence>
1988   </S:Header>
1989   <S:Body>
1990     <!-- Some Application Data -->
1991   </S:Body>
1992 </S:Envelope>
```

1993 **Message 3**

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

```

2005 </wsa:From>
2006 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2007 <wsrm:Sequence>
2008 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2009 <wsrm:MessageNumber>3</wsrm:MessageNumber>
2010 </wsrm:Sequence>
2011 <wsrm:AckRequested>
2012 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2013 </wsrm:AckRequested>
2014 </S:Header>
2015 <S:Body>
2016 <!-- Some Application Data -->
2017 </S:Body>
2018 </S:Envelope>

```

### 2019 **Appendix C.3 First Acknowledgement**

2020 Message number 2 has not been accepted by the RM Destination due to some transmission error so it  
2021 responds with an Acknowledgement for messages 1 and 3:

```

2022 <?xml version="1.0" encoding="UTF-8"?>
2023 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2024 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2025 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2026 <S:Header>
2027 <wsa:MessageID>
2028 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
2029 </wsa:MessageID>
2030 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2031 <wsa:From>
2032 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2033 </wsa:From>
2034 <wsa:Action>
2035 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
2036 </wsa:Action>
2037 <wsrm:SequenceAcknowledgement>
2038 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2039 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2040 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2041 </wsrm:SequenceAcknowledgement>
2042 </S:Header>
2043 <S:Body/>
2044 </S:Envelope>

```

### 2045 **Appendix C.4 Retransmission**

2046 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and  
2047 requests an Acknowledgement:

```

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

```



```

2060 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2061 <wsrm:Sequence>
2062 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2063 <wsrm:MessageNumber>2</wsrm:MessageNumber>
2064 </wsrm:Sequence>
2065 <wsrm:AckRequested>
2066 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2067 </wsrm:AckRequested>
2068 </S:Header>
2069 <S:Body>
2070 <!-- Some Application Data -->
2071 </S:Body>
2072 </S:Envelope>

```

## 2073 Appendix C.5 Termination

2074 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then  
 2075 be terminated:

```

2076 <?xml version="1.0" encoding="UTF-8"?>
2077 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2078 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2079 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2080 <S:Header>
2081 <wsa:MessageID>
2082 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2083 </wsa:MessageID>
2084 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2085 <wsa:From>
2086 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2087 </wsa:From>
2088 <wsa:Action>
2089 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
2090 </wsa:Action>
2091 <wsrm:SequenceAcknowledgement>
2092 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2093 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2094 </wsrm:SequenceAcknowledgement>
2095 </S:Header>
2096 <S:Body/>
2097 </S:Envelope>

```

## 2098 Terminate Sequence

```

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

```

```

2117     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2118     </wsrm:TerminateSequence>
2119   </S:Body>
2120 </S:Envelope>

```

## 2121 Terminate Sequence Response

```

2122 <?xml version="1.0" encoding="UTF-8"?>
2123 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2124   xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2125   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2126   <S:Header>
2127     <wsa:MessageID>
2128       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2129     </wsa:MessageID>
2130     <wsa:To>http://example.com/serviceA/789</wsa:To>
2131     <wsa:Action>
2132       http://docs.oasis-open.org/ws-rx/wsmr/200608/TerminateSequenceResponse
2133     </wsa:Action>
2134     <wsa:RelatesTo>
2135       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2136     </wsa:RelatesTo>
2137     <wsa:From>
2138       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2139     </wsa:From>
2140   </S:Header>
2141   <S:Body>
2142     <wsrm:TerminateSequenceResponse>
2143       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2144     </wsrm:TerminateSequenceResponse>
2145   </S:Body>
2146 </S:Envelope>

```

## 2147 Appendix C.6 MakeConnection

2148 To illustrate how a `MakeConnection` message exchange can be used to deliver messages to an  
 2149 Endpoint that is not addressable, consider the case of a pub/sub scenario in which the Endpoint to which  
 2150 notifications are to be delivered (the "event consumer") is not addressable by the notification sending  
 2151 Endpoint (the "event producer"). In this scenario the event consumer must initiate the connections in order  
 2152 for the notifications to be delivered. One possible set of message exchanges (using HTTP) that  
 2153 demonstrate how this can be achieved using `MakeConnection` is shown below.

2154 **Step 1** – During a "subscribe" operation, the event consumer's EPR specifies the RM anonymous URI  
 2155 and the WS-RM Policy Assertion to indicate whether or not RM is required:

```

2156 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2157   xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2158   xmlns:wsmrp="http://docs.oasis-open.org/ws-rx/wsmrp/200608"
2159   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2160   <S:Header>
2161     <wsa:To> http://example.org/subscriptionService </wsa:To>
2162     <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813</wsa:MessageID>
2163     <wsa:ReplyTo>
2164       <wsa:To> http://client456.org/response </wsa:To>
2165     </wsa:ReplyTo>
2166   </S:Header>
2167   <S:Body>
2168     <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">
2169       <!-- subscription service specific data -->
2170       <targetEPR>

```

```

2171     <wsa:Address>http://docs.oasis-open.org/ws-
2172 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>
2173     <wsa:Metadata>
2174         <wsp:Policy wsu:Id="MyPolicy">
2175             <wsrmp:RMAssertion/>
2176         </wsp:Policy>
2177     </wsa:Metadata>
2178     </targetEPR>
2179 </sub:Subscribe>
2180 </S:Body>
2181 </S:Envelope>

```

2182 In this example the `subscribe` and `targetEPR` elements are simply examples of what a subscription  
2183 request message might contain. Note: the `wsa:Address` element contains the RM anonymous URI  
2184 indicating that the notification producer needs to queue the messages until they are requested using the  
2185 `MakeConnection` message exchange. The EPR also contains the WS-RM Policy Assertion indicating  
2186 the RM must be used when notifications related to this subscription are sent.

2187 **Step 2** – Once the subscription is established, the event consumer checks for a pending message:

```

2188 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2189 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2190 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2191   <S:Header>
2192     <wsa:Action>http://docs.oasis-open.org/ws-
2193 rx/wsrn/200608/MakeConnection</wsa:Action>
2194     <wsa:To> http://example.org/subscriptionService </wsa:To>
2195   </S:Header>
2196   <S:Body>
2197     <wsrm:MakeConnection>
2198       <wsrm:Address>http://docs.oasis-open.org/ws-
2199 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-
2200 446655440000</wsrm:Address>
2201     </wsrm:MakeConnection>
2202   </S:Body>
2203 </S:Envelope>

```

2204 **Step 3** – If there are messages waiting to be delivered then a message will be returned back to the event  
2205 consumer. However, because WS-RM is being used to deliver the messages, the first message returned  
2206 is a `CreateSequence`:

```

2207 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2208 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2209 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2210   <S:Header>
2211     <wsa:Action>http://docs.oasis-open.org/ws-
2212 rx/wsrn/200608/CreateSequence</wsa:Action>
2213     <wsa:To>http://docs.oasis-open.org/ws-
2214 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
2215     <wsa:ReplyTo> http://example.org/subscriptionService </wsa:ReplyTo>
2216     <wsa:MessageID> http://example.org/id-123-456 </wsa:MessageID>
2217   </S:Header>
2218   <S:Body>
2219     <wsrm:CreateSequence>
2220       <wsrm:AcksTo>
2221         <wsa:Address> http://example.org/subscriptionService </wsa:Address>
2222       </wsrm:AcksTo>
2223     </wsrm:CreateSequence>
2224   </S:Body>

```

2225

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

2226 Notice from the perspective of how the RM Source on the event producer interacts with the RM  
2227 Destination of those messages, nothing new is introduced by the use of the `MakeConnection`, the use  
2228 of RM protocol is the same as the case where the event consumer is addressable.

2229 **Step 4** – The event consumer will respond with a `CreateSequenceResponse` message per normal WS-  
2230 Addressing rules:

2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245

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

2246 Note, this message is carried on an HTTP request directed to the `wsa:ReplyTo` EPR, and the HTTP  
2247 response will be an HTTP 202.

2248 **Step 5** – The event consumer checks for another message pending:

2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264

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

2265 Notice this is the same message as the one sent in step 2.

2266 **Step 6** – If there is a message pending for this destination then it is returned on the HTTP response:

2267  
2268  
2269  
2270  
2271  
2272  
2273

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

```

2274     <wsrm:Sequence>
2275         <wsrm:Identifier> http://example.org/rmid-456 </wsrm:Identifier>
2276     </wsrm:Sequence>
2277     <wsrm:MessagePending pending="true"/>
2278 </S:Header>
2279 <S:Body>
2280     <!-- event specific data -->
2281 </S:Body>
2282 </S:Envelope>

```

2283 As noted in step 3, the use of the RM protocol does not change when using `MakeConnection`. The  
2284 format of the messages, the order of the messages sent and the timing of when to send it remains the  
2285 same.

2286 **Step 7** – At some later interval, or immediately due to the `MessagePending` header's "pending"  
2287 attribute being set to "true", the event consumer will poll again:

```

2288 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2289 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrp/200608"
2290 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2291     <S:Header>
2292         <wsa:Action>http://docs.oasis-open.org/ws-
2293 rx/wsrp/200608/MakeConnection</wsa:Action>
2294         <wsa:To> http://example.org/subscriptionService </wsa:To>
2295     </S:Header>
2296     <S:Body>
2297         <wsrm:MakeConnection>
2298             <wsrm:Address>http://docs.oasis-open.org/ws-
2299 rx/wsrp/200608/anonymous?id=550e8400-e29b-11d4-a716-
2300 446655440000</wsrm:Address>
2301         </wsrm:MakeConnection>
2302     </S:Body>
2303 </S:Envelope>

```

2304 Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to  
2305 the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event  
2306 producer to send not only application messages but RM protocol messages (e.g. `CloseSequence`,  
2307 `TerminateSequence` or even additional `CreateSequences`) as needed.

2308 **Step 8** – If at any point in time there are no messages pending, in response to a `MakeConnection` the  
2309 event producer returns an HTTP 202 back to the event consumer. The process then repeats (back to step  
2310 7) until the subscription ends.

## 2311 Appendix D. State Tables

2312 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2313 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2314 Legend:

2315 The first column of these tables contains the motivating event and has the following format:

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

2316 Where:

2317 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as  
2318 described by the specification.

2319 ● [source]: indicates the source of the event; one of:

2320 ● [msg] a Received message

2321 ● [int]: an internal event such as the firing of a timer

2322 ● [app]: the application

2323 ● [unspec]: the source is unspecified

2324 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}

2325 Where:

2326 ● action to take: indicates that the state machine performs the following action. Actions surrounded  
2327 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word  
2328 "Transmit"

2329 ● [next state]: indicates the state to which the state machine will advance upon the performance of  
2330 the action. For ease of reading the next state "same" indicates that the state does not change.

2331 ● {ref} is a reference to the document section describing the behavior in this cell

2332 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these  
2333 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not  
2334 described in this specification and does not indicate normal protocol operation. Implementations MAY  
2335 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations  
2336 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state  
2337 combinations.

2338 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
<b>Create Sequence</b> [unspec] {3.4}	Xmit Create Sequence [Creating] {3.4}	N/A	N/A	N/A	N/A	N/A
<b>Create Sequence Response</b> [msg] {3.4}		Process Create Sequence Response [Created] {3.4}				
<b>Create Sequence Refused Fault</b> [msg] {3.4}		No action [None] {4.6}				
<b>Send message</b> [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [Same] {2}	N/A	N/A
<b>Retransmit of un-ack'd message</b> [int]	N/A	N/A	Xmit message [Same] {2.4}	Xmit message [Same] {2.4}	N/A	N/A
<b>SeqAck (non-final)</b> [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}
<b>Nack</b> [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]
<b>Message Number Rollover Fault</b> [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]
<b>&lt;Close Sequence&gt;</b> [int] {3.5}	N/A		Xmit Close Sequence [Closing] {3.5}	N/A	N/A	N/A
<b>Close Sequence Response</b> [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}
<b>SeqAck (final)</b> [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]
<b>Sequence Closed Fault</b> [msg]	Generate Unknown Sequence Fault	Generate Unknown Sequence Fault	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
{4.7}	[Same] {4.3}	[Same] {4.3}				
<b>Unknown Sequence Fault</b> [msg] {4.3}			Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
<b>Sequence Terminated Fault</b> [msg] {4.2}	N/A		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
<b>Terminate Sequence</b> [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	N/A
<b>Terminate Sequence Response</b> [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}				Terminate Sequence [None] {3.6}
<b>Expires exceeded</b> [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}
<b>Invalid Acknowledgment</b> [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}

2338 Table 2 RM Destination Sequence State Transition Table

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



Events	Sequence States		
	None	Created	Closed
<b>CloseSequence</b> [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}
<b>&lt;CloseSequence autonomously&gt;</b> [int]	N/A	No Action [Closed]	N/A
<b>TerminateSequence</b> [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}
<b>UnknownSequence Fault</b> [msg] {4.3}		Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
<b>SequenceTerminated Fault</b> [msg] {4.2}		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
<b>Invalid Acknowledgement Fault</b> [msg] {4.4}	N/A		
<b>Expires exceeded</b> [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
<b>&lt;Seq Acknowledgement autonomously&gt;</b> [int] {3.9}	N/A	Xmit SeqAck [Same] {3.9}	Xmit SeqAck+Final [Same] {3.9}
<b>Non WSRM message when WSRM required</b> [msg] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	Generate WSRMRequired Fault [Same] {4.8}

2339 The following two tables apply only if the `MakeConnection` mechanism is utilized.

2340 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)]

2341 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)

## 2342 **Appendix E. Acknowledgments**

2343 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following  
2344 authors:

2345 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),  
2346 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),  
2347 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),  
2348 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don  
2349 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),  
2350 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony  
2351 Storey(IBM).

2352 The following individuals have provided invaluable input into the initial contribution:

2353 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen  
2354 Brown(Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM),  
2355 Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott  
2356 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal  
2357 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter  
2358 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish  
2359 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2360 The following individuals were members of the committee during the development of this specification:

2361 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben  
2362 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd  
2363 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul  
2364 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques  
2365 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),  
2366 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair  
2367 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),  
2368 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul  
2369 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt  
2370 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff  
2371 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku  
2372 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert  
2373 Pilz(BEA), Martin Raepfle(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom  
2374 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von  
2375 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçinalp(SAP), Nobuyuki  
2376 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 <a href="http://docs.oasis-open.org/wsrn/200510/">http://docs.oasis-open.org/wsrn/200510/</a> )
wd-05	2005-09-30	Anish Karmarkar	i045 (Include SecureConversation as a reference and move it to non-normative citation)
wd-05	2005-09-30	Anish Karmarkar	i046 (change the type of wsrn:FaultCode element)
wd-06	2005-11-02	Gilbert Pilz	Start wd-06 by changing title page from cd-01.
wd-06	2005-11-03	Gilbert Pilz	i047 (Reorder spec sections)
wd-07	2005-11-17	Gilbert Pilz	Start wd-07
wd-07	2005-11-28	Doug Davis	i071 – except for period in Appendix headings
wd-07	2005-11-28	Doug Davis	i10
wd-07	2005-11-28	Doug Davis	i030
wd-07	2005-11-28	Doug Davis	i037
wd-07	2005-11-28	Doug Davis	i038
wd-07	2005-11-28	Doug Davis	i041
wd-07	2005-11-28	Doug Davis	i043
wd-07	2005-11-28	Doug Davis	i044

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

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

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

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



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

## 2378 **Appendix G. Notices**

2379 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2380 might be claimed to pertain to the implementation or use of the technology described in this document or  
2381 the extent to which any license under such rights might or might not be available; neither does it represent  
2382 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
2383 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
2384 available for publication and any assurances of licenses to be made available, or the result of an attempt  
2385 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
2386 users of this specification, can be obtained from the OASIS Executive Director.

2387 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
2388 other proprietary rights which may cover technology that may be required to implement this specification.  
2389 Please address the information to the OASIS Executive Director.

2390 Copyright (C) OASIS Open (2006). All Rights Reserved.

2391 This document and translations of it may be copied and furnished to others, and derivative works that  
2392 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
2393 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
2394 this paragraph are included on all such copies and derivative works. However, this document itself may  
2395 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
2396 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
2397 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
2398 into languages other than English.

2399 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2400 or assigns.

2401 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2402 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2403 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
2404 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.