



# 1 Web Services Reliable Messaging (WS- 2 ReliableMessaging) Version 1.1

## 3 OASIS Standard incorporating Approved Errata

4 07 January 2008

### 5 Specification URIs:

#### 6 This Version:

- 7 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.pdf>
- 8 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.html>
- 9 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.doc>

#### 10 Previous Version:

- 11 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01.pdf>
- 12 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01.html>
- 13 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01.doc>

#### 14 Latest Version:

- 15 <http://docs.oasis-open.org/ws-rx/wsrn/v1.1/wsrn.pdf>
- 16 <http://docs.oasis-open.org/ws-rx/wsrn/v1.1/wsrn.html>
- 17 <http://docs.oasis-open.org/ws-rx/wsrn/v1.1/wsrn.doc>

#### 18 Technical Committee:

19 OASIS Web Services Reliable Exchange (WS-RX) TC

#### 20 Chairs:

- 21 Paul Fremantle <[paul@wso2.com](mailto:paul@wso2.com)>
- 22 Sanjay Patil <[sanjay.patil@sap.com](mailto:sanjay.patil@sap.com)>

#### 23 Editors:

- 24 Doug Davis, IBM <[dug@us.ibm.com](mailto:dug@us.ibm.com)>
- 25 Anish Karmarkar, Oracle <[Anish.Karmarkar@oracle.com](mailto:Anish.Karmarkar@oracle.com)>
- 26 Gilbert Pilz, BEA <[gpilz@bea.com](mailto:gpilz@bea.com)>
- 27 Steve Winkler, SAP <[steve.winkler@sap.com](mailto:steve.winkler@sap.com)>
- 28 Ümit Yalçınalp, SAP <[umit.yalcinalp@sap.com](mailto:umit.yalcinalp@sap.com)>

#### 29 Related Work:

30 This specification replaces or supercedes:

- 31
  - WS-ReliableMessaging v1.0

#### 32 Declared XML Namespaces:

33 <http://docs.oasis-open.org/ws-rx/wsrn/200702>

#### 34 Abstract:

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

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

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

50 **Status:**

51 This document was last revised or approved by the WS-RX Technical Committee on the above  
52 date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved  
53 Version" location noted above for possible later revisions of this document.

54 Technical Committee members should send comments on this specification to the Technical  
55 Committee's email list. Others should send comments to the Technical Committee by using the  
56 "Send A Comment" button on the Technical Committee's web page at [http://www.oasis-](http://www.oasis-open.org/committees/ws-rx/)  
57 [open.org/committees/ws-rx/](http://www.oasis-open.org/committees/ws-rx/).

58 For information on whether any patents have been disclosed that may be essential to  
59 implementing this specification, and any offers of patent licensing terms, please refer to the  
60 Intellectual Property Rights section of the Technical Committee web page ([http://www.oasis-](http://www.oasis-open.org/committees/ws-rx/ipr.php)  
61 [open.org/committees/ws-rx/ipr.php](http://www.oasis-open.org/committees/ws-rx/ipr.php)).

62 The non-normative errata page for this specification is located at [http://www.oasis-](http://www.oasis-open.org/committees/ws-rx/)  
63 [open.org/committees/ws-rx/](http://www.oasis-open.org/committees/ws-rx/).

---

## 64 Notices

65 Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

66 All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual  
67 Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

68 This document and translations of it may be copied and furnished to others, and derivative works that  
69 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published,  
70 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice  
71 and this section are included on all such copies and derivative works. However, this document itself may  
72 not be modified in any way, including by removing the copyright notice or references to OASIS, except as  
73 needed for the purpose of developing any document or deliverable produced by an OASIS Technical  
74 Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be  
75 followed) or as required to translate it into languages other than English.

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

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

83 OASIS requests that any OASIS Party or any other party that believes it has patent claims that would  
84 necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to  
85 notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such  
86 patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced  
87 this specification.

88 OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any  
89 patent claims that would necessarily be infringed by implementations of this specification by a patent  
90 holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR  
91 Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims  
92 on its website, but disclaims any obligation to do so.

93 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
94 might be claimed to pertain to the implementation or use of the technology described in this document or  
95 the extent to which any license under such rights might or might not be available; neither does it represent  
96 that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to  
97 rights in any document or deliverable produced by an OASIS Technical Committee can be found on the  
98 OASIS website. Copies of claims of rights made available for publication and any assurances of licenses  
99 to be made available, or the result of an attempt made to obtain a general license or permission for the  
100 use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS  
101 Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any  
102 information or list of intellectual property rights will at any time be complete, or that any claims in such list  
103 are, in fact, Essential Claims.

104 The name "OASIS", WS-ReliableMessaging, WSRM and WS-RX are trademarks of [OASIS](http://www.oasis-open.org/who/trademark.php), the owner  
105 and developer of this specification, and should be used only to refer to the organization and its official  
106 outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the  
107 right to enforce its marks against misleading uses. Please see [http://www.oasis-](http://www.oasis-open.org/who/trademark.php)  
108 [open.org/who/trademark.php](http://www.oasis-open.org/who/trademark.php) for above guidance.

---

## 109 Table of Contents

110	1	Introduction .....	6
111	1.1	Terminology .....	6
112	1.2	Normative References .....	7
113	1.3	Non-Normative References.....	7
114	1.4	Namespace .....	8
115	1.5	Conformance.....	9
116	2	Reliable Messaging Model .....	10
117	2.1	Glossary.....	11
118	2.2	Protocol Preconditions .....	12
119	2.3	Protocol Invariants.....	12
120	2.4	Delivery Assurances .....	12
121	2.5	Example Message Exchange.....	13
122	3	RM Protocol Elements.....	16
123	3.1	Considerations on the Use of Extensibility Points .....	16
124	3.2	Considerations on the Use of "Piggy-Backing" .....	16
125	3.3	Composition with WS-Addressing .....	16
126	3.4	Sequence Creation .....	17
127	3.5	Closing A Sequence .....	21
128	3.6	Sequence Termination.....	23
129	3.7	Sequences .....	25
130	3.8	Request Acknowledgement .....	26
131	3.9	Sequence Acknowledgement.....	27
132	4	Faults.....	30
133	4.1	SequenceFault Element.....	31
134	4.2	Sequence Terminated .....	32
135	4.3	Unknown Sequence.....	32
136	4.4	Invalid Acknowledgement .....	33
137	4.5	Message Number Rollover .....	33
138	4.6	Create Sequence Refused.....	34
139	4.7	Sequence Closed .....	34
140	4.8	WSRM Required.....	35
141	5	Security Threats and Countermeasures .....	36
142	5.1	Threats and Countermeasures .....	36
143	5.2	Security Solutions and Technologies .....	38
144	6	Securing Sequences .....	41

145	6.1 Securing Sequences Using WS-Security .....	41
146	6.2 Securing Sequences Using SSL/TLS .....	42
147	Appendix A. Schema .....	44
148	Appendix B. WSDL.....	49
149	Appendix C. Message Examples .....	51
150	Appendix C.1 Create Sequence.....	51
151	Appendix C.2 Initial Transmission .....	51
152	Appendix C.3 First Acknowledgement .....	53
153	Appendix C.4 Retransmission.....	53
154	Appendix C.5 Termination .....	54
155	Appendix D. State Tables .....	56
156	Appendix E. Acknowledgments.....	61
157		

---

# 158 1 Introduction

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

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

## 168 1.1 Terminology

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

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

- 173 • The syntax appears as an XML instance, but values in italics indicate data types instead of  
174 values.
- 175 • Characters are appended to elements and attributes to indicate cardinality:
  - 176 ○ "?" (0 or 1)
  - 177 ○ "\*" (0 or more)
  - 178 ○ "+" (1 or more)
- 179 • The character "|" is used to indicate a choice between alternatives.
- 180 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group  
181 with respect to cardinality or choice.
- 182 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content  
183 specified in this document. Additional children elements and/or attributes MAY be added at the  
184 indicated extension points but they MUST NOT contradict the semantics of the parent and/or  
185 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 186 • XML namespace prefixes (see section 1.4) are used to indicate the namespace of the element  
187 being defined.

188 Elements and Attributes defined by this specification are referred to in the text of this document using  
189 XPath 1.0 [XPath\_10] expressions. Extensibility points are referred to using an extended version of this  
190 syntax:

- 191 • An element extensibility point is referred to using {any} in place of the element name. This  
192 indicates that any element name can be used, from any namespace other than the wsrn:  
193 namespace.
- 194 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This  
195 indicates that any attribute name can be used, from any namespace other than the wsrn:  
196 namespace.

## 197 1.2 Normative References

- 198 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC  
199 2119, Harvard University, March 1997  
200 <http://www.ietf.org/rfc/rfc2119.txt>
- 201 **[WS-RM Policy]** OASIS WS-RX Technical OASIS Standard, "Web Services Reliable Messaging  
202 Policy Assertion( WS-RM Policy)," June 2007  
203 <http://docs.oasis-open.org/ws-rx/wsrmp/v1.1/wsrmp.pdf>
- 204 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.  
205 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 206 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June  
207 2003.  
208 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 209 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):  
210 Generic Syntax," RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January  
211 2005.  
212 <http://ietf.org/rfc/rfc3986>
- 213 **[UUID]** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN  
214 Namespace," RFC 4122, Microsoft, Refactored Networks - LLC, DataPower  
215 Technology Inc, July 2005  
216 <http://www.ietf.org/rfc/rfc4122.txt>
- 217 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth  
218 Edition)", September 2006.  
219 <http://www.w3.org/TR/REC-xml/>
- 220 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.  
221 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 222 **[XML-Schema Part1]** W3C Recommendation, "XML Schema Part 1: Structures," October 2004.  
223 <http://www.w3.org/TR/xmlschema-1/>
- 224 **[XML-Schema Part2]** W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.  
225 <http://www.w3.org/TR/xmlschema-2/>
- 226 **[XPath 1.0]** W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16  
227 November 1999.  
228 <http://www.w3.org/TR/xpath>
- 229 **[WSDL 1.1]** W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.  
230 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 231 **[WS-Addressing]** W3C Recommendation, "Web Services Addressing 1.0 – Core," May 2006.  
232 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>  
233 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding," May  
234 2006  
235 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

## 236 1.3 Non-Normative References

- 237 **[BSP 1.0]** WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006  
238 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>  
239
- 240 **[RDDL 2.0]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language  
241 (RDDL) 2.0," January 2004  
242 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>
- 243 **[RFC 2617]** J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L.  
244 Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June



245 1999.  
 246 <http://www.ietf.org/rfc/rfc2617.txt>

247 **[RFC 4346]** T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version  
 248 1.1," April 2006.  
 249 <http://www.ietf.org/rfc/rfc4346.txt>

250 **[WS-Policy]** W3C Member Submission "Web Services Policy 1.2 - Framework", April 2006  
 251 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>  
 252  
 253 W3C Recommendation, "Web Services Policy 1.5 - Framework," September  
 254 2007.  
 255 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>

256 **[WS-PolicyAttachment]** W3C Member Submission "Web Services Policy 1.2 - Attachment", April  
 257 2006  
 258 <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/>  
 259  
 260 W3C Recommendation, "Web Services Policy 1.5 - Attachment," September  
 261 2007.  
 262 <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>

263 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS  
 264 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",  
 265 OASIS Standard 200401, March 2004.  
 266 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)  
 267 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

268  
 269 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS  
 270 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)",  
 271 OASIS Standard 200602, February 2006.  
 272 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

273 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",  
 274 RFC 1323, May 1992.  
 275 <http://www.rfc-editor.org/rfc/rfc1323.txt>

276 **[SecurityPolicy]** G. Della-Libra, et. al. "Web Services Security Policy Language (WS-  
 277 SecurityPolicy)", July 2005  
 278 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

279 **[SecureConversation]** S. Anderson, et al, "Web Services Secure Conversation Language (WS-  
 280 SecureConversation)," February 2005.  
 281 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

282 **[Trust]** S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.  
 283 <http://schemas.xmlsoap.org/ws/2005/02/trust>

## 284 1.4 Namespace

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

286 <http://docs.oasis-open.org/ws-rx/wsrn/200702>

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

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

291 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/200702">http://docs.oasis-open.org/ws-rx/wsrn/200702</a>
wsa	<a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a>
wsam	<a href="http://www.w3.org/2007/05/addressing/metadata">http://www.w3.org/2007/05/addressing/metadata</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>

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

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

## 295 **1.5 Conformance**

296 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or  
 297 REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace  
 298 identifier for this specification (listed in section 1.4) within SOAP Envelopes unless it is conformant with  
 299 this specification.

300 Normative text within this specification takes precedence over normative outlines, which in turn take  
 301 precedence over the XML Schema [[XML Schema Part 1](#), [Part 2](#)] descriptions.

---

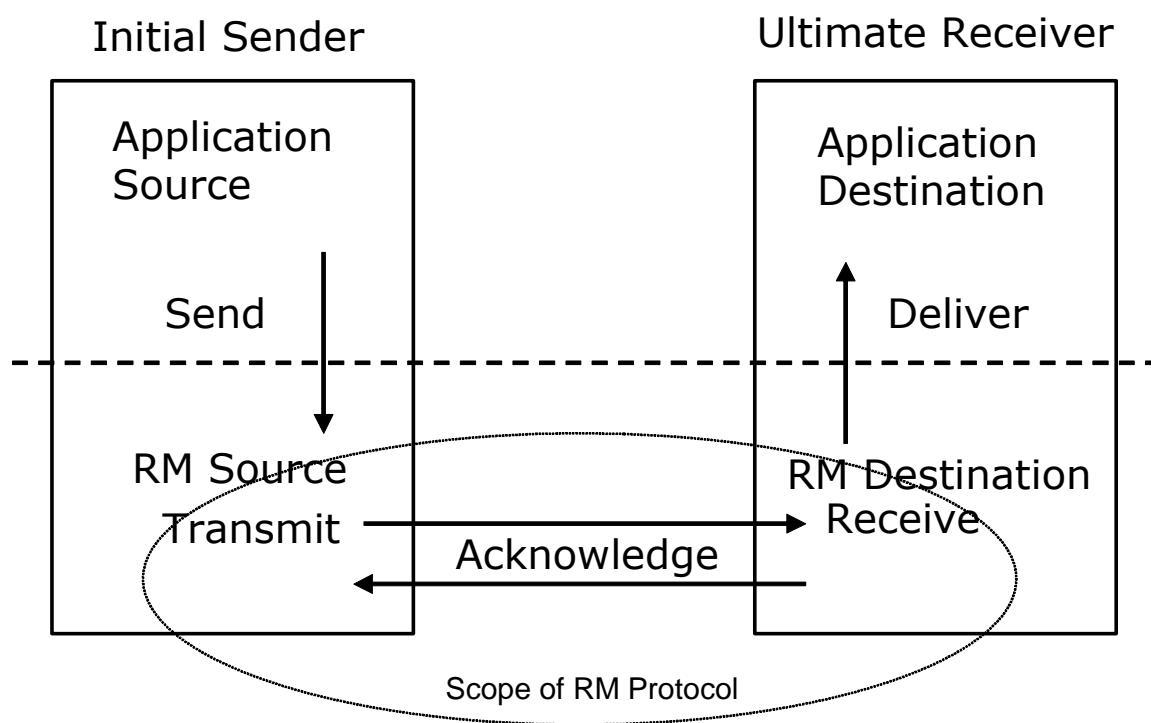
## 302 2 Reliable Messaging Model

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

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

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

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



328 Figure 1: Reliable Messaging Model

## 329 2.1 Glossary

330 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 360 2.2 Protocol Preconditions

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

- 363 • For any single message exchange the RM Source **MUST** have an endpoint reference that  
364 uniquely identifies the RM Destination Endpoint.
- 365 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 366 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's  
367 policies.
- 368 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**  
369 have a security context.

## 370 2.3 Protocol Invariants

371 During the lifetime of a `Sequence`, the following invariants are **REQUIRED** for correctness:

- 372 • The RM Source **MUST** assign each message within a `Sequence` a message number (defined  
373 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers  
374 **MUST** be assigned in the same order in which messages are sent by the Application Source.
- 375 • Within every `Acknowledgement Message` it issues, the RM Destination **MUST** include one or  
376 more `AcknowledgementRange` child elements that contain, in their collective ranges, the  
377 message number of every message accepted by the RM Destination. The RM Destination **MUST**  
378 exclude, in the `AcknowledgementRange` elements, the message numbers of any messages it  
379 has not accepted. If no messages have been received the RM Destination **MUST** return `None`  
380 instead of an `AcknowledgementRange(s)`. The RM Destination **MAY** transmit a `Nack` for a  
381 specific message or messages instead of an `AcknowledgementRange(s)`.
- 382 • While the `Sequence` is not closed or terminated, the RM Source **SHOULD** retransmit  
383 unacknowledged messages.

## 384 2.4 Delivery Assurances

385 This section defines a number of Delivery Assurance assertions, which can be supported by RM Sources  
386 and RM Destinations. These assertions can be specified as policy assertions using the WS-Policy  
387 framework [WS-Policy]. For details on this see the WSRM Policy specification [WS-RM Policy].

388 `AtLeastOnce`

389 Each message is to be delivered at least once, or else an error **MUST** be raised by the RM  
390 Source and/or RM Destination. The requirement on an RM Source is that it **SHOULD** retry  
391 transmission of every message sent by the Application Source until it receives an

392 acknowledgement from the RM Destination. The requirement on the RM Destination is that it  
393 SHOULD retry the transfer to the Application Destination of any message that it accepts from the  
394 RM Source, until that message has been successfully delivered. There is no requirement for the  
395 RM Destination to apply duplicate message filtering.

#### 396 AtMostOnce

397 Each message is to be delivered at most once. The RM Source MAY retry transmission of  
398 unacknowledged messages, but is NOT REQUIRED to do so. The requirement on the RM  
399 Destination is that it MUST filter out duplicate messages, i.e. that it MUST NOT deliver a duplicate  
400 of a message that has already been delivered.

#### 401 ExactlyOnce

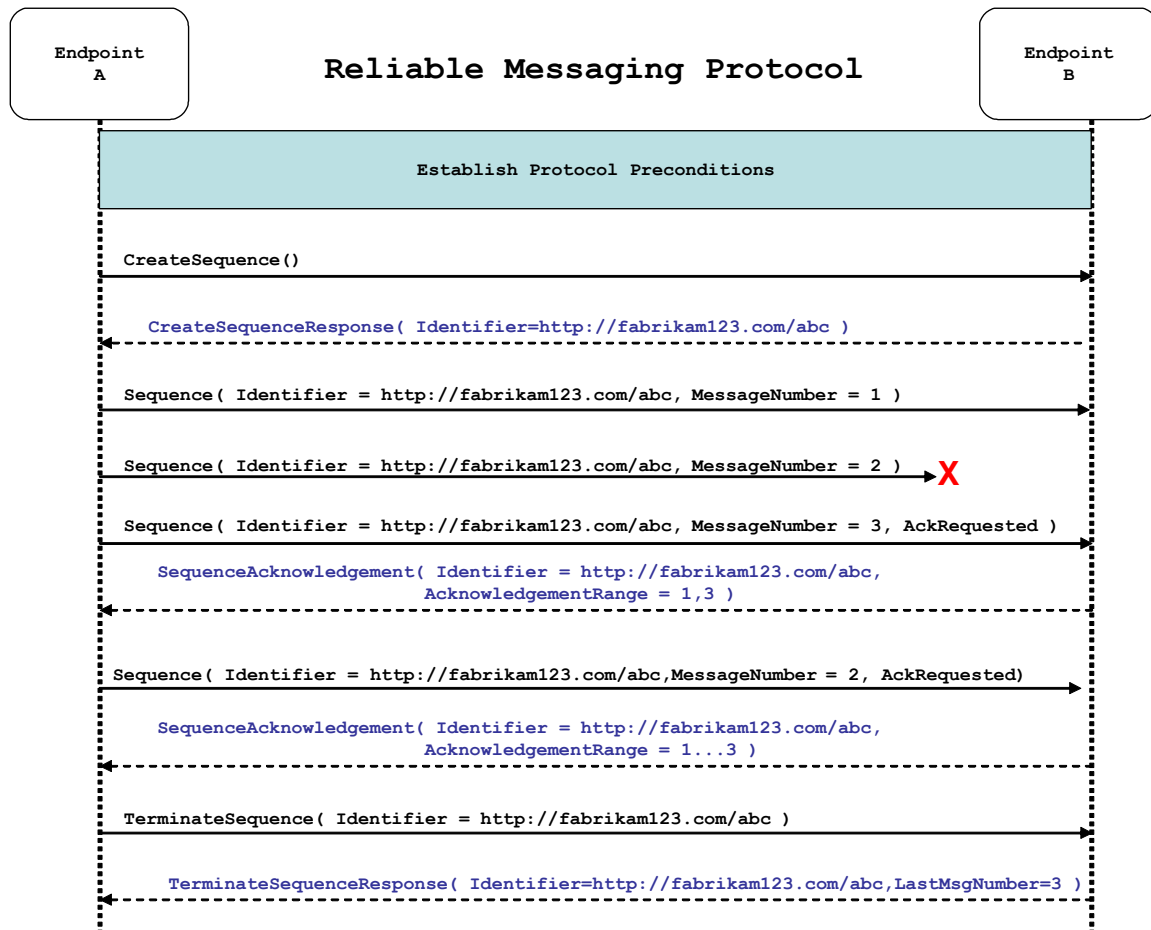
402 Each message is to be delivered exactly once; if a message cannot be delivered then an error  
403 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is  
404 that it SHOULD retry transmission of every message sent by the Application Source until it  
405 receives an acknowledgement from the RM Destination. The requirement on the RM Destination  
406 is that it SHOULD retry the transfer to the Application Destination of any message that it accepts  
407 from the RM Source until that message has been successfully delivered, and that it MUST NOT  
408 deliver a duplicate of a message that has already been delivered.

#### 409 InOrder

410 Messages from each individual Sequence are to be delivered in the same order they have been  
411 sent by the Application Source. The requirement on an RM Source is that it MUST ensure that the  
412 ordinal position of each message in the Sequence (as indicated by a message Sequence number)  
413 is consistent with the order in which the messages have been sent from the Application Source.  
414 The requirement on the RM Destination is that it MUST deliver received messages for each  
415 Sequence in the order indicated by the message numbering. This DeliveryAssurance can be used  
416 in combination with any of the AtLeastOnce, AtMostOnce or ExactlyOnce assertions, and the  
417 requirements of those assertions MUST also be met. In particular if the AtLeastOnce or  
418 ExactlyOnce assertion applies and the RM Destination detects a gap in the Sequence then the  
419 RM Destination MUST NOT deliver any subsequent messages from that Sequence until the  
420 missing messages are received or until the Sequence is closed.

## 421 2.5 Example Message Exchange

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



423 Figure 2: The WS-ReliableMessaging Protocol

- 424 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,  
425 and establishing trust.
- 426 2. The RM Source requests creation of a new Sequence.
- 427 3. The RM Destination creates a new Sequence and returns its unique Identifier.
- 428 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber  
429 1. In the figure above, the RM Source sends 3 messages in the Sequence.
- 430 5. The 2<sup>nd</sup> message in the Sequence is lost in transit.
- 431 6. The 3<sup>rd</sup> message is the last in this Sequence and the RM Source includes an AckRequested  
432 header to ensure that it gets a timely SequenceAcknowledgement for the Sequence.
- 433 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving  
434 the RM Source's AckRequested header.
- 435 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new  
436 message from the perspective of the underlying transport, but it has the same Sequence  
437 Identifier and MessageNumber so the RM Destination can recognize it as a duplicate of the  
438 earlier message, in case the original and retransmitted messages are both Received. The RM  
439 Source includes an AckRequested header in the retransmitted message so the RM Destination  
440 will expedite an acknowledgement.

- 441 9. The RM Destination Receives the second transmission of the message with `MessageNumber 2`  
442 and acknowledges receipt of message numbers 1, 2, and 3.
- 443 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to  
444 the RM Destination indicating that the Sequence is completed. The `TerminateSequence`  
445 message indicates that message number 3 was the last message in the Sequence. The RM  
446 Destination then reclaims any resources associated with the Sequence.
- 447 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source  
448 will not be sending any more messages. The RM Destination sends a  
449 `TerminateSequenceResponse` message to the RM Source and reclaims any resources  
450 associated with the Sequence.
- 451 The RM Source will expect to Receive Acknowledgements from the RM Destination during the course of a  
452 message exchange at occasions described in section 3 below. Should an Acknowledgement not be  
453 Received in a timely fashion, the RM Source MUST re-transmit the message since either the message or  
454 the associated Acknowledgement might have been lost. Since the nature and dynamic characteristics of  
455 the underlying transport and potential intermediaries are unknown in the general case, the timing of re-  
456 transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been  
457 demonstrated to cause transport or intermediary flooding which are counterproductive to the intention of  
458 providing a reliable exchange of messages. Consequently, implementers are encouraged to utilize  
459 adaptive mechanisms that dynamically adjust re-transmission time and the back-off intervals that are  
460 appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP  
461 transports, a mechanism similar to that described as RTTM in RFC 1323 [[RTTM](#)] SHOULD be considered.
- 462 Now that the basic model has been outlined, the details of the elements used in this protocol are now  
463 provided in section 3.



---

## 464 3 RM Protocol Elements

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

### 467 3.1 Considerations on the Use of Extensibility Points

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

### 472 3.2 Considerations on the Use of "Piggy-Backing"

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

### 482 3.3 Composition with WS-Addressing

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

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

491 `http://docs.oasis-open.org/ws-rx/wsrn/200702/CreateSequence`

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

494 `http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement`

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

497 `http://docs.oasis-open.org/ws-rx/wsrn/200702/AckRequested`

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

## 500 3.4 Sequence Creation

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

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

508 The following exemplar defines the `CreateSequence` syntax:

```
509 <wsmr:CreateSequence ...>
510   <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>
511   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?
512   <wsmr:Offer ...>
513     <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
514     <wsmr:Endpoint> wsa:EndpointReferenceType </wsmr:Endpoint>
515     <wsmr:Expires ...> xs:duration </wsmr:Expires> ?
516     <wsmr:IncompleteSequenceBehavior>
517       wsmr:IncompleteSequenceBehaviorType
518     </wsmr:IncompleteSequenceBehavior> ?
519     ...
520   </wsmr:Offer> ?
521   ...
522 </wsmr:CreateSequence>
```

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

524 `/wsmr:CreateSequence`

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

529 `/wsmr:CreateSequence/wsmr:AcksTo`

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

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

539 `/wsmr:CreateSequence/wsmr:Expires`

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

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

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

547 /wsmr:CreateSequence/wsmr:Offer  
548 This element, if present, enables an RM Source to offer a corresponding Sequence for the reliable  
549 exchange of messages Transmitted from RM Destination to RM Source.

550 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier  
551 The RM Source MUST set the value of this element to an absolute URI (conformant with  
552 RFC3986 [URI]) that uniquely identifies the offered Sequence.

553 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier/@{any}  
554 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
555 to the element.

556 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint  
557 An RM Source MUST include this element, of type `wsa:EndpointReferenceType` (as  
558 specified by WS-Addressing). This element specifies the endpoint reference to which Sequence  
559 Lifecycle Messages, Acknowledgement Requests, and fault messages related to the offered  
560 Sequence are to be sent.

561 Implementations MUST NOT use an endpoint reference in the Endpoint element that would  
562 prevent the sending of Sequence Lifecycle Message, etc. For example, using the WS-Addressing  
563 "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM Destination  
564 to ever send Sequence Lifecycle Messages (e.g. `TerminateSequence`) to the RM Source for  
565 the offered Sequence.

566 The offer of an Endpoint containing the "http://www.w3.org/2005/08/addressing/anonymous" IRI  
567 as its address is problematic due to the inability of a source to connect to this address and retry  
568 unacknowledged messages (as described in section 2.3). Note that this specification does not  
569 define any mechanisms for providing this assurance. In the absence of an extension that  
570 addresses this issue, an RM Destination MUST NOT accept (via the  
571 /wsmr:CreateSequenceResponse/wsmr:Accept element described below) an offer that  
572 contains the "http://www.w3.org/2005/08/addressing/anonymous" IRI as its address.

573 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires  
574 This element, if present, of type `xs:duration` specifies the duration for the offered Sequence. A  
575 value of "PT0S" indicates that the offered Sequence will never expire. Absence of the element  
576 indicates an implied value of "PT0S".

577 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires/@{any}  
578 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
579 to the element.

580 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior  
581 This element, if present, specifies the behavior that the destination will exhibit upon the closure or  
582 termination of an incomplete Sequence. For the purposes of defining the values used, the term  
583 "discard" refers to behavior equivalent to the Application Destination never processing a particular  
584 message.

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

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

591 The default value of “NoDiscard” indicates that no acknowledged messages in the Sequence will  
592 be discarded.

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

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

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

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

599 /wsmr:CreateSequence/{any}

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

602 /wsmr:CreateSequence/@{any}

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

605 A *CreateSequenceResponse* is sent in the body of a response message by an RM Destination in  
606 response to receipt of a *CreateSequence* request message. It carries the *Identifier* of the created  
607 Sequence and indicates that the RM Source can begin sending messages in the context of the identified  
608 Sequence.

609 The following exemplar defines the *CreateSequenceResponse* syntax:

```
610 <wsmr:CreateSequenceResponse ...>  
611   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
612   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
613   <wsmr:IncompleteSequenceBehavior>  
614     wsmr:IncompleteSequenceBehaviorType  
615   </wsmr:IncompleteSequenceBehavior> ?  
616   <wsmr:Accept ...>  
617     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
618     ...  
619   </wsmr:Accept> ?  
620   ...  
621 </wsmr:CreateSequenceResponse>
```

622 The following describes the content model of the *CreateSequenceResponse* element.

623 /wsmr:CreateSequenceResponse

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

627 /wsmr:CreateSequenceResponse/wsmr:Identifier

628 The RM Destination MUST include this element within any *CreateSequenceResponse*  
629 message it sends. The RM Destination MUST set the value of this element to the absolute URI  
630 (conformant with RFC3986) that uniquely identifies the Sequence that has been created by the  
631 RM Destination.

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

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

635 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

648 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

661 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

667 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

668 The RM Destination MUST include this element, of type `wsa:EndpointReferenceType` (as  
669 specified by WS-Addressing). It specifies the endpoint reference to which messages containing  
670 `SequenceAcknowledgement` header blocks and faults related to the created Sequence are to  
671 be sent, unless otherwise noted in this specification (for example, see section3.5).

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

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

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

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

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

682 /wsrm:CreateSequenceResponse/{any}

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

685 /wsrm:CreateSequenceResponse/@{any}

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

## 688 3.5 Closing A Sequence

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

694 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of  
695 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept  
696 any new messages for the specified Sequence, other than those already accepted at the time the  
697 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or  
698 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST  
699 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`  
700 element) header block on any messages associated with the Sequence destined to the RM Source,  
701 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM  
702 Source.

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

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

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

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

722 The following exemplar defines the `CloseSequence` syntax:

```
723 <wsrm:CloseSequence ...>  
724   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
725   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?
```

```
726     ...
727 </wsmr:CloseSequence>
```

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

729 `/wsmr:CloseSequence`

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

733 `/wsmr:CloseSequence/wsmr:Identifier`

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

737 `/wsmr:CloseSequence/wsmr:LastMessageNumber`

738 The RM Source SHOULD include this element in any `CloseSequence` message it sends. The  
739 `LastMsgNumber` element specifies the highest assigned message number of all the Sequence  
740 Traffic Messages for the closing Sequence.

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

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

744 `/wsmr:CloseSequence/{any}`

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

747 `/wsmr:CloseSequence/@{any}`

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

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

752 The following exemplar defines the `CloseSequenceResponse` syntax:

```
753 <wsmr:CloseSequenceResponse ...>
754   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
755   ...
756 </wsmr:CloseSequenceResponse>
```

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

758 `/wsmr:CloseSequenceResponse`

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

761 `/wsmr:CloseSequenceResponse/wsmr:Identifier`

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

765 `/wsmr:CloseSequenceResponse/wsmr:Identifier/@{any}`

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



768 /wsrm:CloseSequenceResponse/{any}

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

771 /wsrm:CloseSequenceResponse/@{any}

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

## 774 3.6 Sequence Termination

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

782 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM  
783 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.  
784 The RM Destination can use this information, for example, to implement the behavior indicated by  
785 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the  
786 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the  
787 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same  
788 Sequence.

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

793 The following exemplar defines the `TerminateSequence` syntax:

```
794 <wsrm:TerminateSequence ...>  
795   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
796   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
797   ...  
798 </wsrm:TerminateSequence>
```

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

800 /wsrm:TerminateSequence

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

806 This element MAY also be sent by the RM Destination to indicate that it has unilaterally  
807 terminated the Sequence. Upon sending this message the RM Destination MUST NOT accept  
808 any additional messages (with the exception of the corresponding  
809 `TerminateSequenceResponse`) for this Sequence. Upon receipt of a `TerminateSequence`  
810 the RM Source MUST NOT send any additional messages (with the exception of the  
811 corresponding `TerminateSequenceResponse`) for this Sequence.

812 /wsrm:TerminateSequence/wsrm:Identifier

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

816 `/wsm:TerminateSequence/wsm:LastMsgNumber`

817 The RM Source SHOULD include this element in any `TerminateSequence` message it sends.  
818 The `LastMsgNumber` element specifies the highest assigned message number of all the  
819 Sequence Traffic Messages for the terminating Sequence.

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

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

823 `/wsm:TerminateSequence/{any}`

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

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

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

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

831 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
832 <wsm:TerminateSequenceResponse ...>  
833   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
834   ...  
835 </wsm:TerminateSequenceResponse>
```

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

837 `/wsm:TerminateSequenceResponse`

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

841 `/wsm:TerminateSequenceResponse/wsm:Identifier`

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

845 `/wsm:TerminateSequenceResponse/wsm:Identifier/@{any}`

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

848 `/wsm:TerminateSequenceResponse/{any}`

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

851 `/wsm:TerminateSequenceResponse/@{any}`

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

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

## 857 3.7 Sequences

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

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

865 A following exemplar defines its syntax:

```
866 <wsm:Sequence ...>  
867   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
868   <wsm:MessageNumber> wsm:MessageNumberType </wsm:MessageNumber>  
869   ...  
870 </wsm:Sequence>
```

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

### 872 /wsm:Sequence

873 This protocol element associates the message in which it is contained with a previously  
874 established RM Sequence. It contains the Sequence's unique `Identifier` and the containing  
875 message's ordinal position within that Sequence. The RM Destination MUST understand the  
876 `Sequence` header block. The RM Source MUST assign a `mustUnderstand` attribute with a  
877 value 1/true (from the namespace corresponding to the version of SOAP to which the `Sequence`  
878 SOAP header block is bound) to the `Sequence` header block element.

### 879 /wsm:Sequence/wsm:Identifier

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

### 883 /wsm:Sequence/wsm:Identifier/@{any}

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

### 886 /wsm:Sequence/wsm:MessageNumber

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

### 891 /wsm:Sequence/{any}

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

### 894 /wsm:Sequence/@{any}

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

897 The following example illustrates a Sequence header block.

```
898 <wsrm:Sequence>
899   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
900   <wsrm:MessageNumber>10</wsrm:MessageNumber>
901 </wsrm:Sequence>
```

## 902 3.8 Request Acknowledgement

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

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

912 An RM Destination that Receives a message that contains an AckRequested header block MUST send  
913 a message containing a SequenceAcknowledgement header block to the AcksTo endpoint reference  
914 (see section 3.4) for a known Sequence or else generate an UnknownSequence fault. It is  
915 RECOMMENDED that the RM Destination return a AcknowledgementRange or None element instead  
916 of a Nack element (see section 3.9).

917 The following exemplar defines its syntax:

```
918 <wsrm:AckRequested ...>
919   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
920   ...
921 </wsrm:AckRequested>
```

922 The following describes the content model of the AckRequested header block.

923 /wsrm:AckRequested

924     This element requests an Acknowledgement for the identified Sequence.

925 /wsrm:AckRequested/wsrm:Identifier

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

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

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

933 /wsrm:AckRequested/{any}

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

936 /wsrm:AckRequested/@{any}

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

## 939 3.9 Sequence Acknowledgement

940 The RM Destination informs the RM Source of successful message receipt using a  
941 `SequenceAcknowledgement` header block. Acknowledgements can be explicitly requested using the  
942 `AckRequested` directive (see section 3.8).

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

950 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the  
951 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing  
952 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any  
953 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted  
954 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received  
955 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`  
956 header block for that same Sequence Identifier. When the RM Destination receives an  
957 `AckRequested` header, and the `AcksTo` EPR for that Sequence is the WS-Addressing anonymous IRI,  
958 the RM Destination SHOULD respond on the protocol binding-specific back-channel provided by the  
959 Received message containing the `AckRequested` header block.

960 The following exemplar defines its syntax:

```
961 <wsm:SequenceAcknowledgement ...>  
962   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
963   [ [ [ <wsm:AcknowledgementRange ...  
964     Upper="wsm:MessageNumberType"  
965     Lower="wsm:MessageNumberType"/> +  
966     | <wsm:None/> ]  
967     <wsm:Final/> ? ]  
968     | <wsm:Nack> wsm:MessageNumberType </wsm:Nack> + ]  
969   ...  
970 </wsm:SequenceAcknowledgement>
```

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

973 `/wsm:SequenceAcknowledgement`

974       This element contains the Sequence Acknowledgement information.

975 `/wsm:SequenceAcknowledgement/wsm:Identifier`

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

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

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

984 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange`

- 985 The RM Destination MAY include one or more instances of this element within a  
 986 `SequenceAcknowledgement` header block. It contains a range of Sequence message numbers  
 987 successfully accepted by the RM Destination. The ranges MUST NOT overlap. The RM  
 988 Destination MUST NOT include this element if a sibling `Nack` or `None` element is also present as  
 989 a child of `SequenceAcknowledgement`.
- 990 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper`  
 991 The RM Destination MUST set the value of this attribute equal to the message number of the  
 992 highest contiguous message in a Sequence range accepted by the RM Destination.
- 993 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower`  
 994 The RM Destination MUST set the value of this attribute equal to the message number of the  
 995 lowest contiguous message in a Sequence range accepted by the RM Destination.
- 996 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}`  
 997 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
 998 to the element.
- 999 `/wsrm:SequenceAcknowledgement/wsrm:None`  
 1000 The RM Destination MUST include this element within a `SequenceAcknowledgement` header  
 1001 block if the RM Destination has not accepted any messages for the specified Sequence. The RM  
 1002 Destination MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack`  
 1003 element is also present as a child of the `SequenceAcknowledgement`.
- 1004 `/wsrm:SequenceAcknowledgement/wsrm:Final`  
 1005 The RM Destination MAY include this element within a `SequenceAcknowledgement` header  
 1006 block. This element indicates that the RM Destination is not receiving new messages for the  
 1007 specified Sequence. The RM Source can be assured that the ranges of messages acknowledged  
 1008 by this `SequenceAcknowledgement` header block will not change in the future. The RM  
 1009 Destination MUST include this element when the Sequence is closed. The RM Destination MUST  
 1010 NOT include this element when sending a `Nack`; it can only be used when sending  
 1011 `AcknowledgementRange` elements or a `None`.
- 1012 `/wsrm:SequenceAcknowledgement/wsrm:Nack`  
 1013 The RM Destination MAY include this element within a `SequenceAcknowledgement` header  
 1014 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`  
 1015 representing the `MessageNumber` of an unreceived message in a Sequence. The RM Destination  
 1016 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is  
 1017 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM  
 1018 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT  
 1019 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously  
 1020 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a  
 1021 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been  
 1022 acknowledged within an `AcknowledgementRange`.
- 1023 `/wsrm:SequenceAcknowledgement/{any}`  
 1024 This is an extensibility mechanism to allow different (extensible) types of information, based on a  
 1025 schema, to be passed.
- 1026 `/wsrm:SequenceAcknowledgement/@{any}`  
 1027 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
 1028 to the element.

1029 The following examples illustrate `SequenceAcknowledgement` elements:

- 1030 • Message numbers 1..10 inclusive in a Sequence have been accepted by the RM Destination.

```
1031 <wsrm:SequenceAcknowledgement>  
1032   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1033   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
1034 </wsrm:SequenceAcknowledgement>
```

- 1035 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM  
1036 Destination, messages 3 and 7 have not been accepted.

```
1037 <wsrm:SequenceAcknowledgement>  
1038   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1039   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1040   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1041   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1042 </wsrm:SequenceAcknowledgement>
```

- 1043 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1044 <wsrm:SequenceAcknowledgement>  
1045   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1046   <wsrm:Nack>3</wsrm:Nack>  
1047 </wsrm:SequenceAcknowledgement>
```



## 1048 4 Faults

1049 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. `Create`  
1050 `Sequence Refused` is a possible fault reply for this operation. `Unknown Sequence` is a fault generated by  
1051 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences  
1052 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM  
1053 on a Received message that did not use the protocol. All other faults in this section relate to known  
1054 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those faults.  
1055 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

1056 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault  
1057 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1058 `http://docs.oasis-open.org/ws-rx/wsr/200702/fault`

1059 The faults defined in this section are generated if the condition stated in the preamble is met. Fault  
1060 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1061 The definitions of faults use the following properties:

1062 [Code] The fault code.

1063 [Subcode] The fault subcode.

1064 [Reason] The English language reason element.

1065 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail  
1066 element is defined for a fault, implementations MUST include the elements in the order that they are  
1067 specified.

1068 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or  
1069 "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

1070 The properties above bind to a SOAP 1.2 fault as follows:

```
1071 <S:Envelope>
1072   <S:Header>
1073     <wsa:Action>
1074       http://docs.oasis-open.org/ws-rx/wsr/200702/fault
1075     </wsa:Action>
1076     <!-- Headers elided for brevity. -->
1077   </S:Header>
1078   <S:Body>
1079     <S:Fault>
1080       <S:Code>
1081         <S:Value> [Code] </S:Value>
1082         <S:Subcode>
1083           <S:Value> [Subcode] </S:Value>
1084         </S:Subcode>
1085       </S:Code>
1086       <S:Reason>
1087         <S:Text xml:lang="en"> [Reason] </S:Text>
1088       </S:Reason>
1089       <S:Detail>
1090         [Detail]
```

1091  
1092  
1093  
1094  
1095

```
...  
</S:Detail>  
</S:Fault>  
</S:Body>  
</S:Envelope>
```

1096 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM  
1097 header block:

1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113

```
<S11:Envelope>  
<S11:Header>  
  <wsrm:SequenceFault>  
    <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>  
    <wsrm:Detail> [Detail] </wsrm:Detail>  
    ...  
  </wsrm:SequenceFault>  
  <!-- Headers elided for brevity. -->  
</S11:Header>  
<S11:Body>  
  <S11:Fault>  
    <faultcode> [Code] </faultcode>  
    <faultstring> [Reason] </faultstring>  
  </S11:Fault>  
</S11:Body>  
</S11:Envelope>
```

1114 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a  
1115 CreateSequence request message:

1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123

```
<S11:Envelope>  
<S11:Body>  
  <S11:Fault>  
    <faultcode> [Subcode] </faultcode>  
    <faultstring> [Reason] </faultstring>  
  </S11:Fault>  
</S11:Body>  
</S11:Envelope>
```

## 1124 4.1 SequenceFault Element

1125 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the  
1126 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging  
1127 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.  
1128 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the  
1129 SOAP 1.2 binding.

1130 The following exemplar defines its syntax:

1131  
1132  
1133  
1134  
1135

```
<wsrm:SequenceFault ...>  
  <wsrm:FaultCode> wsrm:FaultCode </wsrm:FaultCode>  
  <wsrm:Detail> ... </wsrm:Detail> ?  
  ...  
</wsrm:SequenceFault>
```

1136 The following describes the content model of the `SequenceFault` element.

1137 /wsrm:SequenceFault

1138       This is the element containing Sequence fault information for WS-ReliableMessaging

1139 /wsrm:SequenceFault/wsrm:FaultCode

- 1140 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this  
 1141 element to a qualified name from the set of faults [Subcodes] defined below.
- 1142 `/wsrm:SequenceFault/wsrm:Detail`  
 1143 This element, if present, carries application specific error information related to the fault being  
 1144 described.
- 1145 `/wsrm:SequenceFault/wsrm:Detail/{any}`  
 1146 The application specific error information related to the fault being described.
- 1147 `/wsrm:SequenceFault/wsrm:Detail/@{any}`  
 1148 The application specific error information related to the fault being described.
- 1149 `/wsrm:SequenceFault/{any}`  
 1150 This is an extensibility mechanism to allow different (extensible) types of information, based on a  
 1151 schema, to be passed.
- 1152 `/wsrm:SequenceFault/@{any}`  
 1153 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
 1154 to the element.

## 1155 4.2 Sequence Terminated

- 1156 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding  
 1157 Endpoint of this decision.
- 1158 Properties:
- 1159 [Code] Sender or Receiver
- 1160 [Subcode] `wrm:SequenceTerminated`
- 1161 [Reason] The Sequence has been terminated due to an unrecoverable error.
- 1162 [Detail]

1163 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

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

## 1164 4.3 Unknown Sequence

- 1165 Properties:
- 1166 [Code] Sender
- 1167 [Subcode] `wrm:UnknownSequence`

1168 [Reason] The value of `wsrc:Identifier` is not a known Sequence identifier.

1169 [Detail]

```
<wsrc:Identifier ...> xs:anyURI </wsrc: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.

#### 1171 4.4 Invalid Acknowledgement

1172 An example of when this fault is generated is when a message is Received by the RM Source containing  
1173 a `SequenceAcknowledgement` covering messages that have not been sent.

1174 [Code] Sender

1175 [Subcode] `wsrc:InvalidAcknowledgement`

1176 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1177 [Detail]

```
<wsrc:SequenceAcknowledgement ...> ... </wsrc:SequenceAcknowledgement>
```

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

#### 1179 4.5 Message Number Rollover

1180 If the condition listed below is reached, the RM Destination MUST generate this fault.

1181 Properties:

1182 [Code] Sender

1183 [Subcode] wsrn:MessageNumberRollover

1184 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1185 [Detail]

1186 `<wsrn:Identifier ...> xs:anyURI </wsrn:Identifier>`

1187 `<wsrn:MaxMessageNumber> wsrn:MessageNumberType </wsrn:MaxMessageNumber>`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	Message number in /wsrn:Sequence/wsrn: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.

## 1188 4.6 Create Sequence Refused

1189 Properties:

1190 [Code] Sender or Receiver

1191 [Subcode] wsrn:CreateSequenceRefused

1192 [Reason] The Create Sequence request has been refused by the RM Destination.

1193 [Detail]

1194 `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.

## 1195 4.7 Sequence Closed

1196 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1197 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that is closed.

1199 Properties:

1200 [Code] Sender

1201 [Subcode] wsrn:SequenceClosed

1202 [Reason] The Sequence is closed and cannot accept new messages.

1203 [Detail]

1204 `<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.

## 1205 **4.8 WSRM Required**

1206 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming  
1207 message that did not use this protocol.

1208 Properties:

1209 [Code] Sender

1210 [Subcode] wsrn:WSRMRequired

1211 [Reason] The RM Destination requires the use of WSRM.

1212 [Detail]

1213 `xs:any`

---

## 1214 5 Security Threats and Countermeasures

1215 This specification considers two sets of security requirements, those of the applications that use the WS-  
1216 RM protocol and those of the protocol itself.

1217 This specification makes no assumptions about the security requirements of the applications that use WS-  
1218 RM. However, once those requirements have been satisfied within a given operational context, the  
1219 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;  
1220 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1221 There are many other security concerns that one may need to consider when implementing or using this  
1222 protocol. The material below should not be considered as a "check list". Implementers and users of this  
1223 protocol are urged to perform a security analysis to determine their particular threat profile and the  
1224 appropriate responses to those threats.

1225 Implementers are also advised that there is a core tension between security and reliable messaging that  
1226 can be problematic if not addressed by implementations; one aspect of security is to prevent message  
1227 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.  
1228 Consequently, if the security sub-system processes a message but a failure occurs before the reliable  
1229 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system  
1230 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-  
1231 system will likely continue to expect and even solicit the missing message(s). Care should be taken to  
1232 avoid and prevent this condition.

### 1233 5.1 Threats and Countermeasures

1234 The primary security requirement of this protocol is to protect the specified semantics and protocol  
1235 invariants against various threats. The following sections describe several threats to the integrity and  
1236 operation of this protocol and provide some general outlines of countermeasures to those threats.  
1237 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable  
1238 to all operational contexts.

#### 1239 5.1.1 Integrity Threats

1240 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic  
1241 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or  
1242 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block  
1243 to its intended message represents a threat to the WS-RM protocol.

1244 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM  
1245 Source and RM Destination then they have undermined the implementation's ability to guarantee the first  
1246 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be  
1247 Delivered to the Application Destination in the same order that they were sent by the Application Source.

##### 1248 5.1.1.1 Countermeasures

1249 Integrity threats are generally countered via the use of digital signatures some level of the communication  
1250 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include  
1251 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers  
1252 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in  
1253 which they occur, implementations MUST allow for signatures that cover only these headers.

## 1254 **5.1.2 Resource Consumption Threats**

1255 The creation of a Sequence with an RM Destination consumes various resources on the systems used to  
1256 implement that RM Destination. These resources can include network connections, database tables,  
1257 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM  
1258 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM  
1259 Destination. Another attack is to create a Sequence for a service that is known to require in-order  
1260 message Delivery and use this Sequence to send a stream of very large messages to that service, making  
1261 sure to omit message number “1” from that stream.

### 1262 **5.1.2.1 Countermeasures**

1263 There are a number of countermeasures against the described resource consumption threats. The  
1264 technique advocated by this specification is for the RM Destination to restrict the ability to create a  
1265 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in  
1266 some cases, allows the identity of any attackers to be determined.

1267 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and  
1268 authenticate the RM Source that issued the `CreateSequence` message.

## 1269 **5.1.3 Sequence Spoofing Threats**

1270 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a  
1271 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a  
1272 fake `TerminateSequence` message that references the target Sequence and sends this message to the  
1273 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the  
1274 current `MessageNumber` for their target Sequence.

1275 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP  
1276 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence  
1277 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to  
1278 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a  
1279 message that it sends to the `AcksTo` EPR of an RM Source.

### 1280 **5.1.3.1 Sequence Hijacking**

1281 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject  
1282 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those  
1283 messages.

1284 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a  
1285 Sequence may be bound to some form of a security session in order to counter the threats described in  
1286 this section, applications **MUST NOT** rely on WS-RM-related information to make determinations about  
1287 the identity of the entity that created a message; applications **SHOULD** rely only upon information that is  
1288 established by the security infrastructure to make such determinations. Failure to observe this rule  
1289 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of  
1290 the ability to authenticate its peers even though the necessary security processing has taken place.

### 1291 **5.1.3.2 Countermeasures**

1292 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by  
1293 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM  
1294 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination  
1295 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter



1296 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it  
1297 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the  
1298 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it  
1299 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the  
1300 referenced Sequence.

1301 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and  
1302 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify  
1303 its Sequence peer it MUST be able to identify and authenticate the entity that sent the  
1304 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a  
1305 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that  
1306 message and, if necessary, correlate this identity with the Sequence peer identity established at  
1307 Sequence creation time.

## 1308 **5.2 Security Solutions and Technologies**

1309 The security threats described in the previous sections are neither new nor unique. The solutions that  
1310 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This  
1311 section maps the facilities provided by common web services security solutions against countermeasures  
1312 described in the previous sections.

1313 Before continuing this discussion, however, some examination of the underlying requirements of the  
1314 previously described countermeasures is necessary. Specifically it should be noted that the technique  
1315 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates  
1316 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization  
1317 check against this authenticated identity and determines if the RM Source is permitted to create  
1318 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime  
1319 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any  
1320 discussion of such facilities is considered to be beyond the scope of this specification.

### 1321 **5.2.1 Transport Layer Security**

1322 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the  
1323 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints  
1324 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1325 The description provided here is general in nature and is not intended to serve as a complete definition on  
1326 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the  
1327 choice of features as well as the manner in which they will be used. The mechanisms described in the  
1328 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the  
1329 requirements and constraints of the use of SSL/TLS.

#### 1330 **5.2.1.1 Model**

1331 The basic model for using SSL/TLS is as follows:

- 1332 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1333 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM  
1334 Destination.
- 1335 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an  
1336 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a  
1337 synchronous `CreateSequenceResponse` using the session established in (1).

- 1338 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit  
1339 any and all messages or faults that refer to that Sequence.
- 1340 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established  
1341 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous  
1342 exchanges, the RM Destination uses the SSL/TLS session established in (1).

### 1343 5.2.1.2 Countermeasure Implementation

1344 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the  
1345 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the  
1346 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If  
1347 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and  
1348 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1349 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification  
1350 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods  
1351 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS  
1352 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1353 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP  
1354 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the  
1355 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party  
1356 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself  
1357 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.  
1358 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an  
1359 Acknowledgement) using BasicAuth.
- 1360 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the  
1361 connection authenticates itself to the party accepting the connection using an X.509 certificate  
1362 that is exchanged during the SSL/TLS handshake.

1363 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself  
1364 using one the above mechanisms. The authenticated identity can then be used to determine if the RM  
1365 Source is authorized to create a Sequence with the RM Destination.

1366 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1367 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the  
1368 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than  
1369 on authentication information. For example, an RM Destination can determine that a Sequence Traffic  
1370 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS  
1371 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a  
1372 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a  
1373 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used  
1374 to protect that Sequence.

1375 This specification does not preclude the use of other methods of using SSL/TLS to implement the  
1376 countermeasures (such as associating specific authentication information with a Sequence) although such  
1377 methods are not covered by this document.

1378 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS  
1379 session) are outside the scope of this specification.

### 1380 5.2.2 SOAP Message Security

1381 The mechanisms described in WS-Security may be used in various ways to implement the  
1382 countermeasures described in the previous sections. This specification advocates using the protocol  
1383 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1384 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component  
1385 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1386 The description provided here is general in nature and is not intended to serve as a complete definition on  
1387 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations  
1388 need to agree on the choice of features as well as the manner in which they will be used. The  
1389 mechanisms described in the Web Services Security Policy Language MAY be used by services to  
1390 describe the requirements and constraints of the use of WS-SecureConversation.

### 1391 **5.2.2.1 Model**

1392 The basic model for using WS-SecureConversation is as follows:

- 1393       1    The RM Source and the RM Destination create a WS-SecureConversation security context. This  
1394            may involve the participation of third parties such as a security token service. The tokens  
1395            exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service  
1396            tickets).
- 1397       2    During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security  
1398            context that will be used to protect the Sequence. This is done so that, in cases where the  
1399            `CreateSequence` message is signed by more than one security context, the RM Source can  
1400            indicate which security context should be used to protect the newly created Sequence.
- 1401       3    For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)  
1402            associated with the security context to sign (as defined by WS-Security) at least the body and  
1403            any relevant WS-RM-defined headers of any and all messages or faults that refer to that  
1404            Sequence.

### 1405 **5.2.2.2 Countermeasure Implementation**

1406 Without relying upon any authentication information, the per-message signatures provide the necessary  
1407 integrity qualities to counter the threats described in section 5.1.1.

1408 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of  
1409 authentication claims must be provided by the RM Source to the RM Destination during the establishment  
1410 of the Security Context. These claims can then be used to determine if the RM Source is authorized to  
1411 create a Sequence with the RM Destination.

1412 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1413 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the  
1414 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security  
1415 context rather than on any authentication claims that may have been established during security context  
1416 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures  
1417 (such as associating specific authentication claims to a Sequence) are possible but not covered by this  
1418 document.

1419 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits  
1420 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the  
1421 association between a Sequence and its protecting security context cannot always be established  
1422 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and  
1423 `CreateSequenceResponse` messages may be signed by more than one security context.

1424 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as  
1425 amending or renewing contexts) are outside the scope of this specification.

---

## 1426 6 Securing Sequences

1427 As noted in section 5, the RM Source and RM Destination should be able to protect their shared  
1428 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of  
1429 achieving this objective depending upon the underlying security infrastructure.

### 1430 6.1 Securing Sequences Using WS-Security

1431 One mechanism for protecting a Sequence is to include a security token using a  
1432 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-  
1433 SecureConversation) in the `CreateSequence` element. This establishes an association between the  
1434 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source  
1435 and Destination MUST use the security token as the basis for authorization of all subsequent interactions  
1436 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as  
1437 there may be more than one token on a `CreateSequence` message or inferred from the communication  
1438 context (e.g. transport protection).

1439 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,  
1440 if the token being referenced supports such mechanism.

1441 The following exemplar defines the `CreateSequence` syntax when extended to include a  
1442 `wsse:SecurityTokenReference`:

```
1443 <wsmr:CreateSequence ...>  
1444   <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
1445   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
1446   <wsmr:Offer ...>  
1447     <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
1448     <wsmr:Endpoint> wsa:EndpointReferenceType </wsmr:Endpoint>  
1449     <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
1450     <wsmr:IncompleteSequenceBehavior>  
1451       wsmr:IncompleteSequenceBehaviorType  
1452     </wsmr:IncompleteSequenceBehavior> ?  
1453     ...  
1454   </wsmr:Offer> ?  
1455   ...  
1456   <wsse:SecurityTokenReference>  
1457     ...  
1458   </wsse:SecurityTokenReference> ?  
1459   ...  
1460 </wsmr:CreateSequence>
```

1461 The following describes the content model of the additional `CreateSequence` elements.

#### 1462 `/wsmr:CreateSequence/wsse:SecurityTokenReference`

1463 This element uses the extensibility mechanism defined for the `CreateSequence` element  
1464 (defined in section 3.4) to communicate an explicit reference to the security token, using a  
1465 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and  
1466 Destination MUST use to authorize messages for the created (and, if present, the offered)  
1467 Sequence(s). All subsequent messages related to the created (and, if present, the offered)  
1468 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token  
1469 (e.g., by using or deriving from a private or secret key).

1470 When a RM Source transmits a `CreateSequence` that has been extended to include a  
1471 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1472 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include  
1473 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This  
1474 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source  
1475 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands  
1476 and conforms with the requirements listed above. Note that an RM Destination understanding this header  
1477 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined  
1478 in WS-Security still applies.

1479 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1480 <wsrm:UsesSequenceSTR ... />
```

1481 The following describes the content model of the `UsesSequenceSTR` header block.

1482 `/wsrm:UsesSequenceSTR`

1483 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that  
1484 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`  
1485 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly  
1486 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus  
1487 aborting the requested Sequence creation.

1488 The following is an example of a `CreateSequence` message using the

1489 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1490 <soap:Envelope ...>  
1491   <soap:Header>  
1492     ...  
1493     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1494     ...  
1495   </soap:Header>  
1496   <soap:Body>  
1497     <wsrm:CreateSequence>  
1498       <wsrm:AcksTo>  
1499         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1500       </wsrm:AcksTo>  
1501       <wsse:SecurityTokenReference>  
1502         ...  
1503       </wsse:SecurityTokenReference>  
1504     </wsrm:CreateSequence>  
1505   </soap:Body>  
1506 </soap:Envelope>
```

## 1507 6.2 Securing Sequences Using SSL/TLS

1508 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).  
1509 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying  
1510 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a  
1511 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a  
1512 SOAP header block within the `CreateSequence` message.

1513 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1514 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1515 The following describes the content model of the `UsesSequenceSSL` header block.

1516 `/wsrm:UsesSequenceSSL`

1517 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`  
1518 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1519           SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM  
1520           Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.  
1521           The receiving RM Destination MUST understand and correctly implement the functionality  
1522           described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the  
1523           requested Sequence creation.

1524           Note that the inclusion of the above header by the RM Source implies that all Sequence-related  
1525           information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from  
1526           the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the  
1527           `CreateSequenceResponse` message.



---

## 1528 Appendix A. Schema

1529 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-  
1530 Schema Part2] is located at:

1531 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1532 The following copy is provided for reference.

```
1533 <?xml version="1.0" encoding="UTF-8"?>
1534 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1535         OASIS trademark, IPR and other policies apply. -->
1536 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1537           xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1538 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1539 rx/wsrn/200702" elementFormDefault="qualified"
1540 attributeFormDefault="unqualified">
1541   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1542     schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1543   <!-- Protocol Elements -->
1544   <xs:complexType name="SequenceType">
1545     <xs:sequence>
1546       <xs:element ref="wsrm:Identifier"/>
1547       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1548       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1549 maxOccurs="unbounded"/>
1550     </xs:sequence>
1551     <xs:anyAttribute namespace="##other" processContents="lax"/>
1552   </xs:complexType>
1553   <xs:element name="Sequence" type="wsrm:SequenceType"/>
1554   <xs:element name="SequenceAcknowledgement">
1555     <xs:complexType>
1556       <xs:sequence>
1557         <xs:element ref="wsrm:Identifier"/>
1558         <xs:choice>
1559           <xs:sequence>
1560             <xs:choice>
1561               <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1562                 <xs:complexType>
1563                   <xs:sequence/>
1564                   <xs:attribute name="Upper" type="xs:unsignedLong"
1565 use="required"/>
1566                   <xs:attribute name="Lower" type="xs:unsignedLong"
1567 use="required"/>
1568                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1569               </xs:complexType>
1570             </xs:choice>
1571             <xs:element name="None">
1572               <xs:complexType>
1573                 <xs:sequence/>
1574               </xs:complexType>
1575             </xs:element>
1576           </xs:choice>
1577           <xs:element name="Final" minOccurs="0">
1578             <xs:complexType>
1579               <xs:sequence/>
1580             </xs:complexType>
1581           </xs:element>
1582         </xs:sequence>
1583       <xs:element name="Nack" type="xs:unsignedLong">
```

```

1584 maxOccurs="unbounded"/>
1585     </xs:choice>
1586     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1587 maxOccurs="unbounded"/>
1588     </xs:sequence>
1589     <xs:anyAttribute namespace="##other" processContents="lax"/>
1590 </xs:complexType>
1591 </xs:element>
1592 <xs:complexType name="AckRequestedType">
1593     <xs:sequence>
1594         <xs:element ref="wsrm:Identifier"/>
1595         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1596 maxOccurs="unbounded"/>
1597     </xs:sequence>
1598     <xs:anyAttribute namespace="##other" processContents="lax"/>
1599 </xs:complexType>
1600 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1601 <xs:element name="Identifier">
1602     <xs:complexType>
1603         <xs:annotation>
1604             <xs:documentation>
1605                 This type is for elements whose [children] is an anyURI and can have
1606 arbitrary attributes.
1607             </xs:documentation>
1608         </xs:annotation>
1609         <xs:simpleContent>
1610             <xs:extension base="xs:anyURI">
1611                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1612             </xs:extension>
1613         </xs:simpleContent>
1614     </xs:complexType>
1615 </xs:element>
1616 <xs:element name="Address">
1617     <xs:complexType>
1618         <xs:simpleContent>
1619             <xs:extension base="xs:anyURI">
1620                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1621             </xs:extension>
1622         </xs:simpleContent>
1623     </xs:complexType>
1624 </xs:element>
1625 <xs:simpleType name="MessageNumberType">
1626     <xs:restriction base="xs:unsignedLong">
1627         <xs:minInclusive value="1"/>
1628         <xs:maxInclusive value="9223372036854775807"/>
1629     </xs:restriction>
1630 </xs:simpleType>
1631 <!-- Fault Container and Codes -->
1632 <xs:simpleType name="FaultCodes">
1633     <xs:restriction base="xs:QName">
1634         <xs:enumeration value="wsrm:SequenceTerminated"/>
1635         <xs:enumeration value="wsrm:UnknownSequence"/>
1636         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1637         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1638         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1639         <xs:enumeration value="wsrm:SequenceClosed"/>
1640         <xs:enumeration value="wsrm:WSRMRequired"/>
1641     </xs:restriction>
1642 </xs:simpleType>
1643 <xs:complexType name="SequenceFaultType">
1644     <xs:sequence>
1645         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1646         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1647         <xs:any namespace="##other" processContents="lax" minOccurs="0"

```



```

1648 maxOccurs="unbounded"/>
1649 </xs:sequence>
1650 <xs:anyAttribute namespace="##other" processContents="lax"/>
1651 </xs:complexType>
1652 <xs:complexType name="DetailType">
1653 <xs:sequence>
1654 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1655 maxOccurs="unbounded"/>
1656 </xs:sequence>
1657 <xs:anyAttribute namespace="##other" processContents="lax"/>
1658 </xs:complexType>
1659 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1660 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1661 <xs:element name="CreateSequenceResponse"
1662 type="wsrm:CreateSequenceResponseType"/>
1663 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1664 <xs:element name="CloseSequenceResponse"
1665 type="wsrm:CloseSequenceResponseType"/>
1666 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1667 <xs:element name="TerminateSequenceResponse"
1668 type="wsrm:TerminateSequenceResponseType"/>
1669 <xs:complexType name="CreateSequenceType">
1670 <xs:sequence>
1671 <xs:element ref="wsrm:AcksTo"/>
1672 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1673 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1674 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1675 maxOccurs="unbounded">
1676 <xs:annotation>
1677 <xs:documentation>
1678 It is the authors intent that this extensibility be used to
1679 transfer a Security Token Reference as defined in WS-Security.
1680 </xs:documentation>
1681 </xs:annotation>
1682 </xs:any>
1683 </xs:sequence>
1684 <xs:anyAttribute namespace="##other" processContents="lax"/>
1685 </xs:complexType>
1686 <xs:complexType name="CreateSequenceResponseType">
1687 <xs:sequence>
1688 <xs:element ref="wsrm:Identifier"/>
1689 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1690 <xs:element name="IncompleteSequenceBehavior"
1691 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1692 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1693 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1694 maxOccurs="unbounded"/>
1695 </xs:sequence>
1696 <xs:anyAttribute namespace="##other" processContents="lax"/>
1697 </xs:complexType>
1698 <xs:complexType name="CloseSequenceType">
1699 <xs:sequence>
1700 <xs:element ref="wsrm:Identifier"/>
1701 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1702 minOccurs="0"/>
1703 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1704 maxOccurs="unbounded"/>
1705 </xs:sequence>
1706 <xs:anyAttribute namespace="##other" processContents="lax"/>
1707 </xs:complexType>
1708 <xs:complexType name="CloseSequenceResponseType">
1709 <xs:sequence>
1710 <xs:element ref="wsrm:Identifier"/>
1711 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1712 maxOccurs="unbounded"/>
1713 </xs:sequence>
1714 <xs:anyAttribute namespace="##other" processContents="lax"/>
1715 </xs:complexType>
1716 <xs:complexType name="TerminateSequenceType">
1717 <xs:sequence>
1718 <xs:element ref="wsrm:Identifier"/>
1719 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1720 minOccurs="0"/>
1721 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1722 maxOccurs="unbounded"/>
1723 </xs:sequence>
1724 <xs:anyAttribute namespace="##other" processContents="lax"/>
1725 </xs:complexType>
1726 <xs:complexType name="TerminateSequenceResponseType">
1727 <xs:sequence>
1728 <xs:element ref="wsrm:Identifier"/>
1729 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1730 maxOccurs="unbounded"/>
1731 </xs:sequence>
1732 <xs:anyAttribute namespace="##other" processContents="lax"/>
1733 </xs:complexType>
1734 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1735 <xs:complexType name="OfferType">
1736 <xs:sequence>
1737 <xs:element ref="wsrm:Identifier"/>
1738 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1739 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1740 <xs:element name="IncompleteSequenceBehavior"
1741 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1742 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1743 maxOccurs="unbounded"/>
1744 </xs:sequence>
1745 <xs:anyAttribute namespace="##other" processContents="lax"/>
1746 </xs:complexType>
1747 <xs:complexType name="AcceptType">
1748 <xs:sequence>
1749 <xs:element ref="wsrm:AcksTo"/>
1750 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1751 maxOccurs="unbounded"/>
1752 </xs:sequence>
1753 <xs:anyAttribute namespace="##other" processContents="lax"/>
1754 </xs:complexType>
1755 <xs:element name="Expires">
1756 <xs:complexType>
1757 <xs:simpleContent>
1758 <xs:extension base="xs:duration">
1759 <xs:anyAttribute namespace="##other" processContents="lax"/>
1760 </xs:extension>
1761 </xs:simpleContent>
1762 </xs:complexType>
1763 </xs:element>
1764 <xs:simpleType name="IncompleteSequenceBehaviorType">
1765 <xs:restriction base="xs:string">
1766 <xs:enumeration value="DiscardEntireSequence"/>
1767 <xs:enumeration value="DiscardFollowingFirstGap"/>
1768 <xs:enumeration value="NoDiscard"/>
1769 </xs:restriction>
1770 </xs:simpleType>
1771 <xs:element name="UsesSequenceSTR">
1772 <xs:complexType>
1773 <xs:sequence/>
1774 <xs:anyAttribute namespace="##other" processContents="lax"/>
1775 </xs:complexType>

```

```
1776 </xs:element>
1777 <xs:element name="UsesSequenceSSL">
1778   <xs:complexType>
1779     <xs:sequence/>
1780     <xs:anyAttribute namespace="##other" processContents="lax"/>
1781   </xs:complexType>
1782 </xs:element>
1783 <xs:element name="UnsupportedElement">
1784   <xs:simpleType>
1785     <xs:restriction base="xs:QName"/>
1786   </xs:simpleType>
1787 </xs:element>
1788 </xs:schema>
```

---

## 1789 Appendix B. WSDL

1790 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where  
1791 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be  
1792 present in exchanges with that endpoint.

1793 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not  
1794 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]  
1795 for a higher-level mechanism to indicate that WS-RM is engaged.

1796 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1797 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

1798 The following non-normative copy is provided for reference.

```
1799 <?xml version="1.0" encoding="utf-8"?>
1800 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1801 OASIS trademark, IPR and other policies apply. -->
1802 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1803 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1804 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1805 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1806 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1807 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL"
1808 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL">
1809
1810   <wSDL:types>
1811     <xs:schema
1812       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1813       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1814       200702.xsd"/>
1815     </xs:schema>
1816   </wSDL:types>
1817
1818   <wSDL:message name="CreateSequence">
1819     <wSDL:part name="create" element="rm:CreateSequence"/>
1820   </wSDL:message>
1821   <wSDL:message name="CreateSequenceResponse">
1822     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1823   </wSDL:message>
1824   <wSDL:message name="CloseSequence">
1825     <wSDL:part name="close" element="rm:CloseSequence"/>
1826   </wSDL:message>
1827   <wSDL:message name="CloseSequenceResponse">
1828     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1829   </wSDL:message>
1830   <wSDL:message name="TerminateSequence">
1831     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1832   </wSDL:message>
1833   <wSDL:message name="TerminateSequenceResponse">
1834     <wSDL:part name="terminateResponse"
1835     element="rm:TerminateSequenceResponse"/>
1836   </wSDL:message>
1837
1838   <wSDL:portType name="SequenceAbstractPortType">
1839     <wSDL:operation name="CreateSequence">
1840       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1841       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1842       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1843 wsam:Action="http://docs.oasis-open.org/ws-
1844 rx/wsrn/200702/CreateSequenceResponse"/>
1845 </wsdl:operation>
1846 <wsdl:operation name="CloseSequence">
1847 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1848 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1849 <wsdl:output message="tns:CloseSequenceResponse"
1850 wsam:Action="http://docs.oasis-open.org/ws-
1851 rx/wsrn/200702/CloseSequenceResponse"/>
1852 </wsdl:operation>
1853 <wsdl:operation name="TerminateSequence">
1854 <wsdl:input message="tns:TerminateSequence"
1855 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1856 <wsdl:output message="tns:TerminateSequenceResponse"
1857 wsam:Action="http://docs.oasis-open.org/ws-
1858 rx/wsrn/200702/TerminateSequenceResponse"/>
1859 </wsdl:operation>
1860 </wsdl:portType>
1861
1862 </wsdl:definitions>
```

---

## 1863 Appendix C. Message Examples

### 1864 Appendix C.1 Create Sequence

#### 1865 Create Sequence

```
1866 <?xml version="1.0" encoding="UTF-8"?>
1867 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1868 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1869 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1870 <S:Header>
1871 <wsa:MessageID>
1872 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1873 </wsa:MessageID>
1874 <wsa:To>http://example.com/serviceB/123</wsa:To>
1875 <wsa:Action>http://docs.oasis-open.org/ws-
1876 rx/wsm/200702/CreateSequence</wsa:Action>
1877 <wsa:ReplyTo>
1878 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1879 </wsa:ReplyTo>
1880 </S:Header>
1881 <S:Body>
1882 <wsm:CreateSequence>
1883 <wsm:AcksTo>
1884 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1885 </wsm:AcksTo>
1886 </wsm:CreateSequence>
1887 </S:Body>
1888 </S:Envelope>
```

#### 1889 Create Sequence Response

```
1890 <?xml version="1.0" encoding="UTF-8"?>
1891 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1892 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1893 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1894 <S:Header>
1895 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1896 <wsa:RelatesTo>
1897 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1898 </wsa:RelatesTo>
1899 <wsa:Action>
1900 http://docs.oasis-open.org/ws-rx/wsm/200702/CreateSequenceResponse
1901 </wsa:Action>
1902 </S:Header>
1903 <S:Body>
1904 <wsm:CreateSequenceResponse>
1905 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
1906 </wsm:CreateSequenceResponse>
1907 </S:Body>
1908 </S:Envelope>
```

### 1909 Appendix C.2 Initial Transmission

1910 The following example WS-ReliableMessaging headers illustrate the message exchange in the above  
1911 figure. The three messages have the following headers; the third message is identified as the last  
1912 message in the Sequence:

### 1913 Message 1

```
1914 <?xml version="1.0" encoding="UTF-8"?>
1915 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1916 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1917 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1918   <S:Header>
1919     <wsa:MessageID>
1920       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1921     </wsa:MessageID>
1922     <wsa:To>http://example.com/serviceB/123</wsa:To>
1923     <wsa:From>
1924       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1925     </wsa:From>
1926     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1927     <wsmr:Sequence>
1928       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1929       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1930     </wsmr:Sequence>
1931   </S:Header>
1932   <S:Body>
1933     <!-- Some Application Data -->
1934   </S:Body>
1935 </S:Envelope>
```

### 1936 Message 2

```
1937 <?xml version="1.0" encoding="UTF-8"?>
1938 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1939 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1940 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1941   <S:Header>
1942     <wsa:MessageID>
1943       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1944     </wsa:MessageID>
1945     <wsa:To>http://example.com/serviceB/123</wsa:To>
1946     <wsa:From>
1947       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1948     </wsa:From>
1949     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1950     <wsmr:Sequence>
1951       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1952       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1953     </wsmr:Sequence>
1954   </S:Header>
1955   <S:Body>
1956     <!-- Some Application Data -->
1957   </S:Body>
1958 </S:Envelope>
```

### 1959 Message 3

```
1960 <?xml version="1.0" encoding="UTF-8"?>
1961 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1962 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1963 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1964   <S:Header>
1965     <wsa:MessageID>
1966       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1967     </wsa:MessageID>
1968     <wsa:To>http://example.com/serviceB/123</wsa:To>
1969     <wsa:From>
1970       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1971     </wsa:From>
1972     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
```

```

1973 <wsrm:Sequence>
1974 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1975 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1976 </wsrm:Sequence>
1977 <wsrm:AckRequested>
1978 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1979 </wsrm:AckRequested>
1980 </S:Header>
1981 <S:Body>
1982 <!-- Some Application Data -->
1983 </S:Body>
1984 </S:Envelope>

```

## 1985 **Appendix C.3 First Acknowledgement**

1986 Message number 2 has not been accepted by the RM Destination due to some transmission error so it  
1987 responds with an Acknowledgement for messages 1 and 3:

```

1988 <?xml version="1.0" encoding="UTF-8"?>
1989 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1990 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1991 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1992 <S:Header>
1993 <wsa:MessageID>
1994 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1995 </wsa:MessageID>
1996 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1997 <wsa:From>
1998 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1999 </wsa:From>
2000 <wsa:Action>
2001 http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
2002 </wsa:Action>
2003 <wsrm:SequenceAcknowledgement>
2004 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2005 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2006 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2007 </wsrm:SequenceAcknowledgement>
2008 </S:Header>
2009 <S:Body/>
2010 </S:Envelope>

```

## 2011 **Appendix C.4 Retransmission**

2012 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and  
2013 requests an Acknowledgement:

```

2014 <?xml version="1.0" encoding="UTF-8"?>
2015 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2016 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2017 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2018 <S:Header>
2019 <wsa:MessageID>
2020 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2021 </wsa:MessageID>
2022 <wsa:To>http://example.com/serviceB/123</wsa:To>
2023 <wsa:From>
2024 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2025 </wsa:From>
2026 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2027 <wsrm:Sequence>

```



```

2028     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2029     <wsrm:MessageNumber>2</wsrm:MessageNumber>
2030   </wsrm:Sequence>
2031   <wsrm:AckRequested>
2032     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2033   </wsrm:AckRequested>
2034 </S:Header>
2035 <S:Body>
2036   <!-- Some Application Data -->
2037 </S:Body>
2038 </S:Envelope>

```

## 2039 Appendix C.5 Termination

2040 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then  
 2041 be terminated:

```

2042 <?xml version="1.0" encoding="UTF-8"?>
2043 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2044 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2045 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2046   <S:Header>
2047     <wsa:MessageID>
2048       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2049     </wsa:MessageID>
2050     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2051     <wsa:From>
2052       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2053     </wsa:From>
2054     <wsa:Action>
2055       http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2056     </wsa:Action>
2057     <wsrm:SequenceAcknowledgement>
2058       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2059       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2060     </wsrm:SequenceAcknowledgement>
2061   </S:Header>
2062   <S:Body/>
2063 </S:Envelope>

```

## 2064 Terminate Sequence

```

2065 <?xml version="1.0" encoding="UTF-8"?>
2066 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2067 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2068 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2069   <S:Header>
2070     <wsa:MessageID>
2071       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2072     </wsa:MessageID>
2073     <wsa:To>http://example.com/serviceB/123</wsa:To>
2074     <wsa:Action>
2075       http://docs.oasis-open.org/ws-rx/wsr/200702/TerminateSequence
2076     </wsa:Action>
2077     <wsa:From>
2078       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2079     </wsa:From>
2080   </S:Header>
2081   <S:Body>
2082     <wsrm:TerminateSequence>
2083       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2084       <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2085     </wsrm:TerminateSequence>

```

2086 </S:Body>  
2087 </S:Envelope>

## 2088 Terminate Sequence Response

```
2089 <?xml version="1.0" encoding="UTF-8"?>
2090 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2091 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2092 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2093   <S:Header>
2094     <wsa:MessageID>
2095       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2096     </wsa:MessageID>
2097     <wsa:To>http://example.com/serviceA/789</wsa:To>
2098     <wsa:Action>
2099       http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
2100     </wsa:Action>
2101     <wsa:RelatesTo>
2102       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2103     </wsa:RelatesTo>
2104     <wsa:From>
2105       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2106     </wsa:From>
2107   </S:Header>
2108   <S:Body>
2109     <wsm:TerminateSequenceResponse>
2110       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2111     </wsm:TerminateSequenceResponse>
2112   </S:Body>
2113 </S:Envelope>
```

---

## 2114 Appendix D. State Tables

2115 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2116 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

2117 Legend:

2118 The first column of these tables contains the motivating event and has the following format:

<b>Event</b> 2119
<i>Event name</i> [source]
{ref}

2120 Where:

- 2121 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as  
2122 described by the specification.
- 2123 • [source]: indicates the source of the event; one of:
  - 2124 ○ [msg] a Received message
  - 2125 ○ [int]: an internal event such as the firing of a timer
  - 2126 ○ [app]: the application
  - 2127 ○ [unspec]: the source is unspecified

2128 Each event / state combination cell in the tables in this appendix has the following format:

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

2129 Where:

- 2130 • action to take: indicates that the state machine performs the following action. Actions surrounded  
2131 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word  
2132 "Transmit"
- 2133 • [next state]: indicates the state to which the state machine will advance upon the performance of  
2134 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2135 • {ref} is a reference to the document section describing the behavior in this cell

2136 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these  
2137 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not  
2138 described in this specification and does not indicate normal protocol operation. Implementations MAY  
2139 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations  
2140 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state  
2141 combinations.

2142 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] {2.3}	Xmit message [Same] {2.3}	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 [Same]	No action [Same]	No action [Same]	No action [Same]
<b>CloseSequence</b> [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}
<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}

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
<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] {4.7}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]
<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>TerminateSequence</b> [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}
<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] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
<b>Invalid Acknowledgement</b> [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

2143 Table 2 RM Destination Sequence State Transition Table

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

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

Events	Sequence States			
	None	Created	Closed	Terminating
		{3.4}	{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}	

---

## 2144 Appendix E. Acknowledgments

2145 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following  
2146 authors:

2147	Ruslan Bilursets, BEA	2159	Amelia Lewis, TIBCO Software
2148	Don Box, Microsoft	2160	Rodney Limprecht, Microsoft
2149	Luis Felipe Cabrera, Microsoft	2161	Steve Lucco, Microsoft
2150	Doug Davis, IBM	2162	Don Mullen, TIBCO Software
2151	Donald Ferguson, IBM	2163	Anthony Nadalin, IBM
2152	Christopher Ferris, IBM	2164	Mark Nottingham, BEA
2153	Tom Freund, IBM	2165	David Orchard, BEA
2154	Mary Ann Hondo, IBM	2166	Jamie Roots, IBM
2155	John Ibbotson, IBM	2167	Shivajee Samdarshi, TIBCO Software
2156	Lei Jin, BEA	2168	John Shewchuk, Microsoft
2157	Chris Kaler, Microsoft	2169	Tony Storey, IBM
2158	David Langworthy-Editor, Microsoft		

2170 The following individuals have provided invaluable input into the initial contribution:

2171	Keith Ballinger, Microsoft	2185	David Ingham, Microsoft
2172	Stefan Batres, Microsoft	2186	Gopal Kakivaya, Microsoft
2173	Rebecca Bergersen, Iona	2187	Johannes Klein, Microsoft
2174	Allen Brown, Microsoft	2188	Frank Leymann, IBM
2175	Michael Conner, IBM	2189	Martin Nally, IBM
2176	George Copeland, Microsoft	2190	Peter Niblett, IBM
2177	Francisco Curbera, IBM	2191	Jeffrey Schlimmer, Microsoft
2178	Paul Fremantle, IBM	2192	James Snell, IBM
2179	Steve Graham, IBM	2193	Keith Stobie, Microsoft
2180	Pat Helland, Microsoft	2194	Satish Thatte, Microsoft
2181	Rick Hill, Microsoft	2195	Stephen Todd, IBM
2182	Scott Hinkelman, IBM	2196	Sanjiva Weerawarana, IBM
2183	Tim Holloway, IBM	2197	Roger Wolter, Microsoft
2184	Efim Hudis, Microsoft		

2198 The following individuals were members of the committee during the development of this specification:

2199	Abbie Barbir, Nortel	2218	Robert Freund, Hitachi
2200	Charlton Barreto, Adobe	2219	Peter Furniss, Erebore
2201	Stefan Batres, Microsoft	2220	Marc Goodner, Microsoft
2202	Hamid Ben Malek, Fujitsu	2221	Alastair Green, Choreology
2203	Andreas Bjarlestam, Ericsson	2222	Mike Grogan, Sun
2204	Toufic Boubez, Layer 7	2223	Ondrej Hrebicek, Microsoft
2205	Doug Bunting, Sun	2224	Kazunori Iwasa, Fujitsu
2206	Lloyd Burch, Novell	2225	Chamikara Jayalath, WSO2
2207	Steve Carter, Novell	2226	Lei Jin, BEA
2208	Martin Chapman, Oracle	2227	Ian Jones, BTplc
2209	Dave Chappell, Sonic	2228	Anish Karmarkar, Oracle
2210	Paul Cotton, Microsoft	2229	Paul Knight, Nortel
2211	Glen Daniels, Sonic	2230	Dan Leshchiner, Tibco
2212	Doug Davis, IBM	2231	Mark Little, JBoss
2213	Blake Dournaee, Intel	2232	Lily Liu, webMethods
2214	Jacques Durand, Fujitsu	2233	Matt Lovett, IBM
2215	Colleen Evans, Microsoft	2234	Ashok Malhotra, Oracle
2216	Christopher Ferris, IBM	2235	Jonathan Marsh, Microsoft
2217	Paul Fremantle, WSO2	2236	Daniel Millwood, IBM



2237	Jeff Mischkinsky, Oracle	2248	Stefan Rossmannith, SAP
2238	Nilo Mitra, Ericsson	2249	Tom Rutt, Fujitsu
2239	Peter Niblett, IBM	2250	Rich Salz, IBM
2240	Duane Nickull, Adobe	2251	Shivajee Samdarshi, Tibco
2241	Eisaku Nishiyama, Hitachi	2252	Vladimir Videlov, SAP
2242	Dave Orchard, BEA	2253	Claus von Riegen, SAP
2243	Chouthri Palanisamy, NEC	2254	Pete Wenzel, Sun
2244	Sanjay Patil, SAP	2255	Steve Winkler, SAP
2245	Gilbert Pilz, BEA	2256	Ümit Yalçinalp, SAP
2246	Martin Raeppele, SAP	2257	Nobuyuki Yamamoto, Hitachi
2247	Eric Rajkovic, Oracle		
2258			