

## 1 **Web Services Reliable Messaging (WS- 2 ReliableMessaging) Version 1.2**

### 3 **Committee Draft 02**

### 4 **20 November 2008**

#### 5 **Specification URIs:**

##### 6 **This Version:**

- 7 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-02.pdf>
- 8 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-02.html>
- 9 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-02.doc> (Authoritative)

##### 10 **Previous Version:**

- 11 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.pdf>
- 12 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.html>
- 13 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.doc>

##### 14 **Latest Version:**

- 15 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.pdf>
- 16 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.html>
- 17 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/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 v1.1](#)

##### 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® 1993–2008. All Rights Reserved.

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, 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 Committee Draft, "Web Services Reliable Messaging  
202 Policy Assertion( WS-RM Policy)," November 2008  
203 <http://docs.oasis-open.org/ws-rx/wsrmp/v1.2/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 Recommendation, "Web Services Policy 1.5 - Framework," September  
 251 2007.  
 252 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>
- 253 **[WS-PolicyAttachment]** W3C Recommendation, "Web Services Policy 1.5 - Attachment,"  
 254 September 2007.  
 255 <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>
- 256 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS  
 257 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",  
 258 OASIS Standard 200401, March 2004.  
 259 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>  
 260
- 261 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS  
 262 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)",  
 263 OASIS Standard 200602, February 2006.  
 264 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 265 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",  
 266 RFC 1323, May 1992.  
 267 <http://www.rfc-editor.org/rfc/rfc1323.txt>
- 268 **[SecurityPolicy]** OASIS WS-SX Technical Committee Editor Draft, "WS-SecurityPolicy 1.3"  
 269 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802>
- 270 **[SecureConversation]** OASIS WS-SX Technical Committee Editor Draft, "WS-  
 271 SecureConversation 1.4"  
 272 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512>
- 273 **[Trust]** OASIS WS-SX Technical Committee Editor Draft, "WS-Trust 1.4"  
 274 <http://docs.oasis-open.org/ws-sx/ws-trust/200802>

## 275 1.4 Namespace

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

277 <http://docs.oasis-open.org/ws-rx/wsrml/200702>

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

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

282 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>
wsrml	<a href="http://docs.oasis-open.org/ws-rx/wsrml/200702">http://docs.oasis-open.org/ws-rx/wsrml/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>

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

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

## 286 **1.5 Conformance**

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

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

---

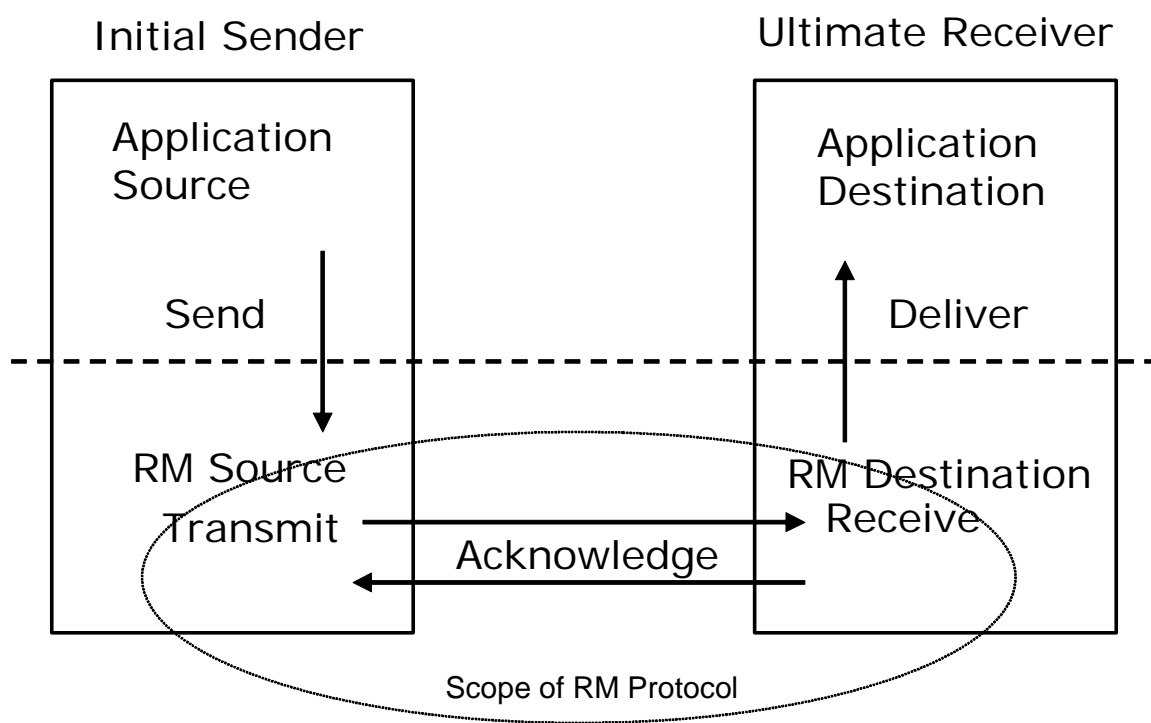
## 293 2 Reliable Messaging Model

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

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

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

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



319 Figure 1: Reliable Messaging Model

## 320 2.1 Glossary

321 The following definitions are used throughout this specification:

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

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

326 **Acknowledgement Message:** A message containing a *SequenceAcknowledgement* header block.  
327 Acknowledgement Messages may or may not contain a SOAP body.

328 **Acknowledgement Request:** A message containing an *AckRequested* header. Acknowledgement  
329 Requests may or may not contain a SOAP body.

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

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

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

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

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

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

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

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

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

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

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

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

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

## 351 2.2 Protocol Preconditions

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

- 354 • For any single message exchange the RM Source **MUST** have an endpoint reference that  
355 uniquely identifies the RM Destination Endpoint.
- 356 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 357 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's  
358 policies.
- 359 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**  
360 have a security context.

## 361 2.3 Protocol Invariants

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

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

## 375 2.4 Delivery Assurances

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

379 `AtLeastOnce`

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

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

#### 387 AtMostOnce

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

#### 392 ExactlyOnce

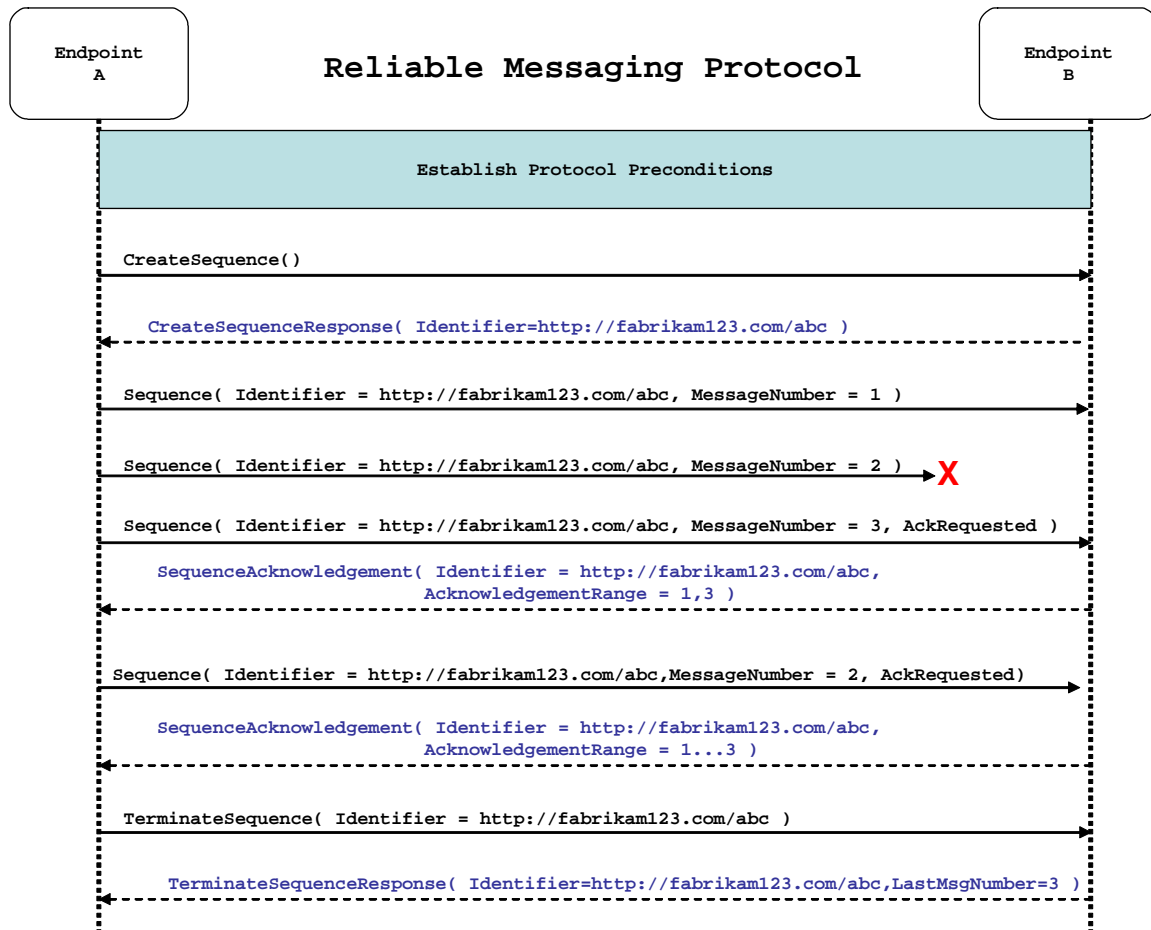
393 Each message is to be delivered exactly once; if a message cannot be delivered then an error  
394 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is  
395 that it SHOULD retry transmission of every message sent by the Application Source until it  
396 receives an acknowledgement from the RM Destination. The requirement on the RM Destination  
397 is that it SHOULD retry the transfer to the Application Destination of any message that it accepts  
398 from the RM Source until that message has been successfully delivered, and that it MUST NOT  
399 deliver a duplicate of a message that has already been delivered.

#### 400 InOrder

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

## 412 2.5 Example Message Exchange

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



414 Figure 2: The WS-ReliableMessaging Protocol

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

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

---

## 455 3 RM Protocol Elements

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

### 458 3.1 Considerations on the Use of Extensibility Points

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

### 463 3.2 Considerations on the Use of "Piggy-Backing"

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

### 473 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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



## 491 3.4 Sequence Creation

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

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

499 The following exemplar defines the `CreateSequence` syntax:

```
500 <wsrm:CreateSequence ...>
501   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
502   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
503   <wsrm:Offer ...>
504     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
505     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
506     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
507     <wsrm:IncompleteSequenceBehavior>
508       wsrml:IncompleteSequenceBehaviorType
509     </wsrm:IncompleteSequenceBehavior> ?
510     ...
511   </wsrm:Offer> ?
512   ...
513 </wsrm:CreateSequence>
```

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

515 `/wsrm:CreateSequence`

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

520 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

530 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

538 /wsm:CreateSequence/wsm:Offer

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

541 /wsm:CreateSequence/wsm:Offer/wsm:Identifier

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

544 /wsm:CreateSequence/wsm:Offer/wsm:Identifier/@{any}

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

547 /wsm:CreateSequence/wsm:Offer/wsm:Endpoint

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

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

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

564 /wsm:CreateSequence/wsm:Offer/wsm:Expires

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

568 /wsm:CreateSequence/wsm:Offer/wsm:Expires/@{any}

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

571 /wsm:CreateSequence/wsm:Offer/wsm:IncompleteSequenceBehavior

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

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

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

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

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

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

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

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

590 /wsmr:CreateSequence/{any}

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

593 /wsmr:CreateSequence/@{any}

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

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

600 The following exemplar defines the `CreateSequenceResponse` syntax:

```
601 <wsmr:CreateSequenceResponse ...>  
602   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
603   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
604   <wsmr:IncompleteSequenceBehavior>  
605     wsmr:IncompleteSequenceBehaviorType  
606   </wsmr:IncompleteSequenceBehavior> ?  
607   <wsmr:Accept ...>  
608     <wsmr:AcksTo wsa:EndpointReferenceType </wsmr:AcksTo>  
609     ...  
610   </wsmr:Accept> ?  
611   ...  
612 </wsmr:CreateSequenceResponse>
```

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

614 /wsmr:CreateSequenceResponse

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

618 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

626 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

639 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

652 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

658 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

673 /wsrm:CreateSequenceResponse/{ any }

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

676 /wsrm:CreateSequenceResponse/@{ any }

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

### 679 3.5 Closing A Sequence

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

685 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of  
686 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept  
687 any new messages for the specified Sequence, other than those already accepted at the time the  
688 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or  
689 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST  
690 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`  
691 element) header block on any messages associated with the Sequence destined to the RM Source,  
692 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM  
693 Source.

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

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

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

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

713 The following exemplar defines the `CloseSequence` syntax:

```
714 <wsrm:CloseSequence ...>  
715   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
716   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?
```

```
717     ...
718 </wsmr:CloseSequence>
```

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

720 `/wsmr:CloseSequence`

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

724 `/wsmr:CloseSequence/wsmr:Identifier`

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

728 `/wsmr:CloseSequence/wsmr:LastMsgNumber`

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

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

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

735 `/wsmr:CloseSequence/{any}`

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

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

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

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

743 The following exemplar defines the `CloseSequenceResponse` syntax:

```
744 <wsmr:CloseSequenceResponse ...>
745   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
746   ...
747 </wsmr:CloseSequenceResponse>
```

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

749 `/wsmr:CloseSequenceResponse`

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

752 `/wsmr:CloseSequenceResponse/wsmr:Identifier`

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

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

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

759 /wsrm:CloseSequenceResponse/{any}  
760 This is an extensibility mechanism to allow different (extensible) types of information, based on a  
761 schema, to be passed.

762 /wsrm:CloseSequenceResponse/@{any}  
763 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
764 to the element.

### 765 3.6 Sequence Termination

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

773 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM  
774 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.  
775 The RM Destination can use this information, for example, to implement the behavior indicated by  
776 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the  
777 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the  
778 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same  
779 Sequence.

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

784 The following exemplar defines the `TerminateSequence` syntax:

```
785 <wsrm:TerminateSequence ...>  
786   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
787   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
788   ...  
789 </wsrm:TerminateSequence>
```

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

791 /wsrm:TerminateSequence

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

797 This element MAY also be sent by the RM Destination to indicate that it has unilaterally  
798 terminated the Sequence. Upon sending this message the RM Destination MUST NOT accept  
799 any additional messages (with the exception of the corresponding  
800 `TerminateSequenceResponse`) for this Sequence. Upon receipt of a `TerminateSequence`  
801 the RM Source MUST NOT send any additional messages (with the exception of the  
802 corresponding `TerminateSequenceResponse`) for this Sequence.

803 /wsrm:TerminateSequence/wsrm:Identifier

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

807 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

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

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

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

814 `/wsrm:TerminateSequence/{any}`

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

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

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

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

822 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
823 <wsrm:TerminateSequenceResponse ...>  
824   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
825   ...  
826 </wsrm:TerminateSequenceResponse>
```

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

828 `/wsrm:TerminateSequenceResponse`

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

832 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

839 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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



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

## 848 3.7 Sequences

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

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

856 A following exemplar defines its syntax:

```
857 <wsrm:Sequence ...>  
858   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
859   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
860   ...  
861 </wsrm:Sequence>
```

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

863 `/wsrm:Sequence`

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

870 `/wsrm:Sequence/wsrm:Identifier`

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

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

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

877 `/wsrm:Sequence/wsrm:MessageNumber`

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

882 `/wsrm:Sequence/{any}`

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

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

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

888 The following example illustrates a Sequence header block.

```
889 <wsrm:Sequence>
890   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
891   <wsrm:MessageNumber>10</wsrm:MessageNumber>
892 </wsrm:Sequence>
```

### 893 3.8 Request Acknowledgement

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

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

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

908 The following exemplar defines its syntax:

```
909 <wsrm:AckRequested ...>
910   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
911   ...
912 </wsrm:AckRequested>
```

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

914 /wsrm:AckRequested

915       This element requests an Acknowledgement for the identified Sequence.

916 /wsrm:AckRequested/wsrm:Identifier

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

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

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

924 /wsrm:AckRequested/{any}

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

927 /wsrm:AckRequested/@{any}

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

## 930 3.9 Sequence Acknowledgement

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

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

941 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the  
942 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing  
943 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any  
944 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted  
945 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received  
946 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`  
947 header block for that same `Sequence Identifier`. When the RM Destination receives an  
948 `AckRequested` header, and the `AcksTo` EPR for that Sequence is the WS-Addressing anonymous IRI,  
949 the RM Destination SHOULD respond on the protocol binding-specific back-channel provided by the  
950 Received message containing the `AckRequested` header block.

951 The following exemplar defines its syntax:

```
952 <wsrm:SequenceAcknowledgement ...>  
953   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
954   [ [ [ <wsrm:AcknowledgementRange ...  
955         Upper="wsrm:MessageNumberType"  
956         Lower="wsrm:MessageNumberType" /> +  
957         | <wsrm:None/> ]  
958         <wsrm:Final/> ? ]  
959         | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]  
960   ...  
961 </wsrm:SequenceAcknowledgement>
```

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

964 `/wsrm:SequenceAcknowledgement`

965       This element contains the Sequence Acknowledgement information.

966 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

975 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

981 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper`  
982 The RM Destination MUST set the value of this attribute equal to the message number of the  
983 highest contiguous message in a Sequence range accepted by the RM Destination.

984 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower`  
985 The RM Destination MUST set the value of this attribute equal to the message number of the  
986 lowest contiguous message in a Sequence range accepted by the RM Destination.

987 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}`  
988 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
989 to the element.

990 `/wsrm:SequenceAcknowledgement/wsrm:None`  
991 The RM Destination MUST include this element within a `SequenceAcknowledgement` header  
992 block if the RM Destination has not accepted any messages for the specified Sequence. The RM  
993 Destination MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack`  
994 element is also present as a child of the `SequenceAcknowledgement`.

995 `/wsrm:SequenceAcknowledgement/wsrm:Final`  
996 The RM Destination MAY include this element within a `SequenceAcknowledgement` header  
997 block. This element indicates that the RM Destination is not receiving new messages for the  
998 specified Sequence. The RM Source can be assured that the ranges of messages acknowledged  
999 by this `SequenceAcknowledgement` header block will not change in the future. The RM  
1000 Destination MUST include this element when the Sequence is closed. The RM Destination MUST  
1001 NOT include this element when sending a `Nack`; it can only be used when sending  
1002 `AcknowledgementRange` elements or a `None`.

1003 `/wsrm:SequenceAcknowledgement/wsrm:Nack`  
1004 The RM Destination MAY include this element within a `SequenceAcknowledgement` header  
1005 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`  
1006 representing the `MessageNumber` of an unreceived message in a Sequence. The RM Destination  
1007 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is  
1008 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM  
1009 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT  
1010 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously  
1011 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a  
1012 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been  
1013 acknowledged within an `AcknowledgementRange`.

1014 `/wsrm:SequenceAcknowledgement/{any}`  
1015 This is an extensibility mechanism to allow different (extensible) types of information, based on a  
1016 schema, to be passed.

1017 `/wsrm:SequenceAcknowledgement/@{any}`  
1018 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
1019 to the element.

1020 The following examples illustrate SequenceAcknowledgement elements:

- 1021 • Message numbers 1...10 inclusive in a Sequence have been accepted by the RM Destination.

```
1022 <wsrm:SequenceAcknowledgement>  
1023   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1024   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
1025 </wsrm:SequenceAcknowledgement>
```

- 1026 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM  
1027 Destination, messages 3 and 7 have not been accepted.

```
1028 <wsrm:SequenceAcknowledgement>  
1029   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1030   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1031   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1032   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1033 </wsrm:SequenceAcknowledgement>
```

- 1034 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1035 <wsrm:SequenceAcknowledgement>  
1036   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1037   <wsrm:Nack>3</wsrm:Nack>  
1038 </wsrm:SequenceAcknowledgement>
```

## 1039 4 Faults

1040 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. `Create`  
1041 `Sequence Refused` is a possible fault reply for this operation. `Unknown Sequence` is a fault generated by  
1042 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences  
1043 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM  
1044 on a Received message that did not use the protocol. All other faults in this section relate to known  
1045 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those faults.  
1046 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

1047 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault  
1048 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1049 `http://docs.oasis-open.org/ws-rx/wsrn/200702/fault`

1050 The faults defined in this section are generated if the condition stated in the preamble is met. Fault  
1051 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1052 The definitions of faults use the following properties:

1053 [Code] The fault code.

1054 [Subcode] The fault subcode.

1055 [Reason] The English language reason element.

1056 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail  
1057 element is defined for a fault, implementations MUST include the elements in the order that they are  
1058 specified.

1059 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or  
1060 "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

1061 The properties above bind to a SOAP 1.2 fault as follows:

```
1062 <S:Envelope>
1063   <S:Header>
1064     <wsa:Action>
1065       http://docs.oasis-open.org/ws-rx/wsrn/200702/fault
1066     </wsa:Action>
1067     <!-- Headers elided for brevity. -->
1068   </S:Header>
1069   <S:Body>
1070     <S:Fault>
1071       <S:Code>
1072         <S:Value> [Code] </S:Value>
1073         <S:Subcode>
1074           <S:Value> [Subcode] </S:Value>
1075         </S:Subcode>
1076       </S:Code>
1077       <S:Reason>
1078         <S:Text xml:lang="en"> [Reason] </S:Text>
1079       </S:Reason>
1080       <S:Detail>
1081         [Detail]
```

```
1082     ...
1083     </S:Detail>
1084     </S:Fault>
1085     </S:Body>
1086     </S:Envelope>
```

1087 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM  
1088 header block:

```
1089 <S11:Envelope>
1090   <S11:Header>
1091     <wsrm:SequenceFault>
1092       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1093       <wsrm:Detail> [Detail] </wsrm:Detail>
1094       ...
1095     </wsrm:SequenceFault>
1096     <!-- Headers elided for brevity. -->
1097   </S11:Header>
1098   <S11:Body>
1099     <S11:Fault>
1100       <faultcode> [Code] </faultcode>
1101       <faultstring> [Reason] </faultstring>
1102     </S11:Fault>
1103   </S11:Body>
1104 </S11:Envelope>
```

1105 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a  
1106 CreateSequence request message:

```
1107 <S11:Envelope>
1108   <S11:Body>
1109     <S11:Fault>
1110       <faultcode> [Subcode] </faultcode>
1111       <faultstring> [Reason] </faultstring>
1112     </S11:Fault>
1113   </S11:Body>
1114 </S11:Envelope>
```

## 1115 4.1 SequenceFault Element

1116 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the  
1117 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging  
1118 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.  
1119 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the  
1120 SOAP 1.2 binding.

1121 The following exemplar defines its syntax:

```
1122 <wsrm:SequenceFault ...>
1123   <wsrm:FaultCode> wsrm:FaultCode </wsrm:FaultCode>
1124   <wsrm:Detail> ... </wsrm:Detail> ?
1125   ...
1126 </wsrm:SequenceFault>
```

1127 The following describes the content model of the `SequenceFault` element.

1128 /wsrm: SequenceFault

1129       This is the element containing Sequence fault information for WS-ReliableMessaging

1130 /wsrm: SequenceFault/wsrm: FaultCode

- 1131 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this  
 1132 element to a qualified name from the set of faults [Subcodes] defined below.
- 1133 `/wsrm:SequenceFault/wsrn:Detail`
- 1134 This element, if present, carries application specific error information related to the fault being  
 1135 described.
- 1136 `/wsrm:SequenceFault/wsrn:Detail/{any}`
- 1137 The application specific error information related to the fault being described.
- 1138 `/wsrm:SequenceFault/wsrn:Detail/@{any}`
- 1139 The application specific error information related to the fault being described.
- 1140 `/wsrm:SequenceFault/{any}`
- 1141 This is an extensibility mechanism to allow different (extensible) types of information, based on a  
 1142 schema, to be passed.
- 1143 `/wsrm:SequenceFault/@{any}`
- 1144 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added  
 1145 to the element.

## 1146 4.2 Sequence Terminated

- 1147 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding  
 1148 Endpoint of this decision.
- 1149 Properties:
- 1150 [Code] Sender or Receiver
- 1151 [Subcode] `wsrn:SequenceTerminated`
- 1152 [Reason] The Sequence has been terminated due to an unrecoverable error.
- 1153 [Detail]
- 1154 `<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.

## 1155 4.3 Unknown Sequence

- 1156 Properties:
- 1157 [Code] Sender
- 1158 [Subcode] `wsrn:UnknownSequence`



1159 [Reason] The value of `wsrc:Identifier` is not a known Sequence identifier.

1160 [Detail]

1161 `<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.

## 1162 4.4 Invalid Acknowledgement

1163 An example of when this fault is generated is when a message is Received by the RM Source containing  
1164 a `SequenceAcknowledgement` covering messages that have not been sent.

1165 [Code] Sender

1166 [Subcode] `wsrc:InvalidAcknowledgement`

1167 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1168 [Detail]

1169 `<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.

## 1170 4.5 Message Number Rollover

1171 If the condition listed below is reached, the RM Destination MUST generate this fault.

1172 Properties:

1173 [Code] Sender

1174 [Subcode] wsrM:MessageNumberRollover

1175 [Reason] The maximum value for wsrM:MessageNumber has been exceeded.

1176 [Detail]

1177 <wsrM:Identifier ...> xs:anyURI </wsrM:Identifier>

1178 <wsrM:MaxMessageNumber> wsrM:MessageNumberType </wsrM:MaxMessageNumber>

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	Message number in /wsrM:Sequence/wsrM:MessageNumber of a Received message exceeds the internal limitations of an RM Destination or reaches the maximum value of 9,223,372,036,854,775,807.	RM Destination SHOULD continue to accept undelivered messages until the Sequence is closed or terminated.	RM Source SHOULD continue to retransmit undelivered messages until the Sequence is closed or terminated.

## 1179 4.6 Create Sequence Refused

1180 Properties:

1181 [Code] Sender or Receiver

1182 [Subcode] wsrM:CreateSequenceRefused

1183 [Reason] The Create Sequence request has been refused by the RM Destination.

1184 [Detail]

1185 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.

## 1186 4.7 Sequence Closed

1187 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1188 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that  
1189 is closed.

1190 Properties:

1191 [Code] Sender

1192 [Subcode] wsr:SequenceClosed

1193 [Reason] The Sequence is closed and cannot accept new messages.

1194 [Detail]

1195 `<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.

## 1196 **4.8 WSRM Required**

1197 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming  
1198 message that did not use this protocol.

1199 Properties:

1200 [Code] Sender

1201 [Subcode] wsr:WSRMRequired

1202 [Reason] The RM Destination requires the use of WSRM.

1203 [Detail]

1204 `xs:any`

---

## 1205 5 Security Threats and Countermeasures

1206 This specification considers two sets of security requirements, those of the applications that use the WS-  
1207 RM protocol and those of the protocol itself.

1208 This specification makes no assumptions about the security requirements of the applications that use WS-  
1209 RM. However, once those requirements have been satisfied within a given operational context, the  
1210 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;  
1211 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1212 There are many other security concerns that one may need to consider when implementing or using this  
1213 protocol. The material below should not be considered as a "check list". Implementers and users of this  
1214 protocol are urged to perform a security analysis to determine their particular threat profile and the  
1215 appropriate responses to those threats.

1216 Implementers are also advised that there is a core tension between security and reliable messaging that  
1217 can be problematic if not addressed by implementations; one aspect of security is to prevent message  
1218 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.  
1219 Consequently, if the security sub-system processes a message but a failure occurs before the reliable  
1220 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system  
1221 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-  
1222 system will likely continue to expect and even solicit the missing message(s). Care should be taken to  
1223 avoid and prevent this condition.

### 1224 5.1 Threats and Countermeasures

1225 The primary security requirement of this protocol is to protect the specified semantics and protocol  
1226 invariants against various threats. The following sections describe several threats to the integrity and  
1227 operation of this protocol and provide some general outlines of countermeasures to those threats.  
1228 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable  
1229 to all operational contexts.

#### 1230 5.1.1 Integrity Threats

1231 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic  
1232 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or  
1233 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block  
1234 to its intended message represents a threat to the WS-RM protocol.

1235 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM  
1236 Source and RM Destination then they have undermined the implementation's ability to guarantee the first  
1237 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be  
1238 Delivered to the Application Destination in the same order that they were sent by the Application Source.

##### 1239 5.1.1.1 Countermeasures

1240 Integrity threats are generally countered via the use of digital signatures some level of the communication  
1241 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include  
1242 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers  
1243 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in  
1244 which they occur, implementations MUST allow for signatures that cover only these headers.

## 1245 **5.1.2 Resource Consumption Threats**

1246 The creation of a Sequence with an RM Destination consumes various resources on the systems used to  
1247 implement that RM Destination. These resources can include network connections, database tables,  
1248 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM  
1249 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM  
1250 Destination. Another attack is to create a Sequence for a service that is known to require in-order  
1251 message Delivery and use this Sequence to send a stream of very large messages to that service, making  
1252 sure to omit message number “1” from that stream.

### 1253 **5.1.2.1 Countermeasures**

1254 There are a number of countermeasures against the described resource consumption threats. The  
1255 technique advocated by this specification is for the RM Destination to restrict the ability to create a  
1256 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in  
1257 some cases, allows the identity of any attackers to be determined.

1258 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and  
1259 authenticate the RM Source that issued the `CreateSequence` message.

## 1260 **5.1.3 Sequence Spoofing Threats**

1261 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a  
1262 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a  
1263 fake `TerminateSequence` message that references the target Sequence and sends this message to the  
1264 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the  
1265 current `MessageNumber` for their target Sequence.

1266 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP  
1267 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence  
1268 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to  
1269 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a  
1270 message that it sends to the `AcksTo` EPR of an RM Source.

### 1271 **5.1.3.1 Sequence Hijacking**

1272 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject  
1273 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those  
1274 messages.

1275 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a  
1276 Sequence may be bound to some form of a security session in order to counter the threats described in  
1277 this section, applications MUST NOT rely on WS-RM-related information to make determinations about  
1278 the identity of the entity that created a message; applications SHOULD rely only upon information that is  
1279 established by the security infrastructure to make such determinations. Failure to observe this rule  
1280 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of  
1281 the ability to authenticate its peers even though the necessary security processing has taken place.

### 1282 **5.1.3.2 Countermeasures**

1283 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by  
1284 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM  
1285 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination  
1286 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

1287 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it  
1288 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the  
1289 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it  
1290 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the  
1291 referenced Sequence.

1292 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and  
1293 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify  
1294 its Sequence peer it MUST be able to identify and authenticate the entity that sent the  
1295 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a  
1296 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that  
1297 message and, if necessary, correlate this identity with the Sequence peer identity established at  
1298 Sequence creation time.

## 1299 **5.2 Security Solutions and Technologies**

1300 The security threats described in the previous sections are neither new nor unique. The solutions that  
1301 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This  
1302 section maps the facilities provided by common web services security solutions against countermeasures  
1303 described in the previous sections.

1304 Before continuing this discussion, however, some examination of the underlying requirements of the  
1305 previously described countermeasures is necessary. Specifically it should be noted that the technique  
1306 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates  
1307 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization  
1308 check against this authenticated identity and determines if the RM Source is permitted to create  
1309 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime  
1310 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any  
1311 discussion of such facilities is considered to be beyond the scope of this specification.

### 1312 **5.2.1 Transport Layer Security**

1313 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the  
1314 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints  
1315 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1316 The description provided here is general in nature and is not intended to serve as a complete definition on  
1317 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the  
1318 choice of features as well as the manner in which they will be used. The mechanisms described in the  
1319 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the  
1320 requirements and constraints of the use of SSL/TLS.

#### 1321 **5.2.1.1 Model**

1322 The basic model for using SSL/TLS is as follows:

- 1323 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1324 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM  
1325 Destination.
- 1326 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an  
1327 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a  
1328 synchronous `CreateSequenceResponse` using the session established in (1).

- 1329 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit  
1330 any and all messages or faults that refer to that Sequence.
- 1331 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established  
1332 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous  
1333 exchanges, the RM Destination uses the SSL/TLS session established in (1).

### 1334 5.2.1.2 Countermeasure Implementation

1335 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the  
1336 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the  
1337 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If  
1338 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and  
1339 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1340 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification  
1341 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods  
1342 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS  
1343 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1344 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP  
1345 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the  
1346 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party  
1347 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself  
1348 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.  
1349 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an  
1350 Acknowledgement) using BasicAuth.
- 1351 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the  
1352 connection authenticates itself to the party accepting the connection using an X.509 certificate  
1353 that is exchanged during the SSL/TLS handshake.

1354 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself  
1355 using one the above mechanisms. The authenticated identity can then be used to determine if the RM  
1356 Source is authorized to create a Sequence with the RM Destination.

1357 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1358 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the  
1359 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than  
1360 on authentication information. For example, an RM Destination can determine that a Sequence Traffic  
1361 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS  
1362 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a  
1363 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a  
1364 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used  
1365 to protect that Sequence.

1366 This specification does not preclude the use of other methods of using SSL/TLS to implement the  
1367 countermeasures (such as associating specific authentication information with a Sequence) although such  
1368 methods are not covered by this document.

1369 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS  
1370 session) are outside the scope of this specification.

### 1371 5.2.2 SOAP Message Security

1372 The mechanisms described in WS-Security may be used in various ways to implement the  
1373 countermeasures described in the previous sections. This specification advocates using the protocol  
1374 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1375 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component  
1376 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1377 The description provided here is general in nature and is not intended to serve as a complete definition on  
1378 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations  
1379 need to agree on the choice of features as well as the manner in which they will be used. The  
1380 mechanisms described in the Web Services Security Policy Language MAY be used by services to  
1381 describe the requirements and constraints of the use of WS-SecureConversation.

### 1382 **5.2.2.1 Model**

1383 The basic model for using WS-SecureConversation is as follows:

- 1384       1    The RM Source and the RM Destination create a WS-SecureConversation security context. This  
1385            may involve the participation of third parties such as a security token service. The tokens  
1386            exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service  
1387            tickets).
- 1388       2    During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security  
1389            context that will be used to protect the Sequence. This is done so that, in cases where the  
1390            `CreateSequence` message is signed by more than one security context, the RM Source can  
1391            indicate which security context should be used to protect the newly created Sequence.
- 1392       3    For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)  
1393            associated with the security context to sign (as defined by WS-Security) at least the body and  
1394            any relevant WS-RM-defined headers of any and all messages or faults that refer to that  
1395            Sequence.

### 1396 **5.2.2.2 Countermeasure Implementation**

1397 Without relying upon any authentication information, the per-message signatures provide the necessary  
1398 integrity qualities to counter the threats described in section 5.1.1.

1399 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of  
1400 authentication claims must be provided by the RM Source to the RM Destination during the establishment  
1401 of the Security Context. These claims can then be used to determine if the RM Source is authorized to  
1402 create a Sequence with the RM Destination.

1403 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1404 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the  
1405 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security  
1406 context rather than on any authentication claims that may have been established during security context  
1407 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures  
1408 (such as associating specific authentication claims to a Sequence) are possible but not covered by this  
1409 document.

1410 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits  
1411 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the  
1412 association between a Sequence and its protecting security context cannot always be established  
1413 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and  
1414 `CreateSequenceResponse` messages may be signed by more than one security context.

1415 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as  
1416 amending or renewing contexts) are outside the scope of this specification.



---

## 1417 6 Securing Sequences

1418 As noted in section 5, the RM Source and RM Destination should be able to protect their shared  
1419 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of  
1420 achieving this objective depending upon the underlying security infrastructure.

### 1421 6.1 Securing Sequences Using WS-Security

1422 One mechanism for protecting a Sequence is to include a security token using a  
1423 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-  
1424 SecureConversation) in the `CreateSequence` element. This establishes an association between the  
1425 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source  
1426 and Destination MUST use the security token as the basis for authorization of all subsequent interactions  
1427 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as  
1428 there may be more than one token on a `CreateSequence` message or inferred from the communication  
1429 context (e.g. transport protection).

1430 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,  
1431 if the token being referenced supports such mechanism.

1432 The following exemplar defines the `CreateSequence` syntax when extended to include a  
1433 `wsse:SecurityTokenReference`:

```
1434 <wsrm:CreateSequence ...>  
1435   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1436   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1437   <wsrm:Offer ...>  
1438     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1439     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1440     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1441     <wsrm:IncompleteSequenceBehavior>  
1442       wsm:IncompleteSequenceBehaviorType  
1443     </wsrm:IncompleteSequenceBehavior> ?  
1444     ...  
1445   </wsrm:Offer> ?  
1446   ...  
1447   <wsse:SecurityTokenReference>  
1448     ...  
1449   </wsse:SecurityTokenReference> ?  
1450   ...  
1451 </wsrm:CreateSequence>
```

1452 The following describes the content model of the additional `CreateSequence` elements.

1453 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1454 This element uses the extensibility mechanism defined for the `CreateSequence` element  
1455 (defined in section 3.4) to communicate an explicit reference to the security token, using a  
1456 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and  
1457 Destination MUST use to authorize messages for the created (and, if present, the offered)  
1458 Sequence(s). All subsequent messages related to the created (and, if present, the offered)  
1459 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token  
1460 (e.g., by using or deriving from a private or secret key).

1461 When a RM Source transmits a `CreateSequence` that has been extended to include a  
1462 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1463 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include  
1464 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This  
1465 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source  
1466 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands  
1467 and conforms with the requirements listed above. Note that an RM Destination understanding this header  
1468 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined  
1469 in WS-Security still applies.

1470 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1471 <wsrm:UsesSequenceSTR ... />
```

1472 The following describes the content model of the `UsesSequenceSTR` header block.

1473 `/wsrm:UsesSequenceSTR`

1474 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that  
1475 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`  
1476 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly  
1477 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus  
1478 aborting the requested Sequence creation.

1479 The following is an example of a `CreateSequence` message using the

1480 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1481 <soap:Envelope ...>  
1482   <soap:Header>  
1483     ...  
1484     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1485     ...  
1486   </soap:Header>  
1487   <soap:Body>  
1488     <wsrm:CreateSequence>  
1489       <wsrm:AcksTo>  
1490         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1491       </wsrm:AcksTo>  
1492       <wsse:SecurityTokenReference>  
1493         ...  
1494       </wsse:SecurityTokenReference>  
1495     </wsrm:CreateSequence>  
1496   </soap:Body>  
1497 </soap:Envelope>
```

## 1498 6.2 Securing Sequences Using SSL/TLS

1499 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).  
1500 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying  
1501 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a  
1502 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a  
1503 SOAP header block within the `CreateSequence` message.

1504 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1505 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1506 The following describes the content model of the `UsesSequenceSSL` header block.

1507 `/wsrm:UsesSequenceSSL`

1508 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`  
1509 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1510 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM  
1511 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.  
1512 The receiving RM Destination MUST understand and correctly implement the functionality  
1513 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the  
1514 requested Sequence creation.

1515 Note that the inclusion of the above header by the RM Source implies that all Sequence-related  
1516 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from  
1517 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the  
1518 `CreateSequenceResponse` message.

---

## 1519 Appendix A. Schema

1520 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-  
1521 Schema Part2] is located at:

1522 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1523 The following copy is provided for reference.

```
1524 <?xml version="1.0" encoding="UTF-8"?>
1525 <!-- Copyright(C) OASIS(R) 1993-2007. All Rights Reserved.
1526 OASIS trademark, IPR and other policies apply. -->
1527 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1528 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsm="http://docs.oasis-
1529 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1530 rx/wsrn/200702" elementFormDefault="qualified"
1531 attributeFormDefault="unqualified">
1532 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1533 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1534 <!-- Protocol Elements -->
1535 <xs:complexType name="SequenceType">
1536 <xs:sequence>
1537 <xs:element ref="wsm:Identifier"/>
1538 <xs:element name="MessageNumber" type="wsm:MessageNumberType"/>
1539 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1540 maxOccurs="unbounded"/>
1541 </xs:sequence>
1542 <xs:anyAttribute namespace="##other" processContents="lax"/>
1543 </xs:complexType>
1544 <xs:element name="Sequence" type="wsm:SequenceType"/>
1545 <xs:element name="SequenceAcknowledgement">
1546 <xs:complexType>
1547 <xs:sequence>
1548 <xs:element ref="wsm:Identifier"/>
1549 <xs:choice>
1550 <xs:sequence>
1551 <xs:choice>
1552 <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1553 <xs:complexType>
1554 <xs:sequence/>
1555 <xs:attribute name="Upper" type="xs:unsignedLong"
1556 use="required"/>
1557 <xs:attribute name="Lower" type="xs:unsignedLong"
1558 use="required"/>
1559 <xs:anyAttribute namespace="##other" processContents="lax"/>
1560 </xs:complexType>
1561 </xs:element>
1562 <xs:element name="None">
1563 <xs:complexType>
1564 <xs:sequence/>
1565 </xs:complexType>
1566 </xs:element>
1567 </xs:choice>
1568 <xs:element name="Final" minOccurs="0">
1569 <xs:complexType>
1570 <xs:sequence/>
1571 </xs:complexType>
1572 </xs:element>
1573 </xs:sequence>
1574 <xs:element name="Nack" type="xs:unsignedLong"
```

```

1575 maxOccurs="unbounded" />
1576     </xs:choice>
1577     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1578 maxOccurs="unbounded" />
1579     </xs:sequence>
1580     <xs:anyAttribute namespace="##other" processContents="lax" />
1581   </xs:complexType>
1582 </xs:element>
1583 <xs:complexType name="AckRequestedType">
1584   <xs:sequence>
1585     <xs:element ref="wsrm:Identifier" />
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 name="AckRequested" type="wsrm:AckRequestedType" />
1592   <xs:element name="Identifier">
1593     <xs:complexType>
1594       <xs:annotation>
1595         <xs:documentation>
1596           This type is for elements whose [children] is an anyURI and can have
1597 arbitrary attributes.
1598         </xs:documentation>
1599       </xs:annotation>
1600       <xs:simpleContent>
1601         <xs:extension base="xs:anyURI">
1602           <xs:anyAttribute namespace="##other" processContents="lax" />
1603         </xs:extension>
1604       </xs:simpleContent>
1605     </xs:complexType>
1606   </xs:element>
1607   <xs:element name="Address">
1608     <xs:complexType>
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:simpleType name="MessageNumberType">
1617     <xs:restriction base="xs:unsignedLong">
1618       <xs:minInclusive value="1" />
1619       <xs:maxInclusive value="9223372036854775807" />
1620     </xs:restriction>
1621   </xs:simpleType>
1622   <!-- Fault Container and Codes -->
1623   <xs:simpleType name="FaultCodes">
1624     <xs:restriction base="xs:QName">
1625       <xs:enumeration value="wsrm:SequenceTerminated" />
1626       <xs:enumeration value="wsrm:UnknownSequence" />
1627       <xs:enumeration value="wsrm:InvalidAcknowledgement" />
1628       <xs:enumeration value="wsrm:MessageNumberRollover" />
1629       <xs:enumeration value="wsrm:CreateSequenceRefused" />
1630       <xs:enumeration value="wsrm:SequenceClosed" />
1631       <xs:enumeration value="wsrm:WSRMRequired" />
1632     </xs:restriction>
1633   </xs:simpleType>
1634   <xs:complexType name="SequenceFaultType">
1635     <xs:sequence>
1636       <xs:element name="FaultCode" type="wsrm:FaultCodes" />
1637       <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0" />
1638     <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1639 maxOccurs="unbounded" />
1640 </xs:sequence>
1641 <xs:anyAttribute namespace="##other" processContents="lax" />
1642 </xs:complexType>
1643 <xs:complexType name="DetailType">
1644 <xs:sequence>
1645 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1646 maxOccurs="unbounded" />
1647 </xs:sequence>
1648 <xs:anyAttribute namespace="##other" processContents="lax" />
1649 </xs:complexType>
1650 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType" />
1651 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType" />
1652 <xs:element name="CreateSequenceResponse"
1653 type="wsrm:CreateSequenceResponseType" />
1654 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType" />
1655 <xs:element name="CloseSequenceResponse"
1656 type="wsrm:CloseSequenceResponseType" />
1657 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType" />
1658 <xs:element name="TerminateSequenceResponse"
1659 type="wsrm:TerminateSequenceResponseType" />
1660 <xs:complexType name="CreateSequenceType">
1661 <xs:sequence>
1662 <xs:element ref="wsrm:AcksTo" />
1663 <xs:element ref="wsrm:Expires" minOccurs="0" />
1664 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0" />
1665 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1666 maxOccurs="unbounded">
1667 <xs:annotation>
1668 <xs:documentation>
1669 It is the authors intent that this extensibility be used to
1670 transfer a Security Token Reference as defined in WS-Security.
1671 </xs:documentation>
1672 </xs:annotation>
1673 </xs:any>
1674 </xs:sequence>
1675 <xs:anyAttribute namespace="##other" processContents="lax" />
1676 </xs:complexType>
1677 <xs:complexType name="CreateSequenceResponseType">
1678 <xs:sequence>
1679 <xs:element ref="wsrm:Identifier" />
1680 <xs:element ref="wsrm:Expires" minOccurs="0" />
1681 <xs:element name="IncompleteSequenceBehavior"
1682 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0" />
1683 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0" />
1684 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1685 maxOccurs="unbounded" />
1686 </xs:sequence>
1687 <xs:anyAttribute namespace="##other" processContents="lax" />
1688 </xs:complexType>
1689 <xs:complexType name="CloseSequenceType">
1690 <xs:sequence>
1691 <xs:element ref="wsrm:Identifier" />
1692 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1693 minOccurs="0" />
1694 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1695 maxOccurs="unbounded" />
1696 </xs:sequence>
1697 <xs:anyAttribute namespace="##other" processContents="lax" />
1698 </xs:complexType>
1699 <xs:complexType name="CloseSequenceResponseType">
1700 <xs:sequence>
1701 <xs:element ref="wsrm:Identifier" />
1702 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1703 maxOccurs="unbounded" />
1704 </xs:sequence>
1705 <xs:anyAttribute namespace="##other" processContents="lax" />
1706 </xs:complexType>
1707 <xs:complexType name="TerminateSequenceType">
1708 <xs:sequence>
1709 <xs:element ref="wsrm:Identifier" />
1710 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1711 minOccurs="0" />
1712 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1713 maxOccurs="unbounded" />
1714 </xs:sequence>
1715 <xs:anyAttribute namespace="##other" processContents="lax" />
1716 </xs:complexType>
1717 <xs:complexType name="TerminateSequenceResponseType">
1718 <xs:sequence>
1719 <xs:element ref="wsrm:Identifier" />
1720 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1721 maxOccurs="unbounded" />
1722 </xs:sequence>
1723 <xs:anyAttribute namespace="##other" processContents="lax" />
1724 </xs:complexType>
1725 <xs:element name="AcksTo" type="wsa:EndpointReferenceType" />
1726 <xs:complexType name="OfferType">
1727 <xs:sequence>
1728 <xs:element ref="wsrm:Identifier" />
1729 <xs:element name="Endpoint" type="wsa:EndpointReferenceType" />
1730 <xs:element ref="wsrm:Expires" minOccurs="0" />
1731 <xs:element name="IncompleteSequenceBehavior"
1732 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0" />
1733 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1734 maxOccurs="unbounded" />
1735 </xs:sequence>
1736 <xs:anyAttribute namespace="##other" processContents="lax" />
1737 </xs:complexType>
1738 <xs:complexType name="AcceptType">
1739 <xs:sequence>
1740 <xs:element ref="wsrm:AcksTo" />
1741 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1742 maxOccurs="unbounded" />
1743 </xs:sequence>
1744 <xs:anyAttribute namespace="##other" processContents="lax" />
1745 </xs:complexType>
1746 <xs:element name="Expires">
1747 <xs:complexType>
1748 <xs:simpleContent>
1749 <xs:extension base="xs:duration">
1750 <xs:anyAttribute namespace="##other" processContents="lax" />
1751 </xs:extension>
1752 </xs:simpleContent>
1753 </xs:complexType>
1754 </xs:element>
1755 <xs:simpleType name="IncompleteSequenceBehaviorType">
1756 <xs:restriction base="xs:string">
1757 <xs:enumeration value="DiscardEntireSequence" />
1758 <xs:enumeration value="DiscardFollowingFirstGap" />
1759 <xs:enumeration value="NoDiscard" />
1760 </xs:restriction>
1761 </xs:simpleType>
1762 <xs:element name="UsesSequenceSTR">
1763 <xs:complexType>
1764 <xs:sequence />
1765 <xs:anyAttribute namespace="##other" processContents="lax" />
1766 </xs:complexType>

```

```
1767 </xs:element>
1768 <xs:element name="UsesSequenceSSL">
1769   <xs:complexType>
1770     <xs:sequence/>
1771     <xs:anyAttribute namespace="##other" processContents="lax" />
1772   </xs:complexType>
1773 </xs:element>
1774 <xs:element name="UnsupportedElement">
1775   <xs:simpleType>
1776     <xs:restriction base="xs:QName" />
1777   </xs:simpleType>
1778 </xs:element>
1779 </xs:schema>
```



---

## 1780 Appendix B. WSDL

1781 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where  
1782 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be  
1783 present in exchanges with that endpoint.

1784 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not  
1785 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]  
1786 for a higher-level mechanism to indicate that WS-RM is engaged.

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

1788 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

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

```
1790 <?xml version="1.0" encoding="utf-8"?>
1791 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1792 OASIS trademark, IPR and other policies apply. -->
1793 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1794 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1795 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1796 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1797 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1798 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL"
1799 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL">
1800
1801   <wSDL:types>
1802     <xs:schema
1803       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1804       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1805       200702.xsd"/>
1806     </xs:schema>
1807   </wSDL:types>
1808
1809   <wSDL:message name="CreateSequence">
1810     <wSDL:part name="create" element="rm:CreateSequence"/>
1811   </wSDL:message>
1812   <wSDL:message name="CreateSequenceResponse">
1813     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1814   </wSDL:message>
1815   <wSDL:message name="CloseSequence">
1816     <wSDL:part name="close" element="rm:CloseSequence"/>
1817   </wSDL:message>
1818   <wSDL:message name="CloseSequenceResponse">
1819     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1820   </wSDL:message>
1821   <wSDL:message name="TerminateSequence">
1822     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1823   </wSDL:message>
1824   <wSDL:message name="TerminateSequenceResponse">
1825     <wSDL:part name="terminateResponse"
1826     element="rm:TerminateSequenceResponse"/>
1827   </wSDL:message>
1828
1829   <wSDL:portType name="SequenceAbstractPortType">
1830     <wSDL:operation name="CreateSequence">
1831       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1832       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1833       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1834 wsam:Action="http://docs.oasis-open.org/ws-
1835 rx/wsrn/200702/CreateSequenceResponse"/>
1836 </wsdl:operation>
1837 <wsdl:operation name="CloseSequence">
1838 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1839 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1840 <wsdl:output message="tns:CloseSequenceResponse"
1841 wsam:Action="http://docs.oasis-open.org/ws-
1842 rx/wsrn/200702/CloseSequenceResponse"/>
1843 </wsdl:operation>
1844 <wsdl:operation name="TerminateSequence">
1845 <wsdl:input message="tns:TerminateSequence"
1846 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1847 <wsdl:output message="tns:TerminateSequenceResponse"
1848 wsam:Action="http://docs.oasis-open.org/ws-
1849 rx/wsrn/200702/TerminateSequenceResponse"/>
1850 </wsdl:operation>
1851 </wsdl:portType>
1852
1853 </wsdl:definitions>
```

---

## 1854 Appendix C. Message Examples

### 1855 Appendix C.1 Create Sequence

#### 1856 Create Sequence

```
1857 <?xml version="1.0" encoding="UTF-8"?>
1858 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1859 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1860 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1861   <S:Header>
1862     <wsa:MessageID>
1863       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1864     </wsa:MessageID>
1865     <wsa:To>http://example.com/serviceB/123</wsa:To>
1866     <wsa:Action>http://docs.oasis-open.org/ws-
1867 rx/wsmr/200702/CreateSequence</wsa:Action>
1868     <wsa:ReplyTo>
1869       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1870     </wsa:ReplyTo>
1871   </S:Header>
1872   <S:Body>
1873     <wsmr:CreateSequence>
1874       <wsmr:AcksTo>
1875         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1876       </wsmr:AcksTo>
1877     </wsmr:CreateSequence>
1878   </S:Body>
1879 </S:Envelope>
```

#### 1880 Create Sequence Response

```
1881 <?xml version="1.0" encoding="UTF-8"?>
1882 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1883 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1884 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1885   <S:Header>
1886     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1887     <wsa:RelatesTo>
1888       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1889     </wsa:RelatesTo>
1890     <wsa:Action>
1891       http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1892     </wsa:Action>
1893   </S:Header>
1894   <S:Body>
1895     <wsmr:CreateSequenceResponse>
1896       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1897     </wsmr:CreateSequenceResponse>
1898   </S:Body>
1899 </S:Envelope>
```

### 1900 Appendix C.2 Initial Transmission

1901 The following example WS-ReliableMessaging headers illustrate the message exchange in the above  
1902 figure. The three messages have the following headers; the third message is identified as the last  
1903 message in the Sequence:

## 1904 Message 1

```
1905 <?xml version="1.0" encoding="UTF-8"?>
1906 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1907 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1908 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1909   <S:Header>
1910     <wsa:MessageID>
1911       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1912     </wsa:MessageID>
1913     <wsa:To>http://example.com/serviceB/123</wsa:To>
1914     <wsa:From>
1915       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1916     </wsa:From>
1917     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1918     <wsmr:Sequence>
1919       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1920       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1921     </wsmr:Sequence>
1922   </S:Header>
1923   <S:Body>
1924     <!-- Some Application Data -->
1925   </S:Body>
1926 </S:Envelope>
```

## 1927 Message 2

```
1928 <?xml version="1.0" encoding="UTF-8"?>
1929 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1930 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1931 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1932   <S:Header>
1933     <wsa:MessageID>
1934       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1935     </wsa:MessageID>
1936     <wsa:To>http://example.com/serviceB/123</wsa:To>
1937     <wsa:From>
1938       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1939     </wsa:From>
1940     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1941     <wsmr:Sequence>
1942       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1943       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1944     </wsmr:Sequence>
1945   </S:Header>
1946   <S:Body>
1947     <!-- Some Application Data -->
1948   </S:Body>
1949 </S:Envelope>
```

## 1950 Message 3

```
1951 <?xml version="1.0" encoding="UTF-8"?>
1952 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1953 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1954 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1955   <S:Header>
1956     <wsa:MessageID>
1957       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1958     </wsa:MessageID>
1959     <wsa:To>http://example.com/serviceB/123</wsa:To>
1960     <wsa:From>
1961       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1962     </wsa:From>
1963     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
```

```

1964 <wsrm:Sequence>
1965   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1966   <wsrm:MessageNumber>3</wsrm:MessageNumber>
1967 </wsrm:Sequence>
1968 <wsrm:AckRequested>
1969   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1970 </wsrm:AckRequested>
1971 </S:Header>
1972 <S:Body>
1973   <!-- Some Application Data -->
1974 </S:Body>
1975 </S:Envelope>

```

## 1976 **Appendix C.3 First Acknowledgement**

1977 Message number 2 has not been accepted by the RM Destination due to some transmission error so it  
1978 responds with an Acknowledgement for messages 1 and 3:

```

1979 <?xml version="1.0" encoding="UTF-8"?>
1980 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1981 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1982 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1983   <S:Header>
1984     <wsa:MessageID>
1985       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1986     </wsa:MessageID>
1987     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1988     <wsa:From>
1989       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1990     </wsa:From>
1991     <wsa:Action>
1992       http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1993     </wsa:Action>
1994     <wsrm:SequenceAcknowledgement>
1995       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1996       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1997       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1998     </wsrm:SequenceAcknowledgement>
1999   </S:Header>
2000   <S:Body/>
2001 </S:Envelope>

```

## 2002 **Appendix C.4 Retransmission**

2003 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and  
2004 requests an Acknowledgement:

```

2005 <?xml version="1.0" encoding="UTF-8"?>
2006 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2007 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2008 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2009   <S:Header>
2010     <wsa:MessageID>
2011       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2012     </wsa:MessageID>
2013     <wsa:To>http://example.com/serviceB/123</wsa:To>
2014     <wsa:From>
2015       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2016     </wsa:From>
2017     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2018   <wsrm:Sequence>

```

```

2019     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2020     <wsrm:MessageNumber>2</wsrm:MessageNumber>
2021     </wsrm:Sequence>
2022     <wsrm:AckRequested>
2023     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2024     </wsrm:AckRequested>
2025     </S:Header>
2026     <S:Body>
2027     <!-- Some Application Data -->
2028     </S:Body>
2029     </S:Envelope>

```

## 2030 Appendix C.5 Termination

2031 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then  
2032 be terminated:

```

2033 <?xml version="1.0" encoding="UTF-8"?>
2034 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2035 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2036 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2037   <S:Header>
2038     <wsa:MessageID>
2039       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2040     </wsa:MessageID>
2041     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2042     <wsa:From>
2043       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2044     </wsa:From>
2045     <wsa:Action>
2046       http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2047     </wsa:Action>
2048     <wsrm:SequenceAcknowledgement>
2049       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2050       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2051     </wsrm:SequenceAcknowledgement>
2052   </S:Header>
2053   <S:Body/>
2054 </S:Envelope>

```

## 2055 Terminate Sequence

```

2056 <?xml version="1.0" encoding="UTF-8"?>
2057 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2058 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2059 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2060   <S:Header>
2061     <wsa:MessageID>
2062       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2063     </wsa:MessageID>
2064     <wsa:To>http://example.com/serviceB/123</wsa:To>
2065     <wsa:Action>
2066       http://docs.oasis-open.org/ws-rx/wsr/200702/TerminateSequence
2067     </wsa:Action>
2068     <wsa:From>
2069       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2070     </wsa:From>
2071   </S:Header>
2072   <S:Body>
2073     <wsrm:TerminateSequence>
2074       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2075       <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2076     </wsrm:TerminateSequence>

```

```
2077 </S:Body>
2078 </S:Envelope>
```

## 2079 Terminate Sequence Response

```
2080 <?xml version="1.0" encoding="UTF-8"?>
2081 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2082 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2083 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2084 <S:Header>
2085 <wsa:MessageID>
2086 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2087 </wsa:MessageID>
2088 <wsa:To>http://example.com/serviceA/789</wsa:To>
2089 <wsa:Action>
2090 http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
2091 </wsa:Action>
2092 <wsa:RelatesTo>
2093 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2094 </wsa:RelatesTo>
2095 <wsa:From>
2096 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2097 </wsa:From>
2098 </S:Header>
2099 <S:Body>
2100 <wsm:TerminateSequenceResponse>
2101 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2102 </wsm:TerminateSequenceResponse>
2103 </S:Body>
2104 </S:Envelope>
```

---

## 2105 Appendix D. State Tables

2106 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2107 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

2108 Legend:

2109 The first column of these tables contains the motivating event and has the following format:

<b>Event</b> 2110
<i>Event name</i> [source]  {ref}

2111 Where:

- 2112 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as  
2113 described by the specification.
- 2114 • [source]: indicates the source of the event; one of:
  - 2115 ○ [msg] a Received message
  - 2116 ○ [int]: an internal event such as the firing of a timer
  - 2117 ○ [app]: the application
  - 2118 ○ [unspec]: the source is unspecified

2119 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}

2120 Where:

- 2121 • action to take: indicates that the state machine performs the following action. Actions surrounded  
2122 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word  
2123 "Transmit"
  - 2124 • [next state]: indicates the state to which the state machine will advance upon the performance of  
2125 the action. For ease of reading the next state "same" indicates that the state does not change.
  - 2126 • {ref} is a reference to the document section describing the behavior in this cell
- 2127 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these  
2128 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not  
2129 described in this specification and does not indicate normal protocol operation. Implementations MAY  
2130 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations  
2131 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state  
2132 combinations.



2133 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}

2134 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}	

---

## 2135 Appendix E. Acknowledgments

2136 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following  
2137 authors:

2138	Ruslan Bilorusets, BEA	2150	Amelia Lewis, TIBCO Software
2139	Don Box, Microsoft	2151	Rodney Limprecht, Microsoft
2140	Luis Felipe Cabrera, Microsoft	2152	Steve Lucco, Microsoft
2141	Doug Davis, IBM	2153	Don Mullen, TIBCO Software
2142	Donald Ferguson, IBM	2154	Anthony Nadalin, IBM
2143	Christopher Ferris, IBM	2155	Mark Nottingham, BEA
2144	Tom Freund, IBM	2156	David Orchard, BEA
2145	Mary Ann Hondo, IBM	2157	Jamie Roots, IBM
2146	John Ibbotson, IBM	2158	Shivajee Samdarshi, TIBCO Software
2147	Lei Jin, BEA	2159	John Shewchuk, Microsoft
2148	Chris Kaler, Microsoft	2160	Tony Storey, IBM
2149	David Langworthy-Editor, Microsoft		

2161 The following individuals have provided invaluable input into the initial contribution:

2162	Keith Ballinger, Microsoft	2176	David Ingham, Microsoft
2163	Stefan Batres, Microsoft	2177	Gopal Kakivaya, Microsoft
2164	Rebecca Bergersen, Iona	2178	Johannes Klein, Microsoft
2165	Allen Brown, Microsoft	2179	Frank Leymann, IBM
2166	Michael Conner, IBM	2180	Martin Nally, IBM
2167	George Copeland, Microsoft	2181	Peter Niblett, IBM
2168	Francisco Curbera, IBM	2182	Jeffrey Schlimmer, Microsoft
2169	Paul Fremantle, IBM	2183	James Snell, IBM
2170	Steve Graham, IBM	2184	Keith Stobie, Microsoft
2171	Pat Helland, Microsoft	2185	Satish Thatte, Microsoft
2172	Rick Hill, Microsoft	2186	Stephen Todd, IBM
2173	Scott Hinkelman, IBM	2187	Sanjiva Weerawarana, IBM
2174	Tim Holloway, IBM	2188	Roger Wolter, Microsoft
2175	Efim Hudis, Microsoft		

2189 The following individuals were members of the committee during the development of this specification:

2190	Abbie Barbir, Nortel	2209	Robert Freund, Hitachi
2191	Charlton Barreto, Adobe	2210	Peter Furniss, Erebore
2192	Stefan Batres, Microsoft	2211	Marc Goodner, Microsoft
2193	Hamid Ben Malek, Fujitsu	2212	Alastair Green, Choreology
2194	Andreas Bjarlestam, Ericsson	2213	Mike Grogan, Sun
2195	Toufic Boubez, Layer 7	2214	Ondrej Hrebicek, Microsoft
2196	Doug Bunting, Sun	2215	Kazunori Iwasa, Fujitsu
2197	Lloyd Burch, Novell	2216	Chamikara Jayalath, WSO2
2198	Steve Carter, Novell	2217	Lei Jin, BEA
2199	Martin Chapman, Oracle	2218	Ian Jones, BTplc
2200	Dave Chappell, Sonic	2219	Anish Karmarkar, Oracle
2201	Paul Cotton, Microsoft	2220	Paul Knight, Nortel
2202	Glen Daniels, Sonic	2221	Dan Leshchiner, Tibco
2203	Doug Davis, IBM	2222	Mark Little, JBoss
2204	Blake Dournaee, Intel	2223	Lily Liu, webMethods
2205	Jacques Durand, Fujitsu	2224	Matt Lovett, IBM
2206	Colleen Evans, Microsoft	2225	Ashok Malhotra, Oracle
2207	Christopher Ferris, IBM	2226	Jonathan Marsh, Microsoft
2208	Paul Fremantle, WSO2	2227	Daniel Millwood, IBM

2228	Jeff Mischkinsky, Oracle	2239	Stefan Rossmann, SAP
2229	Nilo Mitra, Ericsson	2240	Tom Rutt, Fujitsu
2230	Peter Niblett, IBM	2241	Rich Salz, IBM
2231	Duane Nickull, Adobe	2242	Shivajee Samdarshi, Tibco
2232	Eisaku Nishiyama, Hitachi	2243	Vladimir Videlov, SAP
2233	Dave Orchard, BEA	2244	Claus von Riegen, SAP
2234	Chouthri Palanisamy, NEC	2245	Pete Wenzel, Sun
2235	Sanjay Patil, SAP	2246	Steve Winkler, SAP
2236	Gilbert Pilz, BEA	2247	Ümit Yalçinalp, SAP
2237	Martin Raepfle, SAP	2248	Nobuyuki Yamamoto, Hitachi
2238	Eric Rajkovic, Oracle		
2249			