



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

3 Committee Specification 02

4 29 November 2008

5 Specification URIs:

6 This Version:

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

10 Previous Version:

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

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>
22 Sanjay Patil <sanjay.patil@sap.com>

23 Editors:

24 Doug Davis, IBM <dug@us.ibm.com>
25 Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>
26 Gilbert Pilz, BEA <gpilz@bea.com>
27 Steve Winkler, SAP <steve.winkler@sap.com>
28 Ümit Yalçınalp, SAP <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](http://www.oasis-open.org/who/trademark.php), the owner
105 and developer of this specification, and should be used only to refer to the organization and its official
106 outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the
107 right to enforce its marks against misleading uses. Please see [http://www.oasis-](http://www.oasis-open.org/who/trademark.php)
108 [open.org/who/trademark.php](http://www.oasis-open.org/who/trademark.php) for above guidance.

109 Table of Contents

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

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

158 1 Introduction

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

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

168 1.1 Terminology

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

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

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

188 Elements and Attributes defined by this specification are referred to in the text of this document using
189 XPath 1.0 [XPath_10] expressions. Extensibility points are referred to using an extended version of this
190 syntax:

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

197 1.2 Normative References

- 198 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC
199 2119, Harvard University, March 1997
200 <http://www.ietf.org/rfc/rfc2119.txt>
- 201 **[WS-RM Policy]** OASIS Committee Specification 02, "Web Services Reliable Messaging Policy
202 Assertion(WS-RM Policy)," November 2008
203 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.2-spec-cs-02.doc>
- 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 **[RDDDL 2.0]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language
241 (RDDDL) 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-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
 260 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

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 Committee Specification 01, "WS-SecurityPolicy 1.3", November 2008
 269 [http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cs/ws-securitypolicy-1.3-](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cs/ws-securitypolicy-1.3-spec-cs-01.doc)
 270 [spec-cs-01.doc](http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/cs/ws-securitypolicy-1.3-spec-cs-01.doc)

271 **[SecureConversation]** OASIS Committee Specification 01, "WS-SecureConversation 1.4",
 272 November 2008
 273 [http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/cs/ws-](http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/cs/ws-secureconversation-1.4-spec-cs-01.doc)
 274 [secureconversation-1.4-spec-cs-01.doc](http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/cs/ws-secureconversation-1.4-spec-cs-01.doc)

275 **[Trust]** OASIS Committee Specification 01, "WS-Trust 1.4", November 2008
 276 <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/cs/ws-trust-1.4-spec-cs-01.doc>

277 1.4 Namespace

278 The XML namespace [[XML-ns](#)] URI that MUST be used by implementations of this specification is:

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

280 Dereferencing the above URI will produce the Resource Directory Description Language [[RDDL 2.0](#)]
 281 document that describes this namespace.

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

284 Table 1

Prefix	Namespace
S	(Either SOAP 1.1 or 1.2)
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
wsrn	http://docs.oasis-open.org/ws-rx/wsrn/200702
wsa	http://www.w3.org/2005/08/addressing

wsam	http://www.w3.org/2007/05/addressing/metadata
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

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

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

288 **1.5 Conformance**

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

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

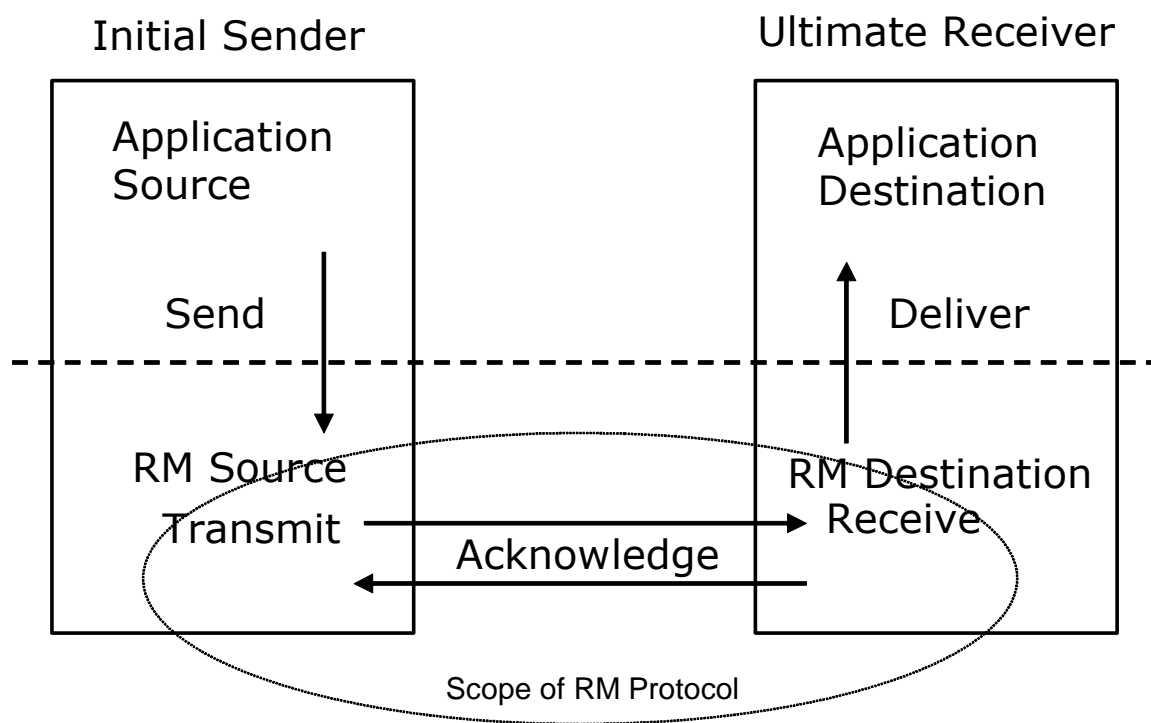
295 **2 Reliable Messaging Model**

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

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

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

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



321 Figure 1: Reliable Messaging Model

322 2.1 Glossary

323 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

353 2.2 Protocol Preconditions

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

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

363 2.3 Protocol Invariants

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

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

377 2.4 Delivery Assurances

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

381 `AtLeastOnce`

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

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

389 AtMostOnce

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

394 ExactlyOnce

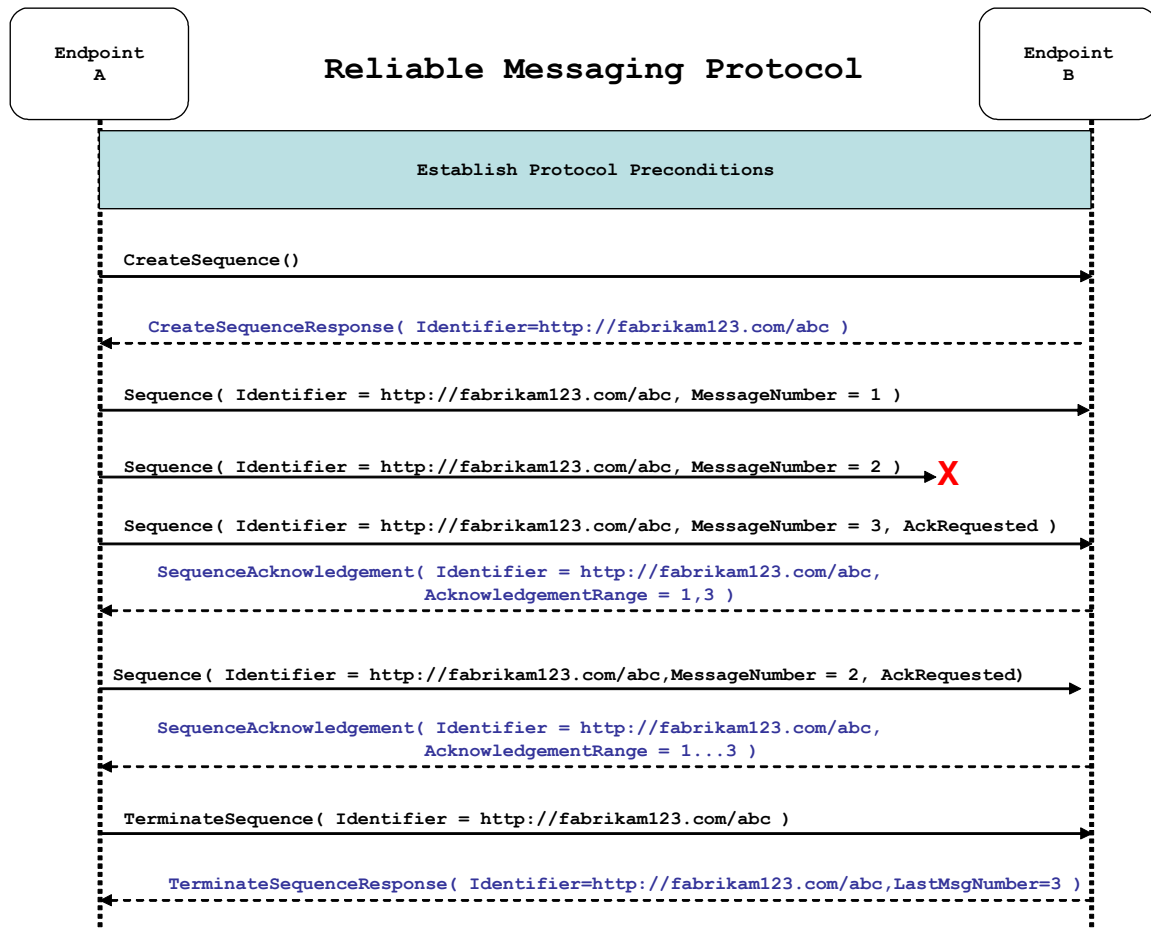
395 Each message is to be delivered exactly once; if a message cannot be delivered then an error
396 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is
397 that it SHOULD retry transmission of every message sent by the Application Source until it
398 receives an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

402 InOrder

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

414 2.5 Example Message Exchange

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



416 Figure 2: The WS-ReliableMessaging Protocol

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

434 9. The RM Destination Receives the second transmission of the message with `MessageNumber 2`
435 and acknowledges receipt of message numbers 1, 2, and 3.

436 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to
437 the RM Destination indicating that the Sequence is completed. The `TerminateSequence`
438 message indicates that message number 3 was the last message in the Sequence. The RM
439 Destination then reclaims any resources associated with the Sequence.

440 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source
441 will not be sending any more messages. The RM Destination sends a
442 `TerminateSequenceResponse` message to the RM Source and reclaims any resources
443 associated with the Sequence.

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

455 Now that the basic model has been outlined, the details of the elements used in this protocol are now
456 provided in section 3.

457 3 RM Protocol Elements

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

460 3.1 Considerations on the Use of Extensibility Points

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

465 3.2 Considerations on the Use of "Piggy-Backing"

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

475 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

493 3.4 Sequence Creation

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

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

501 The following exemplar defines the `CreateSequence` syntax:

```
502 <wsmr:CreateSequence ...>
503   <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>
504   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?
505   <wsmr:Offer ...>
506     <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
507     <wsmr:Endpoint> wsa:EndpointReferenceType </wsmr:Endpoint>
508     <wsmr:Expires ...> xs:duration </wsmr:Expires> ?
509     <wsmr:IncompleteSequenceBehavior>
510       wsmr:IncompleteSequenceBehaviorType
511     </wsmr:IncompleteSequenceBehavior> ?
512     ...
513   </wsmr:Offer> ?
514   ...
515 </wsmr:CreateSequence>
```

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

517 `/wsmr:CreateSequence`

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

522 `/wsmr:CreateSequence/wsmr:AcksTo`

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

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

532 `/wsmr:CreateSequence/wsmr:Expires`

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

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

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

540 /wsmr:CreateSequence/wsmr:Offer

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

543 /wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier

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

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

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

549 /wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint

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

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

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

566 /wsmr:CreateSequence/wsmr:Offer/wsmr:Expires

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

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

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

573 /wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior

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

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

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

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

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

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

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

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

592 /wsmr:CreateSequence/{any}

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

595 /wsmr:CreateSequence/@{any}

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

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

602 The following exemplar defines the *CreateSequenceResponse* syntax:

```
603 <wsmr:CreateSequenceResponse ...>  
604   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
605   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
606   <wsmr:IncompleteSequenceBehavior>  
607     wsmr:IncompleteSequenceBehaviorType  
608   </wsmr:IncompleteSequenceBehavior> ?  
609   <wsmr:Accept ...>  
610     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
611     ...  
612   </wsmr:Accept> ?  
613   ...  
614 </wsmr:CreateSequenceResponse>
```

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

616 /wsmr:CreateSequenceResponse

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

620 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

628 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

641 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

654 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

660 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

675 /wsrm:CreateSequenceResponse/{any}

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

678 /wsrm:CreateSequenceResponse/@{any}

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

681 3.5 Closing A Sequence

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

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

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

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

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

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

715 The following exemplar defines the `CloseSequence` syntax:

```
716 <wsrm:CloseSequence ...>  
717   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
718   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?
```

```
719     ...
720 </wsmr:CloseSequence>
```

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

722 `/wsmr:CloseSequence`

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

726 `/wsmr:CloseSequence/wsmr:Identifier`

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

730 `/wsmr:CloseSequence/wsmr:LastMsgNumber`

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

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

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

737 `/wsmr:CloseSequence/{any}`

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

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

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

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

745 The following exemplar defines the `CloseSequenceResponse` syntax:

```
746 <wsmr:CloseSequenceResponse ...>
747   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
748   ...
749 </wsmr:CloseSequenceResponse>
```

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

751 `/wsmr:CloseSequenceResponse`

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

754 `/wsmr:CloseSequenceResponse/wsmr:Identifier`

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

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

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

761 /wsmr:CloseSequenceResponse/{any}

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

764 /wsmr:CloseSequenceResponse/@{any}

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

767 3.6 Sequence Termination

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

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

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

786 The following exemplar defines the `TerminateSequence` syntax:

```
787 <wsmr:TerminateSequence ...>  
788   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
789   <wsmr>LastMsgNumber> wsmr:MessageNumberType </wsmr>LastMsgNumber> ?  
790   ...  
791 </wsmr:TerminateSequence>
```

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

793 /wsmr:TerminateSequence

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

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

805 /wsmr:TerminateSequence/wsmr:Identifier

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

809 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

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

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

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

816 `/wsrm:TerminateSequence/{any}`

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

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

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

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

824 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
825 <wsrm:TerminateSequenceResponse ...>  
826   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
827   ...  
828 </wsrm:TerminateSequenceResponse>
```

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

830 `/wsrm:TerminateSequenceResponse`

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

834 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

841 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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

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

850 3.7 Sequences

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

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

858 A following exemplar defines its syntax:

```
859 <wsm:Sequence ...>  
860   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
861   <wsm:MessageNumber> wsm:MessageNumberType </wsm:MessageNumber>  
862   ...  
863 </wsm:Sequence>
```

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

865 `/wsm:Sequence`

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

872 `/wsm:Sequence/wsm:Identifier`

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

876 `/wsm:Sequence/wsm:Identifier/@{any}`

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

879 `/wsm:Sequence/wsm:MessageNumber`

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

884 `/wsm:Sequence/{any}`

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

887 `/wsm:Sequence/@{any}`

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

890 The following example illustrates a `Sequence` header block.

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

895 3.8 Request Acknowledgement

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

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

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

910 The following exemplar defines its syntax:

```
911 <wsrm:AckRequested ...>
912   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
913   ...
914 </wsrm:AckRequested>
```

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

916 `/wsrm:AckRequested`

917 This element requests an Acknowledgement for the identified `Sequence`.

918 `/wsrm:AckRequested/wsrm:Identifier`

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

923 `/wsrm:AckRequested/wsrm:Identifier/@{any}`

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

926 `/wsrm:AckRequested/{any}`

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

929 `/wsrm:AckRequested/@{any}`

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

932 3.9 Sequence Acknowledgement

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

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

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

953 The following exemplar defines its syntax:

```
954 <wsrm:SequenceAcknowledgement ...>  
955   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
956   [ [ [ <wsrm:AcknowledgementRange ...  
957     Upper="wsrm:MessageNumberType"  
958     Lower="wsrm:MessageNumberType"/> +  
959     | <wsrm:None/> ]  
960     <wsrm:Final/> ? ]  
961     | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]  
962   ]  
963   ...  
964 </wsrm:SequenceAcknowledgement>
```

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

966 `/wsrm:SequenceAcknowledgement`

967 This element contains the Sequence Acknowledgement information.

968 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

977 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

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

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

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

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

997 `/wsrm:SequenceAcknowledgement/wsrm:Final`
998 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
999 block. This element indicates that the RM Destination is not receiving new messages for the
1000 specified Sequence. The RM Source can be assured that the ranges of messages acknowledged
1001 by this `SequenceAcknowledgement` header block will not change in the future. The RM
1002 Destination MUST include this element when the Sequence is closed. The RM Destination MUST
1003 NOT include this element when sending a `Nack`; it can only be used when sending
1004 `AcknowledgementRange` elements or a `None`.

1005 `/wsrm:SequenceAcknowledgement/wsrm:Nack`
1006 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1007 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`
1008 representing the `MessageNumber` of an unreceived message in a Sequence. The RM Destination
1009 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is
1010 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM
1011 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT
1012 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously
1013 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a
1014 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been
1015 acknowledged within an `AcknowledgementRange`.

1016 `/wsrm:SequenceAcknowledgement/{any}`
1017 This is an extensibility mechanism to allow different (extensible) types of information, based on a
1018 schema, to be passed.

1019 `/wsrm:SequenceAcknowledgement/@{any}`
1020 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1021 to the element.

1022 The following examples illustrate `SequenceAcknowledgement` elements:

- 1023 • Message numbers 1...10 inclusive in a Sequence have been accepted by the RM Destination.

```
1024 <wsrm:SequenceAcknowledgement>  
1025   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1026   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
1027 </wsrm:SequenceAcknowledgement>
```

- 1028 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM
1029 Destination, messages 3 and 7 have not been accepted.

```
1030 <wsrm:SequenceAcknowledgement>  
1031   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1032   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1033   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1034   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1035 </wsrm:SequenceAcknowledgement>
```

- 1036 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1037 <wsrm:SequenceAcknowledgement>  
1038   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1039   <wsrm:Nack>3</wsrm:Nack>  
1040 </wsrm:SequenceAcknowledgement>
```

1041 4 Faults

1042 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. `Create`
1043 `Sequence Refused` is a possible fault reply for this operation. `Unknown Sequence` is a fault generated by
1044 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences
1045 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM
1046 on a Received message that did not use the protocol. All other faults in this section relate to known
1047 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those faults.
1048 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

1049 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault
1050 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1051 `http://docs.oasis-open.org/ws-rx/wsrp/200702/fault`

1052 The faults defined in this section are generated if the condition stated in the preamble is met. Fault
1053 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1054 The definitions of faults use the following properties:

1055 [Code] The fault code.

1056 [Subcode] The fault subcode.

1057 [Reason] The English language reason element.

1058 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail
1059 element is defined for a fault, implementations MUST include the elements in the order that they are
1060 specified.

1061 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
1062 "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

1063 The properties above bind to a SOAP 1.2 fault as follows:

```
1064 <S:Envelope>
1065   <S:Header>
1066     <wsa:Action>
1067       http://docs.oasis-open.org/ws-rx/wsrp/200702/fault
1068     </wsa:Action>
1069     <!-- Headers elided for brevity. -->
1070   </S:Header>
1071   <S:Body>
1072     <S:Fault>
1073       <S:Code>
1074         <S:Value> [Code] </S:Value>
1075         <S:Subcode>
1076           <S:Value> [Subcode] </S:Value>
1077         </S:Subcode>
1078       </S:Code>
1079       <S:Reason>
1080         <S:Text xml:lang="en"> [Reason] </S:Text>
1081       </S:Reason>
1082       <S:Detail>
1083         [Detail]
```

1084
1085
1086
1087
1088

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

1089 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM
1090 header block:

1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106

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

1107 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
1108 CreateSequence request message:

1109
1110
1111
1112
1113
1114
1115
1116

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

1117 4.1 SequenceFault Element

1118 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the
1119 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging
1120 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.
1121 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the
1122 SOAP 1.2 binding.

1123 The following exemplar defines its syntax:

1124
1125
1126
1127
1128

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

1129 The following describes the content model of the `SequenceFault` element.

1130 /wsrm:SequenceFault

1131 This is the element containing Sequence fault information for WS-ReliableMessaging

1132 /wsrm:SequenceFault/wsrm:FaultCode

- 1133 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this
 1134 element to a qualified name from the set of faults [Subcodes] defined below.
- 1135 `/wsrm:SequenceFault/wsrm:Detail`
 1136 This element, if present, carries application specific error information related to the fault being
 1137 described.
- 1138 `/wsrm:SequenceFault/wsrm:Detail/{any}`
 1139 The application specific error information related to the fault being described.
- 1140 `/wsrm:SequenceFault/wsrm:Detail/@{any}`
 1141 The application specific error information related to the fault being described.
- 1142 `/wsrm:SequenceFault/{any}`
 1143 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 1144 schema, to be passed.
- 1145 `/wsrm:SequenceFault/@{any}`
 1146 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 1147 to the element.

1148 4.2 Sequence Terminated

- 1149 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 1150 Endpoint of this decision.
- 1151 Properties:
- 1152 [Code] Sender or Receiver
- 1153 [Subcode] `wrm:SequenceTerminated`
- 1154 [Reason] The Sequence has been terminated due to an unrecoverable error.
- 1155 [Detail]

1156 `<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.

1157 4.3 Unknown Sequence

- 1158 Properties:
- 1159 [Code] Sender
- 1160 [Subcode] `wrm:UnknownSequence`

1161 [Reason] The value of `wsrc:Identifier` is not a known Sequence identifier.

1162 [Detail]

1163 `<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.

1164 4.4 Invalid Acknowledgement

1165 An example of when this fault is generated is when a message is Received by the RM Source containing
1166 a `SequenceAcknowledgement` covering messages that have not been sent.

1167 [Code] Sender

1168 [Subcode] `wsrc:InvalidAcknowledgement`

1169 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1170 [Detail]

1171 `<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.

1172 4.5 Message Number Rollover

1173 If the condition listed below is reached, the RM Destination MUST generate this fault.

1174 Properties:

1175 [Code] Sender

1176 [Subcode] wsrn:MessageNumberRollover

1177 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1178 [Detail]

1179 `<wsrn:Identifier ...> xs:anyURI </wsrn:Identifier>`

1180 `<wsrn:MaxMessageNumber> wsrn:MessageNumberType </wsrn:MaxMessageNumber>`

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

1181 4.6 Create Sequence Refused

1182 Properties:

1183 [Code] Sender or Receiver

1184 [Subcode] wsrn:CreateSequenceRefused

1185 [Reason] The Create Sequence request has been refused by the RM Destination.

1186 [Detail]

1187 `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.

1188 4.7 Sequence Closed

1189 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1190 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1191 is closed.

1192 Properties:

1193 [Code] Sender

1194 [Subcode] wsrn:SequenceClosed

1195 [Reason] The Sequence is closed and cannot accept new messages.

1196 [Detail]

1197 `<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.

1198 **4.8 WSRM Required**

1199 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1200 message that did not use this protocol.

1201 Properties:

1202 [Code] Sender

1203 [Subcode] wsrn:WSRMRequired

1204 [Reason] The RM Destination requires the use of WSRM.

1205 [Detail]

1206 `xs:any`

1207 5 Security Threats and Countermeasures

1208 This specification considers two sets of security requirements, those of the applications that use the WS-
1209 RM protocol and those of the protocol itself.

1210 This specification makes no assumptions about the security requirements of the applications that use WS-
1211 RM. However, once those requirements have been satisfied within a given operational context, the
1212 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1213 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1214 There are many other security concerns that one may need to consider when implementing or using this
1215 protocol. The material below should not be considered as a "check list". Implementers and users of this
1216 protocol are urged to perform a security analysis to determine their particular threat profile and the
1217 appropriate responses to those threats.

1218 Implementers are also advised that there is a core tension between security and reliable messaging that
1219 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1220 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1221 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1222 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1223 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1224 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1225 avoid and prevent this condition.

1226 5.1 Threats and Countermeasures

1227 The primary security requirement of this protocol is to protect the specified semantics and protocol
1228 invariants against various threats. The following sections describe several threats to the integrity and
1229 operation of this protocol and provide some general outlines of countermeasures to those threats.
1230 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1231 to all operational contexts.

1232 5.1.1 Integrity Threats

1233 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1234 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1235 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1236 to its intended message represents a threat to the WS-RM protocol.

1237 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM
1238 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1239 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be
1240 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1241 5.1.1.1 Countermeasures

1242 Integrity threats are generally countered via the use of digital signatures some level of the communication
1243 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1244 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers
1245 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in
1246 which they occur, implementations MUST allow for signatures that cover only these headers.

1247 **5.1.2 Resource Consumption Threats**

1248 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1249 implement that RM Destination. These resources can include network connections, database tables,
1250 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1251 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1252 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1253 message Delivery and use this Sequence to send a stream of very large messages to that service, making
1254 sure to omit message number “1” from that stream.

1255 **5.1.2.1 Countermeasures**

1256 There are a number of countermeasures against the described resource consumption threats. The
1257 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1258 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1259 some cases, allows the identity of any attackers to be determined.

1260 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and
1261 authenticate the RM Source that issued the `CreateSequence` message.

1262 **5.1.3 Sequence Spoofing Threats**

1263 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1264 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1265 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1266 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the
1267 current `MessageNumber` for their target Sequence.

1268 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP
1269 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence
1270 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to
1271 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a
1272 message that it sends to the `AcksTo` EPR of an RM Source.

1273 **5.1.3.1 Sequence Hijacking**

1274 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject
1275 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1276 messages.

1277 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a
1278 Sequence may be bound to some form of a security session in order to counter the threats described in
1279 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1280 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1281 established by the security infrastructure to make such determinations. Failure to observe this rule
1282 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1283 the ability to authenticate its peers even though the necessary security processing has taken place.

1284 **5.1.3.2 Countermeasures**

1285 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by
1286 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM
1287 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination
1288 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

1289 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it
1290 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the
1291 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it
1292 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the
1293 referenced Sequence.

1294 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and
1295 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify
1296 its Sequence peer it MUST be able to identify and authenticate the entity that sent the
1297 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1298 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that
1299 message and, if necessary, correlate this identity with the Sequence peer identity established at
1300 Sequence creation time.

1301 **5.2 Security Solutions and Technologies**

1302 The security threats described in the previous sections are neither new nor unique. The solutions that
1303 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1304 section maps the facilities provided by common web services security solutions against countermeasures
1305 described in the previous sections.

1306 Before continuing this discussion, however, some examination of the underlying requirements of the
1307 previously described countermeasures is necessary. Specifically it should be noted that the technique
1308 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1309 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization
1310 check against this authenticated identity and determines if the RM Source is permitted to create
1311 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1312 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1313 discussion of such facilities is considered to be beyond the scope of this specification.

1314 **5.2.1 Transport Layer Security**

1315 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1316 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1317 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1318 The description provided here is general in nature and is not intended to serve as a complete definition on
1319 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1320 choice of features as well as the manner in which they will be used. The mechanisms described in the
1321 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1322 requirements and constraints of the use of SSL/TLS.

1323 **5.2.1.1 Model**

1324 The basic model for using SSL/TLS is as follows:

- 1325 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1326 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1327 Destination.
- 1328 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1329 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1330 synchronous `CreateSequenceResponse` using the session established in (1).

- 1331 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1332 any and all messages or faults that refer to that Sequence.
- 1333 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1334 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1335 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1336 5.2.1.2 Countermeasure Implementation

1337 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1338 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the
1339 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1340 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1341 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1342 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification
1343 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1344 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1345 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1346 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1347 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1348 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1349 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1350 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1351 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1352 Acknowledgement) using BasicAuth.
- 1353 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1354 connection authenticates itself to the party accepting the connection using an X.509 certificate
1355 that is exchanged during the SSL/TLS handshake.

1356 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1357 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1358 Source is authorized to create a Sequence with the RM Destination.

1359 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1360 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1361 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1362 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1363 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1364 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1365 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1366 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1367 to protect that Sequence.

1368 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1369 countermeasures (such as associating specific authentication information with a Sequence) although such
1370 methods are not covered by this document.

1371 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1372 session) are outside the scope of this specification.

1373 5.2.2 SOAP Message Security

1374 The mechanisms described in WS-Security may be used in various ways to implement the
1375 countermeasures described in the previous sections. This specification advocates using the protocol
1376 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1377 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1378 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1379 The description provided here is general in nature and is not intended to serve as a complete definition on
1380 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1381 need to agree on the choice of features as well as the manner in which they will be used. The
1382 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1383 describe the requirements and constraints of the use of WS-SecureConversation.

1384 **5.2.2.1 Model**

1385 The basic model for using WS-SecureConversation is as follows:

- 1386 1 The RM Source and the RM Destination create a WS-SecureConversation security context. This
1387 may involve the participation of third parties such as a security token service. The tokens
1388 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service
1389 tickets).
- 1390 2 During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1391 context that will be used to protect the Sequence. This is done so that, in cases where the
1392 `CreateSequence` message is signed by more than one security context, the RM Source can
1393 indicate which security context should be used to protect the newly created Sequence.
- 1394 3 For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1395 associated with the security context to sign (as defined by WS-Security) at least the body and
1396 any relevant WS-RM-defined headers of any and all messages or faults that refer to that
1397 Sequence.

1398 **5.2.2.2 Countermeasure Implementation**

1399 Without relying upon any authentication information, the per-message signatures provide the necessary
1400 integrity qualities to counter the threats described in section 5.1.1.

1401 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1402 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1403 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1404 create a Sequence with the RM Destination.

1405 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1406 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1407 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1408 context rather than on any authentication claims that may have been established during security context
1409 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1410 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1411 document.

1412 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits
1413 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the
1414 association between a Sequence and its protecting security context cannot always be established
1415 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1416 `CreateSequenceResponse` messages may be signed by more than one security context.

1417 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1418 amending or renewing contexts) are outside the scope of this specification.

1419 6 Securing Sequences

1420 As noted in section 5, the RM Source and RM Destination should be able to protect their shared
1421 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1422 achieving this objective depending upon the underlying security infrastructure.

1423 6.1 Securing Sequences Using WS-Security

1424 One mechanism for protecting a Sequence is to include a security token using a
1425 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1426 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1427 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1428 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1429 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1430 there may be more than one token on a `CreateSequence` message or inferred from the communication
1431 context (e.g. transport protection).

1432 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1433 if the token being referenced supports such mechanism.

1434 The following exemplar defines the `CreateSequence` syntax when extended to include a
1435 `wsse:SecurityTokenReference`:

```
1436 <wsrm:CreateSequence ...>  
1437   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1438   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1439   <wsrm:Offer ...>  
1440     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1441     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1442     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1443     <wsrm:IncompleteSequenceBehavior>  
1444       wsrml:IncompleteSequenceBehaviorType  
1445     </wsrm:IncompleteSequenceBehavior> ?  
1446     ...  
1447   </wsrm:Offer> ?  
1448   ...  
1449   <wsse:SecurityTokenReference>  
1450     ...  
1451   </wsse:SecurityTokenReference> ?  
1452   ...  
1453 </wsrm:CreateSequence>
```

1454 The following describes the content model of the additional `CreateSequence` elements.

1455 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1456 This element uses the extensibility mechanism defined for the `CreateSequence` element
1457 (defined in section 3.4) to communicate an explicit reference to the security token, using a
1458 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and
1459 Destination MUST use to authorize messages for the created (and, if present, the offered)
1460 Sequence(s). All subsequent messages related to the created (and, if present, the offered)
1461 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token
1462 (e.g., by using or deriving from a private or secret key).

1463 When a RM Source transmits a `CreateSequence` that has been extended to include a
1464 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1465 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1466 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1467 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1468 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1469 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1470 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1471 in WS-Security still applies.

1472 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1473 <wsmr:UsesSequenceSTR ... />
```

1474 The following describes the content model of the `UsesSequenceSTR` header block.

1475 `/wsmr:UsesSequenceSTR`

1476 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that
1477 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`
1478 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly
1479 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus
1480 aborting the requested Sequence creation.

1481 The following is an example of a `CreateSequence` message using the

1482 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1483 <soap:Envelope ...>  
1484   <soap:Header>  
1485     ...  
1486     <wsmr:UsesSequenceSTR soap:mustUnderstand='true' />  
1487     ...  
1488   </soap:Header>  
1489   <soap:Body>  
1490     <wsmr:CreateSequence>  
1491       <wsmr:AcksTo>  
1492         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1493       </wsmr:AcksTo>  
1494       <wsse:SecurityTokenReference>  
1495         ...  
1496       </wsse:SecurityTokenReference>  
1497     </wsmr:CreateSequence>  
1498   </soap:Body>  
1499 </soap:Envelope>
```

1500 6.2 Securing Sequences Using SSL/TLS

1501 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1502 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1503 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1504 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1505 SOAP header block within the `CreateSequence` message.

1506 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1507 <wsmr:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1508 The following describes the content model of the `UsesSequenceSSL` header block.

1509 `/wsmr:UsesSequenceSSL`

1510 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`
1511 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1512 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM
1513 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.
1514 The receiving RM Destination MUST understand and correctly implement the functionality
1515 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the
1516 requested Sequence creation.

1517 Note that the inclusion of the above header by the RM Source implies that all Sequence-related
1518 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1519 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1520 `CreateSequenceResponse` message.

1521 Appendix A. Schema

1522 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1523 Schema Part2] is located at:

1524 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1525 The following copy is provided for reference.

```
1526 <?xml version="1.0" encoding="UTF-8"?>
1527 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1528      OASIS trademark, IPR and other policies apply. -->
1529 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1530 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1531 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1532 rx/wsrn/200702" elementFormDefault="qualified"
1533 attributeFormDefault="unqualified">
1534   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1535 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1536   <!-- Protocol Elements -->
1537   <xs:complexType name="SequenceType">
1538     <xs:sequence>
1539       <xs:element ref="wsrm:Identifier"/>
1540       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1541       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1542 maxOccurs="unbounded"/>
1543     </xs:sequence>
1544     <xs:anyAttribute namespace="##other" processContents="lax"/>
1545   </xs:complexType>
1546   <xs:element name="Sequence" type="wsrm:SequenceType"/>
1547   <xs:element name="SequenceAcknowledgement">
1548     <xs:complexType>
1549       <xs:sequence>
1550         <xs:element ref="wsrm:Identifier"/>
1551         <xs:choice>
1552           <xs:sequence>
1553             <xs:choice>
1554               <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1555                 <xs:complexType>
1556                   <xs:sequence/>
1557                   <xs:attribute name="Upper" type="xs:unsignedLong"
1558 use="required"/>
1559                   <xs:attribute name="Lower" type="xs:unsignedLong"
1560 use="required"/>
1561                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1562               </xs:complexType>
1563             </xs:choice>
1564             <xs:element name="None">
1565               <xs:complexType>
1566                 <xs:sequence/>
1567               </xs:complexType>
1568             </xs:element>
1569           </xs:choice>
1570           <xs:element name="Final" minOccurs="0">
1571             <xs:complexType>
1572               <xs:sequence/>
1573             </xs:complexType>
1574           </xs:element>
1575         </xs:sequence>
1576       <xs:element name="Nack" type="xs:unsignedLong"

```

```

1577 maxOccurs="unbounded"/>
1578     </xs:choice>
1579     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1580 maxOccurs="unbounded"/>
1581     </xs:sequence>
1582     <xs:anyAttribute namespace="##other" processContents="lax"/>
1583 </xs:complexType>
1584 </xs:element>
1585 <xs:complexType name="AckRequestedType">
1586     <xs:sequence>
1587         <xs:element ref="wsrm:Identifier"/>
1588         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1589 maxOccurs="unbounded"/>
1590     </xs:sequence>
1591     <xs:anyAttribute namespace="##other" processContents="lax"/>
1592 </xs:complexType>
1593 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1594 <xs:element name="Identifier">
1595     <xs:complexType>
1596         <xs:annotation>
1597             <xs:documentation>
1598                 This type is for elements whose [children] is an anyURI and can have
1599 arbitrary attributes.
1600             </xs:documentation>
1601         </xs:annotation>
1602         <xs:simpleContent>
1603             <xs:extension base="xs:anyURI">
1604                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1605             </xs:extension>
1606         </xs:simpleContent>
1607     </xs:complexType>
1608 </xs:element>
1609 <xs:element name="Address">
1610     <xs:complexType>
1611         <xs:simpleContent>
1612             <xs:extension base="xs:anyURI">
1613                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1614             </xs:extension>
1615         </xs:simpleContent>
1616     </xs:complexType>
1617 </xs:element>
1618 <xs:simpleType name="MessageNumberType">
1619     <xs:restriction base="xs:unsignedLong">
1620         <xs:minInclusive value="1"/>
1621         <xs:maxInclusive value="9223372036854775807"/>
1622     </xs:restriction>
1623 </xs:simpleType>
1624 <!-- Fault Container and Codes -->
1625 <xs:simpleType name="FaultCodes">
1626     <xs:restriction base="xs:QName">
1627         <xs:enumeration value="wsrm:SequenceTerminated"/>
1628         <xs:enumeration value="wsrm:UnknownSequence"/>
1629         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1630         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1631         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1632         <xs:enumeration value="wsrm:SequenceClosed"/>
1633         <xs:enumeration value="wsrm:WSRMRequired"/>
1634     </xs:restriction>
1635 </xs:simpleType>
1636 <xs:complexType name="SequenceFaultType">
1637     <xs:sequence>
1638         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1639         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1640     <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1641 maxOccurs="unbounded"/>
1642 </xs:sequence>
1643 <xs:anyAttribute namespace="##other" processContents="lax"/>
1644 </xs:complexType>
1645 <xs:complexType name="DetailType">
1646 <xs:sequence>
1647 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1648 maxOccurs="unbounded"/>
1649 </xs:sequence>
1650 <xs:anyAttribute namespace="##other" processContents="lax"/>
1651 </xs:complexType>
1652 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1653 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1654 <xs:element name="CreateSequenceResponse"
1655 type="wsrm:CreateSequenceResponseType"/>
1656 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1657 <xs:element name="CloseSequenceResponse"
1658 type="wsrm:CloseSequenceResponseType"/>
1659 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1660 <xs:element name="TerminateSequenceResponse"
1661 type="wsrm:TerminateSequenceResponseType"/>
1662 <xs:complexType name="CreateSequenceType">
1663 <xs:sequence>
1664 <xs:element ref="wsrm:AcksTo"/>
1665 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1666 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1667 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1668 maxOccurs="unbounded">
1669 <xs:annotation>
1670 <xs:documentation>
1671 It is the authors intent that this extensibility be used to
1672 transfer a Security Token Reference as defined in WS-Security.
1673 </xs:documentation>
1674 </xs:annotation>
1675 </xs:any>
1676 </xs:sequence>
1677 <xs:anyAttribute namespace="##other" processContents="lax"/>
1678 </xs:complexType>
1679 <xs:complexType name="CreateSequenceResponseType">
1680 <xs:sequence>
1681 <xs:element ref="wsrm:Identifier"/>
1682 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1683 <xs:element name="IncompleteSequenceBehavior"
1684 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1685 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1686 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1687 maxOccurs="unbounded"/>
1688 </xs:sequence>
1689 <xs:anyAttribute namespace="##other" processContents="lax"/>
1690 </xs:complexType>
1691 <xs:complexType name="CloseSequenceType">
1692 <xs:sequence>
1693 <xs:element ref="wsrm:Identifier"/>
1694 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1695 minOccurs="0"/>
1696 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1697 maxOccurs="unbounded"/>
1698 </xs:sequence>
1699 <xs:anyAttribute namespace="##other" processContents="lax"/>
1700 </xs:complexType>
1701 <xs:complexType name="CloseSequenceResponseType">
1702 <xs:sequence>
1703 <xs:element ref="wsrm:Identifier"/>
1704 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1705 maxOccurs="unbounded"/>
1706 </xs:sequence>
1707 <xs:anyAttribute namespace="##other" processContents="lax"/>
1708 </xs:complexType>
1709 <xs:complexType name="TerminateSequenceType">
1710 <xs:sequence>
1711 <xs:element ref="wsrm:Identifier"/>
1712 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1713 minOccurs="0"/>
1714 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1715 maxOccurs="unbounded"/>
1716 </xs:sequence>
1717 <xs:anyAttribute namespace="##other" processContents="lax"/>
1718 </xs:complexType>
1719 <xs:complexType name="TerminateSequenceResponseType">
1720 <xs:sequence>
1721 <xs:element ref="wsrm:Identifier"/>
1722 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1723 maxOccurs="unbounded"/>
1724 </xs:sequence>
1725 <xs:anyAttribute namespace="##other" processContents="lax"/>
1726 </xs:complexType>
1727 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1728 <xs:complexType name="OfferType">
1729 <xs:sequence>
1730 <xs:element ref="wsrm:Identifier"/>
1731 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1732 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1733 <xs:element name="IncompleteSequenceBehavior"
1734 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1735 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1736 maxOccurs="unbounded"/>
1737 </xs:sequence>
1738 <xs:anyAttribute namespace="##other" processContents="lax"/>
1739 </xs:complexType>
1740 <xs:complexType name="AcceptType">
1741 <xs:sequence>
1742 <xs:element ref="wsrm:AcksTo"/>
1743 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1744 maxOccurs="unbounded"/>
1745 </xs:sequence>
1746 <xs:anyAttribute namespace="##other" processContents="lax"/>
1747 </xs:complexType>
1748 <xs:element name="Expires">
1749 <xs:complexType>
1750 <xs:simpleContent>
1751 <xs:extension base="xs:duration">
1752 <xs:anyAttribute namespace="##other" processContents="lax"/>
1753 </xs:extension>
1754 </xs:simpleContent>
1755 </xs:complexType>
1756 </xs:element>
1757 <xs:simpleType name="IncompleteSequenceBehaviorType">
1758 <xs:restriction base="xs:string">
1759 <xs:enumeration value="DiscardEntireSequence"/>
1760 <xs:enumeration value="DiscardFollowingFirstGap"/>
1761 <xs:enumeration value="NoDiscard"/>
1762 </xs:restriction>
1763 </xs:simpleType>
1764 <xs:element name="UsesSequenceSTR">
1765 <xs:complexType>
1766 <xs:sequence/>
1767 <xs:anyAttribute namespace="##other" processContents="lax"/>
1768 </xs:complexType>

```

```
1769 </xs:element>
1770 <xs:element name="UsesSequenceSSL">
1771   <xs:complexType>
1772     <xs:sequence/>
1773     <xs:anyAttribute namespace="##other" processContents="lax"/>
1774   </xs:complexType>
1775 </xs:element>
1776 <xs:element name="UnsupportedElement">
1777   <xs:simpleType>
1778     <xs:restriction base="xs:QName"/>
1779   </xs:simpleType>
1780 </xs:element>
1781 </xs:schema>
```

1782 Appendix B. WSDL

1783 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1784 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1785 present in exchanges with that endpoint.

1786 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1787 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1788 for a higher-level mechanism to indicate that WS-RM is engaged.

1789 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1790 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

1791 The following non-normative copy is provided for reference.

```
1792 <?xml version="1.0" encoding="utf-8"?>
1793 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1794 OASIS trademark, IPR and other policies apply. -->
1795 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1796 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1797 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1798 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1799 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1800 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL"
1801 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL">
1802
1803   <wSDL:types>
1804     <xs:schema
1805       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1806       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1807       200702.xsd"/>
1808     </xs:schema>
1809   </wSDL:types>
1810
1811   <wSDL:message name="CreateSequence">
1812     <wSDL:part name="create" element="rm:CreateSequence"/>
1813   </wSDL:message>
1814   <wSDL:message name="CreateSequenceResponse">
1815     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1816   </wSDL:message>
1817   <wSDL:message name="CloseSequence">
1818     <wSDL:part name="close" element="rm:CloseSequence"/>
1819   </wSDL:message>
1820   <wSDL:message name="CloseSequenceResponse">
1821     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1822   </wSDL:message>
1823   <wSDL:message name="TerminateSequence">
1824     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1825   </wSDL:message>
1826   <wSDL:message name="TerminateSequenceResponse">
1827     <wSDL:part name="terminateResponse"
1828     element="rm:TerminateSequenceResponse"/>
1829   </wSDL:message>
1830
1831   <wSDL:portType name="SequenceAbstractPortType">
1832     <wSDL:operation name="CreateSequence">
1833       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1834       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1835       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1836 wsam:Action="http://docs.oasis-open.org/ws-
1837 rx/wsrn/200702/CreateSequenceResponse"/>
1838 </wsdl:operation>
1839 <wsdl:operation name="CloseSequence">
1840 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1841 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1842 <wsdl:output message="tns:CloseSequenceResponse"
1843 wsam:Action="http://docs.oasis-open.org/ws-
1844 rx/wsrn/200702/CloseSequenceResponse"/>
1845 </wsdl:operation>
1846 <wsdl:operation name="TerminateSequence">
1847 <wsdl:input message="tns:TerminateSequence"
1848 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1849 <wsdl:output message="tns:TerminateSequenceResponse"
1850 wsam:Action="http://docs.oasis-open.org/ws-
1851 rx/wsrn/200702/TerminateSequenceResponse"/>
1852 </wsdl:operation>
1853 </wsdl:portType>
1854
1855 </wsdl:definitions>
```

1856 Appendix C. Message Examples

1857 Appendix C.1 Create Sequence

1858 Create Sequence

```
1859 <?xml version="1.0" encoding="UTF-8"?>
1860 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1861 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1862 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1863   <S:Header>
1864     <wsa:MessageID>
1865       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1866     </wsa:MessageID>
1867     <wsa:To>http://example.com/serviceB/123</wsa:To>
1868     <wsa:Action>http://docs.oasis-open.org/ws-
1869 rx/wsmr/200702/CreateSequence</wsa:Action>
1870     <wsa:ReplyTo>
1871       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1872     </wsa:ReplyTo>
1873   </S:Header>
1874   <S:Body>
1875     <wsmr:CreateSequence>
1876       <wsmr:AcksTo>
1877         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1878       </wsmr:AcksTo>
1879     </wsmr:CreateSequence>
1880   </S:Body>
1881 </S:Envelope>
```

1882 Create Sequence Response

```
1883 <?xml version="1.0" encoding="UTF-8"?>
1884 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1885 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1886 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1887   <S:Header>
1888     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1889     <wsa:RelatesTo>
1890       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1891     </wsa:RelatesTo>
1892     <wsa:Action>
1893       http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1894     </wsa:Action>
1895   </S:Header>
1896   <S:Body>
1897     <wsmr:CreateSequenceResponse>
1898       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1899     </wsmr:CreateSequenceResponse>
1900   </S:Body>
1901 </S:Envelope>
```

1902 Appendix C.2 Initial Transmission

1903 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1904 figure. The three messages have the following headers; the third message is identified as the last
1905 message in the Sequence:

1906 Message 1

```
1907 <?xml version="1.0" encoding="UTF-8"?>
1908 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1909 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1910 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1911   <S:Header>
1912     <wsa:MessageID>
1913       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1914     </wsa:MessageID>
1915     <wsa:To>http://example.com/serviceB/123</wsa:To>
1916     <wsa:From>
1917       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1918     </wsa:From>
1919     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1920     <wsmr:Sequence>
1921       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1922       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1923     </wsmr:Sequence>
1924   </S:Header>
1925   <S:Body>
1926     <!-- Some Application Data -->
1927   </S:Body>
1928 </S:Envelope>
```

1929 Message 2

```
1930 <?xml version="1.0" encoding="UTF-8"?>
1931 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1932 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1933 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1934   <S:Header>
1935     <wsa:MessageID>
1936       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1937     </wsa:MessageID>
1938     <wsa:To>http://example.com/serviceB/123</wsa:To>
1939     <wsa:From>
1940       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1941     </wsa:From>
1942     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1943     <wsmr:Sequence>
1944       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1945       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1946     </wsmr:Sequence>
1947   </S:Header>
1948   <S:Body>
1949     <!-- Some Application Data -->
1950   </S:Body>
1951 </S:Envelope>
```

1952 Message 3

```
1953 <?xml version="1.0" encoding="UTF-8"?>
1954 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1955 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1956 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1957   <S:Header>
1958     <wsa:MessageID>
1959       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1960     </wsa:MessageID>
1961     <wsa:To>http://example.com/serviceB/123</wsa:To>
1962     <wsa:From>
1963       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1964     </wsa:From>
1965     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
```

```

1966 <wsrm:Sequence>
1967 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1968 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1969 </wsrm:Sequence>
1970 <wsrm:AckRequested>
1971 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1972 </wsrm:AckRequested>
1973 </S:Header>
1974 <S:Body>
1975 <!-- Some Application Data -->
1976 </S:Body>
1977 </S:Envelope>

```

1978 **Appendix C.3 First Acknowledgement**

1979 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1980 responds with an Acknowledgement for messages 1 and 3:

```

1981 <?xml version="1.0" encoding="UTF-8"?>
1982 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1983 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1984 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1985 <S:Header>
1986 <wsa:MessageID>
1987 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1988 </wsa:MessageID>
1989 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1990 <wsa:From>
1991 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1992 </wsa:From>
1993 <wsa:Action>
1994 http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1995 </wsa:Action>
1996 <wsrm:SequenceAcknowledgement>
1997 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1998 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1999 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2000 </wsrm:SequenceAcknowledgement>
2001 </S:Header>
2002 <S:Body/>
2003 </S:Envelope>

```

2004 **Appendix C.4 Retransmission**

2005 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2006 requests an Acknowledgement:

```

2007 <?xml version="1.0" encoding="UTF-8"?>
2008 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2009 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2010 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2011 <S:Header>
2012 <wsa:MessageID>
2013 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2014 </wsa:MessageID>
2015 <wsa:To>http://example.com/serviceB/123</wsa:To>
2016 <wsa:From>
2017 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2018 </wsa:From>
2019 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2020 <wsrm:Sequence>

```

```

2021     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2022     <wsrm:MessageNumber>2</wsrm:MessageNumber>
2023     </wsrm:Sequence>
2024     <wsrm:AckRequested>
2025     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2026     </wsrm:AckRequested>
2027     </S:Header>
2028     <S:Body>
2029     <!-- Some Application Data -->
2030     </S:Body>
2031     </S:Envelope>

```

2032 Appendix C.5 Termination

2033 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2034 be terminated:

```

2035 <?xml version="1.0" encoding="UTF-8"?>
2036 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2037 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2038 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2039   <S:Header>
2040     <wsa:MessageID>
2041       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2042     </wsa:MessageID>
2043     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2044     <wsa:From>
2045       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2046     </wsa:From>
2047     <wsa:Action>
2048       http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2049     </wsa:Action>
2050     <wsrm:SequenceAcknowledgement>
2051       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2052       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2053     </wsrm:SequenceAcknowledgement>
2054   </S:Header>
2055   <S:Body/>
2056 </S:Envelope>

```

2057 Terminate Sequence

```

2058 <?xml version="1.0" encoding="UTF-8"?>
2059 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2060 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2061 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2062   <S:Header>
2063     <wsa:MessageID>
2064       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2065     </wsa:MessageID>
2066     <wsa:To>http://example.com/serviceB/123</wsa:To>
2067     <wsa:Action>
2068       http://docs.oasis-open.org/ws-rx/wsr/200702/TerminateSequence
2069     </wsa:Action>
2070     <wsa:From>
2071       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2072     </wsa:From>
2073   </S:Header>
2074   <S:Body>
2075     <wsrm:TerminateSequence>
2076       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2077       <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2078     </wsrm:TerminateSequence>

```

```
2079 </S:Body>
2080 </S:Envelope>
```

2081 Terminate Sequence Response

```
2082 <?xml version="1.0" encoding="UTF-8"?>
2083 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2084 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2085 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2086 <S:Header>
2087 <wsa:MessageID>
2088 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2089 </wsa:MessageID>
2090 <wsa:To>http://example.com/serviceA/789</wsa:To>
2091 <wsa:Action>
2092 http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
2093 </wsa:Action>
2094 <wsa:RelatesTo>
2095 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2096 </wsa:RelatesTo>
2097 <wsa:From>
2098 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2099 </wsa:From>
2100 </S:Header>
2101 <S:Body>
2102 <wsm:TerminateSequenceResponse>
2103 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2104 </wsm:TerminateSequenceResponse>
2105 </S:Body>
2106 </S:Envelope>
```

2107 Appendix D. State Tables

2108 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2109 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

2110 Legend:

2111 The first column of these tables contains the motivating event and has the following format:

Event 2112
<i>Event name</i> [source]
{ref}

2113 Where:

- 2114 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2115 described by the specification.
- 2116 • [source]: indicates the source of the event; one of:
 - 2117 ○ [msg] a Received message
 - 2118 ○ [int]: an internal event such as the firing of a timer
 - 2119 ○ [app]: the application
 - 2120 ○ [unspec]: the source is unspecified

2121 Each event / state combination cell in the tables in this appendix has the following format:

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

2122 Where:

- 2123 • action to take: indicates that the state machine performs the following action. Actions surrounded
2124 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2125 "Transmit"
- 2126 • [next state]: indicates the state to which the state machine will advance upon the performance of
2127 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2128 • {ref} is a reference to the document section describing the behavior in this cell

2129 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2130 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2131 described in this specification and does not indicate normal protocol operation. Implementations MAY
2132 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2133 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2134 combinations.

2135 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
Create Sequence [unspec] {3.4}	Xmit Create Sequence [Creating] {3.4}	N/A	N/A	N/A	N/A	N/A
Create Sequence Response [msg] {3.4}		Process Create Sequence Response [Created] {3.4}				
Create Sequence Refused Fault [msg] {3.4}		No action [None] {4.6}				
Send message [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [Same] {2}	N/A	N/A
Retransmit of un-ack'd message [int]	N/A	N/A	Xmit message [Same] {2.3}	Xmit message [Same] {2.3}	N/A	N/A
SeqAck (non-final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}
Nack [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	<Xmit message(s)> [Same] {3.9}	<Xmit message(s)> [Same] {3.9}	No action [Same]	No action [Same]
Message Number Rollover Fault [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Same]	No action [Same]	No action [Same]	No action [Same]
CloseSequence [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}
<Close Sequence> [int] {3.5}	N/A		Xmit Close Sequence [Closing] {3.5}	N/A	N/A	N/A
Close Sequence Response [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}		No action [Closed] {3.5}	No action [Same] {3.5}	No action [Same] {3.5}

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
SeqAck (final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Same]	Process Ack ranges [Same]
Sequence Closed Fault [msg] {4.7}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]
Unknown Sequence Fault [msg] {4.3}			Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
Sequence Terminated Fault [msg] {4.2}	N/A		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
TerminateSequence [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}
Terminate Sequence [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	N/A
Terminate Sequence Response [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}				Terminate Sequence [None] {3.6}
Expires exceeded [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
Invalid Acknowledgement [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

2136 Table 2 RM Destination Sequence State Transition Table

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

Events	Sequence States			
	None	Created	Closed	Terminating
CreateSequence (unsuccessful) [msg/int] {3.4}	Generate Create Sequence Refused Fault [None] {3.4}	N/A	N/A	
Message (with message number within range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Accept Message; <Xmit SeqAck> [Same]	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
Message (with message number outside of range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.7}{4.5}	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
<AckRequested> [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}
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
<CloseSequence autonomously> [int]		Xmit CloseSequence with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence with SeqAck+Final [Same] {3.5}	
CloseSequenceResponse [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}		No Action [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
TerminateSequence [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}
<TerminateSequence autonomously> [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}
TerminateSequenceResponse [msg]	Generate Unknown Sequence Fault [Same] {4.3}			Terminate Sequence [None]
UnknownSequence Fault [msg] {4.3}		Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
SequenceTerminated Fault [msg] {4.2}		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.3}
Invalid Acknowledgement Fault [msg] {4.4}	N/A			
Expires exceeded [int]	N/A	Terminate Sequence [None]	Terminate Sequence [None]	

Events	Sequence States			
	None	Created	Closed	Terminating
		{3.4}	{3.4}	
<Seq Acknowledgement autonomously> [int] {3.9}	N/A	Xmit SeqAck [Same] {3.9}	Xmit SeqAck+Final [Same] {3.9}	
Non WSRM message when WSRM required [msg] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	Generate WSRMRequired Fault [Same] {4.8}	

2137 Appendix E. Acknowledgments

2138 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2139 authors:

2140	Ruslan Bilorusets, BEA	2152	Amelia Lewis, TIBCO Software
2141	Don Box, Microsoft	2153	Rodney Limprecht, Microsoft
2142	Luis Felipe Cabrera, Microsoft	2154	Steve Lucco, Microsoft
2143	Doug Davis, IBM	2155	Don Mullen, TIBCO Software
2144	Donald Ferguson, IBM	2156	Anthony Nadalin, IBM
2145	Christopher Ferris, IBM	2157	Mark Nottingham, BEA
2146	Tom Freund, IBM	2158	David Orchard, BEA
2147	Mary Ann Hondo, IBM	2159	Jamie Roots, IBM
2148	John Ibbotson, IBM	2160	Shivajee Samdarshi, TIBCO Software
2149	Lei Jin, BEA	2161	John Shewchuk, Microsoft
2150	Chris Kaler, Microsoft	2162	Tony Storey, IBM
2151	David Langworthy-Editor, Microsoft		

2163 The following individuals have provided invaluable input into the initial contribution:

2164	Keith Ballinger, Microsoft	2178	David Ingham, Microsoft
2165	Stefan Batres, Microsoft	2179	Gopal Kakivaya, Microsoft
2166	Rebecca Bergersen, Iona	2180	Johannes Klein, Microsoft
2167	Allen Brown, Microsoft	2181	Frank Leymann, IBM
2168	Michael Conner, IBM	2182	Martin Nally, IBM
2169	George Copeland, Microsoft	2183	Peter Niblett, IBM
2170	Francisco Curbera, IBM	2184	Jeffrey Schlimmer, Microsoft
2171	Paul Fremantle, IBM	2185	James Snell, IBM
2172	Steve Graham, IBM	2186	Keith Stobie, Microsoft
2173	Pat Helland, Microsoft	2187	Satish Thatte, Microsoft
2174	Rick Hill, Microsoft	2188	Stephen Todd, IBM
2175	Scott Hinkelman, IBM	2189	Sanjiva Weerawarana, IBM
2176	Tim Holloway, IBM	2190	Roger Wolter, Microsoft
2177	Efim Hudis, Microsoft		

2191 The following individuals were members of the committee during the development of this specification:

2192	Abbie Barbir, Nortel	2211	Robert Freund, Hitachi
2193	Charlton Barreto, Adobe	2212	Peter Furniss, Erebore
2194	Stefan Batres, Microsoft	2213	Marc Goodner, Microsoft
2195	Hamid Ben Malek, Fujitsu	2214	Alastair Green, Choreology
2196	Andreas Bjarlestam, Ericsson	2215	Mike Grogan, Sun
2197	Toufic Boubez, Layer 7	2216	Ondrej Hrebicek, Microsoft
2198	Doug Bunting, Sun	2217	Kazunori Iwasa, Fujitsu
2199	Lloyd Burch, Novell	2218	Chamikara Jayalath, WSO2
2200	Steve Carter, Novell	2219	Lei Jin, BEA
2201	Martin Chapman, Oracle	2220	Ian Jones, BTplc
2202	Dave Chappell, Sonic	2221	Anish Karmarkar, Oracle
2203	Paul Cotton, Microsoft	2222	Paul Knight, Nortel
2204	Glen Daniels, Sonic	2223	Dan Leshchiner, Tibco
2205	Doug Davis, IBM	2224	Mark Little, JBoss
2206	Blake Dournaee, Intel	2225	Lily Liu, webMethods
2207	Jacques Durand, Fujitsu	2226	Matt Lovett, IBM
2208	Colleen Evans, Microsoft	2227	Ashok Malhotra, Oracle
2209	Christopher Ferris, IBM	2228	Jonathan Marsh, Microsoft
2210	Paul Fremantle, WSO2	2229	Daniel Millwood, IBM

2230	Jeff Mischkinsky, Oracle	2241	Stefan Rossmanith, SAP
2231	Nilo Mitra, Ericsson	2242	Tom Rutt, Fujitsu
2232	Peter Niblett, IBM	2243	Rich Salz, IBM
2233	Duane Nickull, Adobe	2244	Shivajee Samdarshi, Tibco
2234	Eisaku Nishiyama, Hitachi	2245	Vladimir Videlov, SAP
2235	Dave Orchard, BEA	2246	Claus von Riegen, SAP
2236	Chouthri Palanisamy, NEC	2247	Pete Wenzel, Sun
2237	Sanjay Patil, SAP	2248	Steve Winkler, SAP
2238	Gilbert Pilz, BEA	2249	Ümit Yalçinalp, SAP
2239	Martin Raepfle, SAP	2250	Nobuyuki Yamamoto, Hitachi
2240	Eric Rajkovic, Oracle		
2251			