



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

3 Committee Specification 01

4 10 September 2008

5 Specification URIs:

6 This Version:

7 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.pdf>

8 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.html>

9 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cs-01.doc>

10 Previous Version:

11 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-01.pdf>

12 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-01.html>

13 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-cd-01.doc>

14 Latest Version:

15 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.pdf>

16 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.html>

17 <http://docs.oasis-open.org/ws-rx/wsrn/v1.2/wsrn.doc>

18 Technical Committee:

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

20 Chairs:

21 Paul Fremantle <paul@wso2.com>

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 WS-RX Technical Committee Specification 01, "Web Services Reliable
202 Messaging Policy Assertion(WS-RM Policy)," September 2008
203 <http://docs.oasis-open.org/ws-rx/wsrmp/v1.2/wsrmp-1.2-spec-cs-01.html>
- 204 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
205 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 206 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June
207 2003.
208 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 209 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
210 Generic Syntax," RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January
211 2005.
212 <http://ietf.org/rfc/rfc3986>
- 213 **[UUID]** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN
214 Namespace," RFC 4122, Microsoft, Refactored Networks - LLC, DataPower
215 Technology Inc, July 2005
216 <http://www.ietf.org/rfc/rfc4122.txt>
- 217 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth
218 Edition)", September 2006.
219 <http://www.w3.org/TR/REC-xml/>
- 220 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.
221 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 222 **[XML-Schema Part1]** W3C Recommendation, "XML Schema Part 1: Structures," October 2004.
223 <http://www.w3.org/TR/xmlschema-1/>
- 224 **[XML-Schema Part2]** W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
225 <http://www.w3.org/TR/xmlschema-2/>
- 226 **[XPath 1.0]** W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16
227 November 1999.
228 <http://www.w3.org/TR/xpath>
- 229 **[WSDL 1.1]** W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
230 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 231 **[WS-Addressing]** W3C Recommendation, "Web Services Addressing 1.0 – Core," May 2006.
232 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
233 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding," May
234 2006
235 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

236 1.3 Non-Normative References

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

- 245 1999.
 246 <http://www.ietf.org/rfc/rfc2617.txt>
- 247 **[RFC 4346]** T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version
 248 1.1," April 2006.
 249 <http://www.ietf.org/rfc/rfc4346.txt>
- 250 **[WS-Policy]** W3C Recommendation, "Web Services Policy 1.5 - Framework," September
 251 2007.
 252 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>
- 253 **[WS-PolicyAttachment]** W3C Recommendation, "Web Services Policy 1.5 - Attachment,"
 254 September 2007.
 255 <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>
- 256 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 257 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",
 258 OASIS Standard 200401, March 2004.
 259 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](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
 262 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 263 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)",
 264 OASIS Standard 200602, February 2006.
 265 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 266 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",
 267 RFC 1323, May 1992.
 268 <http://www.rfc-editor.org/rfc/rfc1323.txt>
- 269 **[SecurityPolicy]** OASIS WS-SX Technical Committee Editor Draft, "WS-SecurityPolicy 1.3"
 270 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200802>
- 271 **[SecureConversation]** S. Anderson, et al, "Web Services Secure Conversation Language (WS-
 272 SecureConversation)," February 2005.
 273 <http://schemas.xmlsoap.org/ws/2004/04/sc/>
- 274 **[Trust]** OASIS WS-SX Technical Committee Editor Draft, "WS-Trust 1.4"
 275 <http://docs.oasis-open.org/ws-sx/ws-trust/200802>

276 1.4 Namespace

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

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

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

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

283 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
wsrml	http://docs.oasis-open.org/ws-rx/wsrml/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

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

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

287 **1.5 Conformance**

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

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

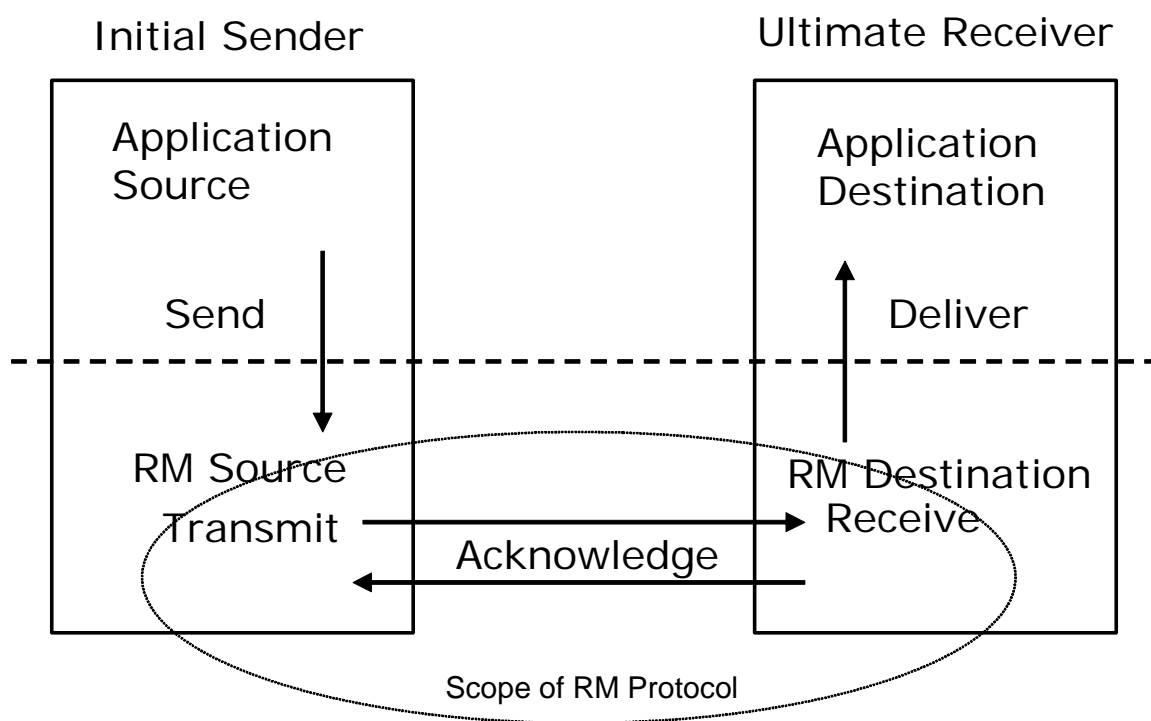
294 **2 Reliable Messaging Model**

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

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

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

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



320 Figure 1: Reliable Messaging Model

321 2.1 Glossary

322 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

352 2.2 Protocol Preconditions

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

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

362 2.3 Protocol Invariants

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

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

376 2.4 Delivery Assurances

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

380 `AtLeastOnce`

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

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

388 AtMostOnce

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

393 ExactlyOnce

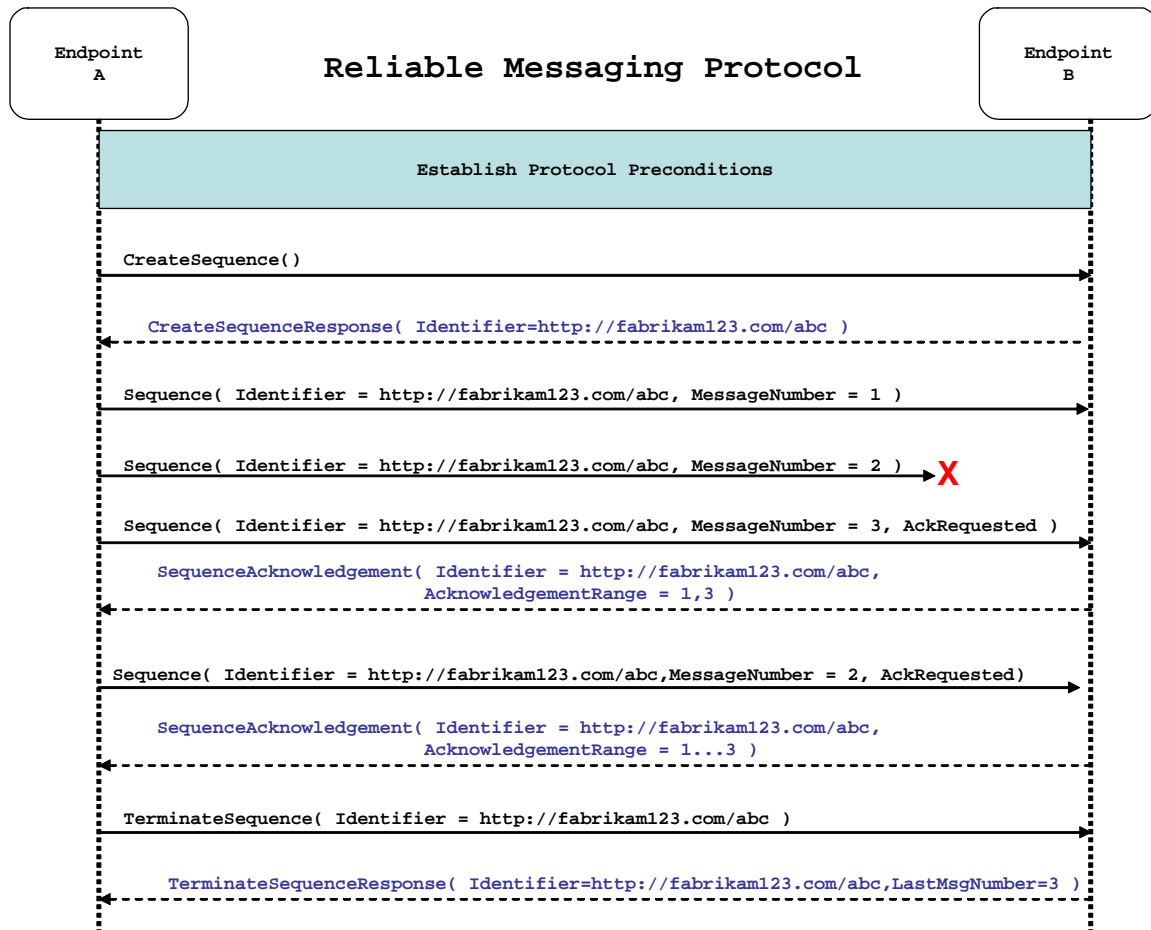
394 Each message is to be delivered exactly once; if a message cannot be delivered then an error
395 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is
396 that it SHOULD retry transmission of every message sent by the Application Source until it
397 receives an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

401 InOrder

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

413 2.5 Example Message Exchange

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



415 Figure 2: The WS-ReliableMessaging Protocol

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

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

456 3 RM Protocol Elements

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

459 3.1 Considerations on the Use of Extensibility Points

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

464 3.2 Considerations on the Use of "Piggy-Backing"

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

474 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

492 3.4 Sequence Creation

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

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

500 The following exemplar defines the `CreateSequence` syntax:

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

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

516 `/wsrm:CreateSequence`

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

521 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

531 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

591 /wsmr:CreateSequence/{any}

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

594 /wsmr:CreateSequence/@{any}

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

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

601 The following exemplar defines the `CreateSequenceResponse` syntax:

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

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

615 /wsmr:CreateSequenceResponse

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

619 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

627 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

640 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

653 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

659 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

674 /wsrm:CreateSequenceResponse/{ any }

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

677 /wsrm:CreateSequenceResponse/@{ any }

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

680 3.5 Closing A Sequence

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

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

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

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

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

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

714 The following exemplar defines the `CloseSequence` syntax:

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

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

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

721 `/wsmr:CloseSequence`

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

725 `/wsmr:CloseSequence/wsmr:Identifier`

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

729 `/wsmr:CloseSequence/wsmr:LastMessageNumber`

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

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

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

736 `/wsmr:CloseSequence/{any}`

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

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

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

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

744 The following exemplar defines the `CloseSequenceResponse` syntax:

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

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

750 `/wsmr:CloseSequenceResponse`

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

753 `/wsmr:CloseSequenceResponse/wsmr:Identifier`

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

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

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

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

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

766 3.6 Sequence Termination

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

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

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

785 The following exemplar defines the `TerminateSequence` syntax:

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

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

792 /wsrm:TerminateSequence

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

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

804 /wsrm:TerminateSequence/wsrm:Identifier

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

808 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

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

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

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

815 `/wsrm:TerminateSequence/{any}`

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

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

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

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

823 The following exemplar defines the `TerminateSequenceResponse` syntax:

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

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

829 `/wsrm:TerminateSequenceResponse`

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

833 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

840 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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

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

849 3.7 Sequences

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

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

857 A following exemplar defines its syntax:

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

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

864 `/wsrm:Sequence`

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

871 `/wsrm:Sequence/wsrm:Identifier`

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

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

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

878 `/wsrm:Sequence/wsrm:MessageNumber`

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

883 `/wsrm:Sequence/{any}`

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

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

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

889 The following example illustrates a Sequence header block.

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

894 3.8 Request Acknowledgement

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

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

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

909 The following exemplar defines its syntax:

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

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

915 /wsrm:AckRequested

916 This element requests an Acknowledgement for the identified Sequence.

917 /wsrm:AckRequested/wsrm:Identifier

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

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

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

925 /wsrm:AckRequested/{any}

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

928 /wsrm:AckRequested/@{any}

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

931 3.9 Sequence Acknowledgement

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

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

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

952 The following exemplar defines its syntax:

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

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

965 `/wsrm:SequenceAcknowledgement`

966 This element contains the Sequence Acknowledgement information.

967 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

976 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

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

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

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

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

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

1004 `/wsrm:SequenceAcknowledgement/wsrm:Nack`
1005 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1006 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`
1007 representing the `MessageNumber` of an unreceived message in a `Sequence`. The RM Destination
1008 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is
1009 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM
1010 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT
1011 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously
1012 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a
1013 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been
1014 acknowledged within an `AcknowledgementRange`.

1015 `/wsrm:SequenceAcknowledgement/{any}`
1016 This is an extensibility mechanism to allow different (extensible) types of information, based on a
1017 schema, to be passed.

1018 `/wsrm:SequenceAcknowledgement/@{any}`
1019 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1020 to the element.

1021 The following examples illustrate `SequenceAcknowledgement` elements:

- 1022 • Message numbers 1..10 inclusive in a Sequence have been accepted by the RM Destination.

```
1023 <wsrm:SequenceAcknowledgement>  
1024   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1025   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
1026 </wsrm:SequenceAcknowledgement>
```

- 1027 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM
1028 Destination, messages 3 and 7 have not been accepted.

```
1029 <wsrm:SequenceAcknowledgement>  
1030   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1031   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1032   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1033   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1034 </wsrm:SequenceAcknowledgement>
```

- 1035 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1036 <wsrm:SequenceAcknowledgement>  
1037   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1038   <wsrm:Nack>3</wsrm:Nack>  
1039 </wsrm:SequenceAcknowledgement>
```

1040 4 Faults

1041 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. `Create`
1042 `Sequence Refused` is a possible fault reply for this operation. `Unknown Sequence` is a fault generated by
1043 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences
1044 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM
1045 on a Received message that did not use the protocol. All other faults in this section relate to known
1046 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those faults.
1047 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

1048 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault
1049 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1050 `http://docs.oasis-open.org/ws-rx/wsr/200702/fault`

1051 The faults defined in this section are generated if the condition stated in the preamble is met. Fault
1052 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1053 The definitions of faults use the following properties:

1054 [Code] The fault code.

1055 [Subcode] The fault subcode.

1056 [Reason] The English language reason element.

1057 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail
1058 element is defined for a fault, implementations MUST include the elements in the order that they are
1059 specified.

1060 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
1061 "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

1062 The properties above bind to a SOAP 1.2 fault as follows:

```
1063 <S:Envelope>
1064   <S:Header>
1065     <wsa:Action>
1066       http://docs.oasis-open.org/ws-rx/wsr/200702/fault
1067     </wsa:Action>
1068     <!-- Headers elided for brevity. -->
1069   </S:Header>
1070   <S:Body>
1071     <S:Fault>
1072       <S:Code>
1073         <S:Value> [Code] </S:Value>
1074         <S:Subcode>
1075           <S:Value> [Subcode] </S:Value>
1076         </S:Subcode>
1077       </S:Code>
1078       <S:Reason>
1079         <S:Text xml:lang="en"> [Reason] </S:Text>
1080       </S:Reason>
1081       <S:Detail>
1082         [Detail]
```

1083
1084
1085
1086
1087

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

1088 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM
1089 header block:

1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105

```
<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>
```

1106 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
1107 CreateSequence request message:

1108
1109
1110
1111
1112
1113
1114
1115

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

1116 4.1 SequenceFault Element

1117 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the
1118 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging
1119 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.
1120 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the
1121 SOAP 1.2 binding.

1122 The following exemplar defines its syntax:

1123
1124
1125
1126
1127

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

1128 The following describes the content model of the `SequenceFault` element.

1129 /wsrm: SequenceFault

1130 This is the element containing Sequence fault information for WS-ReliableMessaging

1131 /wsrm: SequenceFault/wsrm: FaultCode

- 1132 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this
 1133 element to a qualified name from the set of faults [Subcodes] defined below.
- 1134 `/wsrm:SequenceFault/wsrm:Detail`
 1135 This element, if present, carries application specific error information related to the fault being
 1136 described.
- 1137 `/wsrm:SequenceFault/wsrm:Detail/{any}`
 1138 The application specific error information related to the fault being described.
- 1139 `/wsrm:SequenceFault/wsrm:Detail/@{any}`
 1140 The application specific error information related to the fault being described.
- 1141 `/wsrm:SequenceFault/{any}`
 1142 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 1143 schema, to be passed.
- 1144 `/wsrm:SequenceFault/@{any}`
 1145 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 1146 to the element.

1147 4.2 Sequence Terminated

- 1148 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 1149 Endpoint of this decision.
- 1150 Properties:
- 1151 [Code] Sender or Receiver
- 1152 [Subcode] `wrm:SequenceTerminated`
- 1153 [Reason] The Sequence has been terminated due to an unrecoverable error.
- 1154 [Detail]
- 1155 `<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.

1156 4.3 Unknown Sequence

- 1157 Properties:
- 1158 [Code] Sender
- 1159 [Subcode] `wrm:UnknownSequence`

1160 [Reason] The value of `wsrc:Identifier` is not a known Sequence identifier.

1161 [Detail]

1162 `<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.

1163 4.4 Invalid Acknowledgement

1164 An example of when this fault is generated is when a message is Received by the RM Source containing
1165 a `SequenceAcknowledgement` covering messages that have not been sent.

1166 [Code] Sender

1167 [Subcode] `wsrc:InvalidAcknowledgement`

1168 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1169 [Detail]

1170 `<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.

1171 4.5 Message Number Rollover

1172 If the condition listed below is reached, the RM Destination MUST generate this fault.

1173 Properties:

1174 [Code] Sender

1175 [Subcode] wsrn:MessageNumberRollover

1176 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1177 [Detail]

```
1178 <wsrn:Identifier ...> xs:anyURI </wsrn:Identifier>
1179 <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.

1180 4.6 Create Sequence Refused

1181 Properties:

1182 [Code] Sender or Receiver

1183 [Subcode] wsrn:CreateSequenceRefused

1184 [Reason] The Create Sequence request has been refused by the RM Destination.

1185 [Detail]

```
1186 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.

1187 4.7 Sequence Closed

1188 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1189 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1190 is closed.

1191 Properties:

1192 [Code] Sender

1193 [Subcode] wsrn:SequenceClosed

1194 [Reason] The Sequence is closed and cannot accept new messages.

1195 [Detail]

1196 `<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.

1197 **4.8 WSRM Required**

1198 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1199 message that did not use this protocol.

1200 Properties:

1201 [Code] Sender

1202 [Subcode] wsrn:WSRMRequired

1203 [Reason] The RM Destination requires the use of WSRM.

1204 [Detail]

1205 `xs:any`

1206 5 Security Threats and Countermeasures

1207 This specification considers two sets of security requirements, those of the applications that use the WS-
1208 RM protocol and those of the protocol itself.

1209 This specification makes no assumptions about the security requirements of the applications that use WS-
1210 RM. However, once those requirements have been satisfied within a given operational context, the
1211 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1212 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1213 There are many other security concerns that one may need to consider when implementing or using this
1214 protocol. The material below should not be considered as a "check list". Implementers and users of this
1215 protocol are urged to perform a security analysis to determine their particular threat profile and the
1216 appropriate responses to those threats.

1217 Implementers are also advised that there is a core tension between security and reliable messaging that
1218 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1219 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1220 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1221 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1222 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1223 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1224 avoid and prevent this condition.

1225 5.1 Threats and Countermeasures

1226 The primary security requirement of this protocol is to protect the specified semantics and protocol
1227 invariants against various threats. The following sections describe several threats to the integrity and
1228 operation of this protocol and provide some general outlines of countermeasures to those threats.
1229 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1230 to all operational contexts.

1231 5.1.1 Integrity Threats

1232 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1233 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1234 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1235 to its intended message represents a threat to the WS-RM protocol.

1236 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM
1237 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1238 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be
1239 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1240 5.1.1.1 Countermeasures

1241 Integrity threats are generally countered via the use of digital signatures some level of the communication
1242 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1243 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers
1244 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in
1245 which they occur, implementations MUST allow for signatures that cover only these headers.

1246 5.1.2 Resource Consumption Threats

1247 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1248 implement that RM Destination. These resources can include network connections, database tables,
1249 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1250 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1251 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1252 message Delivery and use this Sequence to send a stream of very large messages to that service, making
1253 sure to omit message number “1” from that stream.

1254 5.1.2.1 Countermeasures

1255 There are a number of countermeasures against the described resource consumption threats. The
1256 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1257 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1258 some cases, allows the identity of any attackers to be determined.

1259 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and
1260 authenticate the RM Source that issued the `CreateSequence` message.

1261 5.1.3 Sequence Spoofing Threats

1262 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1263 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1264 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1265 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the
1266 current `MessageNumber` for their target Sequence.

1267 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP
1268 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence
1269 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to
1270 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a
1271 message that it sends to the `AcksTo` EPR of an RM Source.

1272 5.1.3.1 Sequence Hijacking

1273 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject
1274 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1275 messages.

1276 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a
1277 Sequence may be bound to some form of a security session in order to counter the threats described in
1278 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1279 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1280 established by the security infrastructure to make such determinations. Failure to observe this rule
1281 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1282 the ability to authenticate its peers even though the necessary security processing has taken place.

1283 5.1.3.2 Countermeasures

1284 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by
1285 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM
1286 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination
1287 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

1288 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it
1289 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the
1290 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it
1291 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the
1292 referenced Sequence.

1293 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and
1294 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify
1295 its Sequence peer it MUST be able to identify and authenticate the entity that sent the
1296 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1297 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that
1298 message and, if necessary, correlate this identity with the Sequence peer identity established at
1299 Sequence creation time.

1300 **5.2 Security Solutions and Technologies**

1301 The security threats described in the previous sections are neither new nor unique. The solutions that
1302 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1303 section maps the facilities provided by common web services security solutions against countermeasures
1304 described in the previous sections.

1305 Before continuing this discussion, however, some examination of the underlying requirements of the
1306 previously described countermeasures is necessary. Specifically it should be noted that the technique
1307 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1308 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization
1309 check against this authenticated identity and determines if the RM Source is permitted to create
1310 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1311 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1312 discussion of such facilities is considered to be beyond the scope of this specification.

1313 **5.2.1 Transport Layer Security**

1314 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1315 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1316 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1317 The description provided here is general in nature and is not intended to serve as a complete definition on
1318 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1319 choice of features as well as the manner in which they will be used. The mechanisms described in the
1320 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1321 requirements and constraints of the use of SSL/TLS.

1322 **5.2.1.1 Model**

1323 The basic model for using SSL/TLS is as follows:

- 1324 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1325 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1326 Destination.
- 1327 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1328 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1329 synchronous `CreateSequenceResponse` using the session established in (1).

- 1330 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1331 any and all messages or faults that refer to that Sequence.
- 1332 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1333 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1334 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1335 5.2.1.2 Countermeasure Implementation

1336 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1337 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the
1338 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1339 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1340 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1341 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification
1342 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1343 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1344 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1345 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1346 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1347 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1348 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1349 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1350 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1351 Acknowledgement) using BasicAuth.
- 1352 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1353 connection authenticates itself to the party accepting the connection using an X.509 certificate
1354 that is exchanged during the SSL/TLS handshake.

1355 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1356 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1357 Source is authorized to create a Sequence with the RM Destination.

1358 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1359 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1360 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1361 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1362 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1363 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1364 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1365 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1366 to protect that Sequence.

1367 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1368 countermeasures (such as associating specific authentication information with a Sequence) although such
1369 methods are not covered by this document.

1370 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1371 session) are outside the scope of this specification.

1372 5.2.2 SOAP Message Security

1373 The mechanisms described in WS-Security may be used in various ways to implement the
1374 countermeasures described in the previous sections. This specification advocates using the protocol
1375 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1376 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1377 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1378 The description provided here is general in nature and is not intended to serve as a complete definition on
1379 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1380 need to agree on the choice of features as well as the manner in which they will be used. The
1381 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1382 describe the requirements and constraints of the use of WS-SecureConversation.

1383 **5.2.2.1 Model**

1384 The basic model for using WS-SecureConversation is as follows:

- 1385 1 The RM Source and the RM Destination create a WS-SecureConversation security context. This
1386 may involve the participation of third parties such as a security token service. The tokens
1387 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service
1388 tickets).
- 1389 2 During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1390 context that will be used to protect the Sequence. This is done so that, in cases where the
1391 `CreateSequence` message is signed by more than one security context, the RM Source can
1392 indicate which security context should be used to protect the newly created Sequence.
- 1393 3 For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1394 associated with the security context to sign (as defined by WS-Security) at least the body and
1395 any relevant WS-RM-defined headers of any and all messages or faults that refer to that
1396 Sequence.

1397 **5.2.2.2 Countermeasure Implementation**

1398 Without relying upon any authentication information, the per-message signatures provide the necessary
1399 integrity qualities to counter the threats described in section 5.1.1.

1400 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1401 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1402 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1403 create a Sequence with the RM Destination.

1404 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1405 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1406 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1407 context rather than on any authentication claims that may have been established during security context
1408 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1409 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1410 document.

1411 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits
1412 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the
1413 association between a Sequence and its protecting security context cannot always be established
1414 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1415 `CreateSequenceResponse` messages may be signed by more than one security context.

1416 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1417 amending or renewing contexts) are outside the scope of this specification.

1418 6 Securing Sequences

1419 As noted in section 5, the RM Source and RM Destination should be able to protect their shared
1420 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1421 achieving this objective depending upon the underlying security infrastructure.

1422 6.1 Securing Sequences Using WS-Security

1423 One mechanism for protecting a Sequence is to include a security token using a
1424 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1425 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1426 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1427 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1428 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1429 there may be more than one token on a `CreateSequence` message or inferred from the communication
1430 context (e.g. transport protection).

1431 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1432 if the token being referenced supports such mechanism.

1433 The following exemplar defines the `CreateSequence` syntax when extended to include a
1434 `wsse:SecurityTokenReference`:

```
1435 <wsrm:CreateSequence ...>  
1436   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1437   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1438   <wsrm:Offer ...>  
1439     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1440     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1441     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1442     <wsrm:IncompleteSequenceBehavior>  
1443       wsm:IncompleteSequenceBehaviorType  
1444     </wsrm:IncompleteSequenceBehavior> ?  
1445     ...  
1446   </wsrm:Offer> ?  
1447   ...  
1448   <wsse:SecurityTokenReference>  
1449     ...  
1450   </wsse:SecurityTokenReference> ?  
1451   ...  
1452 </wsrm:CreateSequence>
```

1453 The following describes the content model of the additional `CreateSequence` elements.

1454 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1455 This element uses the extensibility mechanism defined for the `CreateSequence` element
1456 (defined in section 3.4) to communicate an explicit reference to the security token, using a
1457 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and
1458 Destination MUST use to authorize messages for the created (and, if present, the offered)
1459 Sequence(s). All subsequent messages related to the created (and, if present, the offered)
1460 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token
1461 (e.g., by using or deriving from a private or secret key).

1462 When a RM Source transmits a `CreateSequence` that has been extended to include a
1463 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1464 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1465 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1466 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1467 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1468 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1469 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1470 in WS-Security still applies.

1471 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1472 <wsmr:UsesSequenceSTR ... />
```

1473 The following describes the content model of the `UsesSequenceSTR` header block.

1474 `/wsmr:UsesSequenceSTR`

1475 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that
1476 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`
1477 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly
1478 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus
1479 aborting the requested Sequence creation.

1480 The following is an example of a `CreateSequence` message using the

1481 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1482 <soap:Envelope ...>  
1483 <soap:Header>  
1484 ...  
1485 <wsmr:UsesSequenceSTR soap:mustUnderstand='true' />  
1486 ...  
1487 </soap:Header>  
1488 <soap:Body>  
1489 <wsmr:CreateSequence>  
1490 <wsmr:AcksTo>  
1491 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1492 </wsmr:AcksTo>  
1493 <wsse:SecurityTokenReference>  
1494 ...  
1495 </wsse:SecurityTokenReference>  
1496 </wsmr:CreateSequence>  
1497 </soap:Body>  
1498 </soap:Envelope>
```

1499 6.2 Securing Sequences Using SSL/TLS

1500 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1501 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1502 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1503 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1504 SOAP header block within the `CreateSequence` message.

1505 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1506 <wsmr:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1507 The following describes the content model of the `UsesSequenceSSL` header block.

1508 `/wsmr:UsesSequenceSSL`

1509 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`
1510 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1511 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM
1512 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.
1513 The receiving RM Destination MUST understand and correctly implement the functionality
1514 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the
1515 requested Sequence creation.

1516 Note that the inclusion of the above header by the RM Source implies that all Sequence-related
1517 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1518 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1519 `CreateSequenceResponse` message.

1520 Appendix A. Schema

1521 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1522 Schema Part2] is located at:

1523 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1524 The following copy is provided for reference.

```
1525 <?xml version="1.0" encoding="UTF-8"?>
1526 <!-- Copyright(C) OASIS(R) 1993-2007. All Rights Reserved.
1527 OASIS trademark, IPR and other policies apply. -->
1528 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1529 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1530 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1531 rx/wsrn/200702" elementFormDefault="qualified"
1532 attributeFormDefault="unqualified">
1533 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1534 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1535 <!-- Protocol Elements -->
1536 <xs:complexType name="SequenceType">
1537 <xs:sequence>
1538 <xs:element ref="wsrm:Identifier"/>
1539 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1540 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1541 maxOccurs="unbounded"/>
1542 </xs:sequence>
1543 <xs:anyAttribute namespace="##other" processContents="lax"/>
1544 </xs:complexType>
1545 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1546 <xs:element name="SequenceAcknowledgement">
1547 <xs:complexType>
1548 <xs:sequence>
1549 <xs:element ref="wsrm:Identifier"/>
1550 <xs:choice>
1551 <xs:sequence>
1552 <xs:choice>
1553 <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1554 <xs:complexType>
1555 <xs:sequence/>
1556 <xs:attribute name="Upper" type="xs:unsignedLong"
1557 use="required"/>
1558 <xs:attribute name="Lower" type="xs:unsignedLong"
1559 use="required"/>
1560 <xs:anyAttribute namespace="##other" processContents="lax"/>
1561 </xs:complexType>
1562 </xs:element>
1563 <xs:element name="None">
1564 <xs:complexType>
1565 <xs:sequence/>
1566 </xs:complexType>
1567 </xs:element>
1568 </xs:choice>
1569 <xs:element name="Final" minOccurs="0">
1570 <xs:complexType>
1571 <xs:sequence/>
1572 </xs:complexType>
1573 </xs:element>
1574 </xs:sequence>
1575 <xs:element name="Nack" type="xs:unsignedLong"
```

```

1576 maxOccurs="unbounded" />
1577     </xs:choice>
1578     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1579 maxOccurs="unbounded" />
1580 </xs:sequence>
1581 <xs:anyAttribute namespace="##other" processContents="lax" />
1582 </xs:complexType>
1583 </xs:element>
1584 <xs:complexType name="AckRequestedType">
1585 <xs:sequence>
1586 <xs:element ref="wsrm:Identifier" />
1587 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1588 maxOccurs="unbounded" />
1589 </xs:sequence>
1590 <xs:anyAttribute namespace="##other" processContents="lax" />
1591 </xs:complexType>
1592 <xs:element name="AckRequested" type="wsrm:AckRequestedType" />
1593 <xs:element name="Identifier">
1594 <xs:complexType>
1595 <xs:annotation>
1596 <xs:documentation>
1597     This type is for elements whose [children] is an anyURI and can have
1598 arbitrary attributes.
1599 </xs:documentation>
1600 </xs:annotation>
1601 <xs:simpleContent>
1602 <xs:extension base="xs:anyURI">
1603 <xs:anyAttribute namespace="##other" processContents="lax" />
1604 </xs:extension>
1605 </xs:simpleContent>
1606 </xs:complexType>
1607 </xs:element>
1608 <xs:element name="Address">
1609 <xs:complexType>
1610 <xs:simpleContent>
1611 <xs:extension base="xs:anyURI">
1612 <xs:anyAttribute namespace="##other" processContents="lax" />
1613 </xs:extension>
1614 </xs:simpleContent>
1615 </xs:complexType>
1616 </xs:element>
1617 <xs:simpleType name="MessageNumberType">
1618 <xs:restriction base="xs:unsignedLong">
1619 <xs:minInclusive value="1" />
1620 <xs:maxInclusive value="9223372036854775807" />
1621 </xs:restriction>
1622 </xs:simpleType>
1623 <!-- Fault Container and Codes -->
1624 <xs:simpleType name="FaultCodes">
1625 <xs:restriction base="xs:QName">
1626 <xs:enumeration value="wsrm:SequenceTerminated" />
1627 <xs:enumeration value="wsrm:UnknownSequence" />
1628 <xs:enumeration value="wsrm:InvalidAcknowledgement" />
1629 <xs:enumeration value="wsrm:MessageNumberRollover" />
1630 <xs:enumeration value="wsrm:CreateSequenceRefused" />
1631 <xs:enumeration value="wsrm:SequenceClosed" />
1632 <xs:enumeration value="wsrm:WSRMRequired" />
1633 </xs:restriction>
1634 </xs:simpleType>
1635 <xs:complexType name="SequenceFaultType">
1636 <xs:sequence>
1637 <xs:element name="FaultCode" type="wsrm:FaultCodes" />
1638 <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0" />
1639 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1640 maxOccurs="unbounded" />
1641 </xs:sequence>
1642 <xs:anyAttribute namespace="##other" processContents="lax" />
1643 </xs:complexType>
1644 <xs:complexType name="DetailType">
1645 <xs:sequence>
1646 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1647 maxOccurs="unbounded" />
1648 </xs:sequence>
1649 <xs:anyAttribute namespace="##other" processContents="lax" />
1650 </xs:complexType>
1651 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType" />
1652 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType" />
1653 <xs:element name="CreateSequenceResponse"
1654 type="wsrm:CreateSequenceResponseType" />
1655 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType" />
1656 <xs:element name="CloseSequenceResponse"
1657 type="wsrm:CloseSequenceResponseType" />
1658 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType" />
1659 <xs:element name="TerminateSequenceResponse"
1660 type="wsrm:TerminateSequenceResponseType" />
1661 <xs:complexType name="CreateSequenceType">
1662 <xs:sequence>
1663 <xs:element ref="wsrm:AcksTo" />
1664 <xs:element ref="wsrm:Expires" minOccurs="0" />
1665 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0" />
1666 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1667 maxOccurs="unbounded">
1668 <xs:annotation>
1669 <xs:documentation>
1670 It is the authors intent that this extensibility be used to
1671 transfer a Security Token Reference as defined in WS-Security.
1672 </xs:documentation>
1673 </xs:annotation>
1674 </xs:any>
1675 </xs:sequence>
1676 <xs:anyAttribute namespace="##other" processContents="lax" />
1677 </xs:complexType>
1678 <xs:complexType name="CreateSequenceResponseType">
1679 <xs:sequence>
1680 <xs:element ref="wsrm:Identifier" />
1681 <xs:element ref="wsrm:Expires" minOccurs="0" />
1682 <xs:element name="IncompleteSequenceBehavior"
1683 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0" />
1684 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0" />
1685 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1686 maxOccurs="unbounded" />
1687 </xs:sequence>
1688 <xs:anyAttribute namespace="##other" processContents="lax" />
1689 </xs:complexType>
1690 <xs:complexType name="CloseSequenceType">
1691 <xs:sequence>
1692 <xs:element ref="wsrm:Identifier" />
1693 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1694 minOccurs="0" />
1695 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1696 maxOccurs="unbounded" />
1697 </xs:sequence>
1698 <xs:anyAttribute namespace="##other" processContents="lax" />
1699 </xs:complexType>
1700 <xs:complexType name="CloseSequenceResponseType">
1701 <xs:sequence>
1702 <xs:element ref="wsrm:Identifier" />
1703 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1704 maxOccurs="unbounded" />
1705     </xs:sequence>
1706     <xs:anyAttribute namespace="##other" processContents="lax" />
1707 </xs:complexType>
1708 <xs:complexType name="TerminateSequenceType">
1709     <xs:sequence>
1710         <xs:element ref="wsrm:Identifier" />
1711         <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1712 minOccurs="0" />
1713         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1714 maxOccurs="unbounded" />
1715     </xs:sequence>
1716     <xs:anyAttribute namespace="##other" processContents="lax" />
1717 </xs:complexType>
1718 <xs:complexType name="TerminateSequenceResponseType">
1719     <xs:sequence>
1720         <xs:element ref="wsrm:Identifier" />
1721         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1722 maxOccurs="unbounded" />
1723     </xs:sequence>
1724     <xs:anyAttribute namespace="##other" processContents="lax" />
1725 </xs:complexType>
1726 <xs:element name="AcksTo" type="wsa:EndpointReferenceType" />
1727 <xs:complexType name="OfferType">
1728     <xs:sequence>
1729         <xs:element ref="wsrm:Identifier" />
1730         <xs:element name="Endpoint" type="wsa:EndpointReferenceType" />
1731         <xs:element ref="wsrm:Expires" minOccurs="0" />
1732         <xs:element name="IncompleteSequenceBehavior"
1733 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0" />
1734         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1735 maxOccurs="unbounded" />
1736     </xs:sequence>
1737     <xs:anyAttribute namespace="##other" processContents="lax" />
1738 </xs:complexType>
1739 <xs:complexType name="AcceptType">
1740     <xs:sequence>
1741         <xs:element ref="wsrm:AcksTo" />
1742         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1743 maxOccurs="unbounded" />
1744     </xs:sequence>
1745     <xs:anyAttribute namespace="##other" processContents="lax" />
1746 </xs:complexType>
1747 <xs:element name="Expires">
1748     <xs:complexType>
1749         <xs:simpleContent>
1750             <xs:extension base="xs:duration">
1751                 <xs:anyAttribute namespace="##other" processContents="lax" />
1752             </xs:extension>
1753         </xs:simpleContent>
1754     </xs:complexType>
1755 </xs:element>
1756 <xs:simpleType name="IncompleteSequenceBehaviorType">
1757     <xs:restriction base="xs:string">
1758         <xs:enumeration value="DiscardEntireSequence" />
1759         <xs:enumeration value="DiscardFollowingFirstGap" />
1760         <xs:enumeration value="NoDiscard" />
1761     </xs:restriction>
1762 </xs:simpleType>
1763 <xs:element name="UsesSequenceSTR">
1764     <xs:complexType>
1765         <xs:sequence />
1766         <xs:anyAttribute namespace="##other" processContents="lax" />
1767     </xs:complexType>

```

```
1768 </xs:element>
1769 <xs:element name="UsesSequenceSSL">
1770 <xs:complexType>
1771 <xs:sequence/>
1772 <xs:anyAttribute namespace="##other" processContents="lax" />
1773 </xs:complexType>
1774 </xs:element>
1775 <xs:element name="UnsupportedElement">
1776 <xs:simpleType>
1777 <xs:restriction base="xs:QName" />
1778 </xs:simpleType>
1779 </xs:element>
1780 </xs:schema>
```

1781 Appendix B. WSDL

1782 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1783 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1784 present in exchanges with that endpoint.

1785 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1786 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1787 for a higher-level mechanism to indicate that WS-RM is engaged.

1788 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1789 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

1790 The following non-normative copy is provided for reference.

```
1791 <?xml version="1.0" encoding="utf-8"?>
1792 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1793 OASIS trademark, IPR and other policies apply. -->
1794 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1795 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1796 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1797 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1798 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1799 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL"
1800 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL">
1801
1802   <wSDL:types>
1803     <xs:schema
1804       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1805       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1806       200702.xsd"/>
1807     </xs:schema>
1808   </wSDL:types>
1809
1810   <wSDL:message name="CreateSequence">
1811     <wSDL:part name="create" element="rm:CreateSequence"/>
1812   </wSDL:message>
1813   <wSDL:message name="CreateSequenceResponse">
1814     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1815   </wSDL:message>
1816   <wSDL:message name="CloseSequence">
1817     <wSDL:part name="close" element="rm:CloseSequence"/>
1818   </wSDL:message>
1819   <wSDL:message name="CloseSequenceResponse">
1820     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1821   </wSDL:message>
1822   <wSDL:message name="TerminateSequence">
1823     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1824   </wSDL:message>
1825   <wSDL:message name="TerminateSequenceResponse">
1826     <wSDL:part name="terminateResponse"
1827     element="rm:TerminateSequenceResponse"/>
1828   </wSDL:message>
1829
1830   <wSDL:portType name="SequenceAbstractPortType">
1831     <wSDL:operation name="CreateSequence">
1832       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1833       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1834       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1835 wsam:Action="http://docs.oasis-open.org/ws-
1836 rx/wsrn/200702/CreateSequenceResponse"/>
1837 </wsdl:operation>
1838 <wsdl:operation name="CloseSequence">
1839 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1840 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1841 <wsdl:output message="tns:CloseSequenceResponse"
1842 wsam:Action="http://docs.oasis-open.org/ws-
1843 rx/wsrn/200702/CloseSequenceResponse"/>
1844 </wsdl:operation>
1845 <wsdl:operation name="TerminateSequence">
1846 <wsdl:input message="tns:TerminateSequence"
1847 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1848 <wsdl:output message="tns:TerminateSequenceResponse"
1849 wsam:Action="http://docs.oasis-open.org/ws-
1850 rx/wsrn/200702/TerminateSequenceResponse"/>
1851 </wsdl:operation>
1852 </wsdl:portType>
1853
1854 </wsdl:definitions>
```

1855 Appendix C. Message Examples

1856 Appendix C.1 Create Sequence

1857 Create Sequence

```
1858 <?xml version="1.0" encoding="UTF-8"?>
1859 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1860 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1861 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1862   <S:Header>
1863     <wsa:MessageID>
1864       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1865     </wsa:MessageID>
1866     <wsa:To>http://example.com/serviceB/123</wsa:To>
1867     <wsa:Action>http://docs.oasis-open.org/ws-
1868 rx/wsmr/200702/CreateSequence</wsa:Action>
1869     <wsa:ReplyTo>
1870     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1871     </wsa:ReplyTo>
1872   </S:Header>
1873   <S:Body>
1874     <wsmr:CreateSequence>
1875       <wsmr:AcksTo>
1876         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1877       </wsmr:AcksTo>
1878     </wsmr:CreateSequence>
1879   </S:Body>
1880 </S:Envelope>
```

1881 Create Sequence Response

```
1882 <?xml version="1.0" encoding="UTF-8"?>
1883 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1884 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1885 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1886   <S:Header>
1887     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1888     <wsa:RelatesTo>
1889       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1890     </wsa:RelatesTo>
1891     <wsa:Action>
1892       http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1893     </wsa:Action>
1894   </S:Header>
1895   <S:Body>
1896     <wsmr:CreateSequenceResponse>
1897       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1898     </wsmr:CreateSequenceResponse>
1899   </S:Body>
1900 </S:Envelope>
```

1901 Appendix C.2 Initial Transmission

1902 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1903 figure. The three messages have the following headers; the third message is identified as the last
1904 message in the Sequence:

1905 Message 1

```
1906 <?xml version="1.0" encoding="UTF-8"?>
1907 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1908 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1909 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1910   <S:Header>
1911     <wsa:MessageID>
1912       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1913     </wsa:MessageID>
1914     <wsa:To>http://example.com/serviceB/123</wsa:To>
1915     <wsa:From>
1916       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1917     </wsa:From>
1918     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1919     <wsmr:Sequence>
1920       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1921       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1922     </wsmr:Sequence>
1923   </S:Header>
1924   <S:Body>
1925     <!-- Some Application Data -->
1926   </S:Body>
1927 </S:Envelope>
```

1928 Message 2

```
1929 <?xml version="1.0" encoding="UTF-8"?>
1930 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1931 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1932 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1933   <S:Header>
1934     <wsa:MessageID>
1935       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1936     </wsa:MessageID>
1937     <wsa:To>http://example.com/serviceB/123</wsa:To>
1938     <wsa:From>
1939       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1940     </wsa:From>
1941     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1942     <wsmr:Sequence>
1943       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1944       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1945     </wsmr:Sequence>
1946   </S:Header>
1947   <S:Body>
1948     <!-- Some Application Data -->
1949   </S:Body>
1950 </S:Envelope>
```

1951 Message 3

```
1952 <?xml version="1.0" encoding="UTF-8"?>
1953 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1954 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1955 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1956   <S:Header>
1957     <wsa:MessageID>
1958       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1959     </wsa:MessageID>
1960     <wsa:To>http://example.com/serviceB/123</wsa:To>
1961     <wsa:From>
1962       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1963     </wsa:From>
1964     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
```

```

1965 <wsrm:Sequence>
1966   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1967   <wsrm:MessageNumber>3</wsrm:MessageNumber>
1968 </wsrm:Sequence>
1969 <wsrm:AckRequested>
1970   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1971 </wsrm:AckRequested>
1972 </S:Header>
1973 <S:Body>
1974   <!-- Some Application Data -->
1975 </S:Body>
1976 </S:Envelope>

```

1977 **Appendix C.3 First Acknowledgement**

1978 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1979 responds with an Acknowledgement for messages 1 and 3:

```

1980 <?xml version="1.0" encoding="UTF-8"?>
1981 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1982 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1983 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1984   <S:Header>
1985     <wsa:MessageID>
1986       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1987     </wsa:MessageID>
1988     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1989     <wsa:From>
1990       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1991     </wsa:From>
1992     <wsa:Action>
1993       http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1994     </wsa:Action>
1995     <wsrm:SequenceAcknowledgement>
1996       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1997       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1998       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1999     </wsrm:SequenceAcknowledgement>
2000   </S:Header>
2001   <S:Body/>
2002 </S:Envelope>

```

2003 **Appendix C.4 Retransmission**

2004 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2005 requests an Acknowledgement:

```

2006 <?xml version="1.0" encoding="UTF-8"?>
2007 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2008 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2009 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2010   <S:Header>
2011     <wsa:MessageID>
2012       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2013     </wsa:MessageID>
2014     <wsa:To>http://example.com/serviceB/123</wsa:To>
2015     <wsa:From>
2016       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2017     </wsa:From>
2018     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2019   <wsrm:Sequence>

```

```

2020     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2021     <wsrm:MessageNumber>2</wsrm:MessageNumber>
2022     </wsrm:Sequence>
2023     <wsrm:AckRequested>
2024     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2025     </wsrm:AckRequested>
2026     </S:Header>
2027     <S:Body>
2028     <!-- Some Application Data -->
2029     </S:Body>
2030     </S:Envelope>

```

2031 Appendix C.5 Termination

2032 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2033 be terminated:

```

2034 <?xml version="1.0" encoding="UTF-8"?>
2035 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2036 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2037 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2038   <S:Header>
2039     <wsa:MessageID>
2040       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2041     </wsa:MessageID>
2042     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2043     <wsa:From>
2044       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2045     </wsa:From>
2046     <wsa:Action>
2047       http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2048     </wsa:Action>
2049     <wsrm:SequenceAcknowledgement>
2050       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2051       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2052     </wsrm:SequenceAcknowledgement>
2053   </S:Header>
2054   <S:Body/>
2055 </S:Envelope>

```

2056 Terminate Sequence

```

2057 <?xml version="1.0" encoding="UTF-8"?>
2058 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2059 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2060 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2061   <S:Header>
2062     <wsa:MessageID>
2063       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2064     </wsa:MessageID>
2065     <wsa:To>http://example.com/serviceB/123</wsa:To>
2066     <wsa:Action>
2067       http://docs.oasis-open.org/ws-rx/wsr/200702/TerminateSequence
2068     </wsa:Action>
2069     <wsa:From>
2070       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2071     </wsa:From>
2072   </S:Header>
2073   <S:Body>
2074     <wsrm:TerminateSequence>
2075       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2076       <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2077     </wsrm:TerminateSequence>

```

```
2078 </S:Body>
2079 </S:Envelope>
```

2080 Terminate Sequence Response

```
2081 <?xml version="1.0" encoding="UTF-8"?>
2082 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2083 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2084 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2085   <S:Header>
2086     <wsa:MessageID>
2087       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2088     </wsa:MessageID>
2089     <wsa:To>http://example.com/serviceA/789</wsa:To>
2090     <wsa:Action>
2091       http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
2092     </wsa:Action>
2093     <wsa:RelatesTo>
2094       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2095     </wsa:RelatesTo>
2096     <wsa:From>
2097       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2098     </wsa:From>
2099   </S:Header>
2100   <S:Body>
2101     <wsm:TerminateSequenceResponse>
2102       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2103     </wsm:TerminateSequenceResponse>
2104   </S:Body>
2105 </S:Envelope>
```

2106 Appendix D. State Tables

2107 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2108 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

2109 Legend:

2110 The first column of these tables contains the motivating event and has the following format:

Event 2111
<i>Event name</i> [source] {ref}

2112 Where:

- 2113 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2114 described by the specification.
- 2115 • [source]: indicates the source of the event; one of:
 - 2116 ○ [msg] a Received message
 - 2117 ○ [int]: an internal event such as the firing of a timer
 - 2118 ○ [app]: the application
 - 2119 ○ [unspec]: the source is unspecified

2120 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}

2121 Where:

- 2122 • action to take: indicates that the state machine performs the following action. Actions surrounded
2123 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2124 "Transmit"
- 2125 • [next state]: indicates the state to which the state machine will advance upon the performance of
2126 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2127 • {ref} is a reference to the document section describing the behavior in this cell

2128 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2129 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2130 described in this specification and does not indicate normal protocol operation. Implementations MAY
2131 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2132 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2133 combinations.

2134 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}

2135 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}	

2136 Appendix E. Acknowledgments

2137 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2138 authors:

2139	Ruslan Bilorusets, BEA	2151	Amelia Lewis, TIBCO Software
2140	Don Box, Microsoft	2152	Rodney Limprecht, Microsoft
2141	Luis Felipe Cabrera, Microsoft	2153	Steve Lucco, Microsoft
2142	Doug Davis, IBM	2154	Don Mullen, TIBCO Software
2143	Donald Ferguson, IBM	2155	Anthony Nadalin, IBM
2144	Christopher Ferris, IBM	2156	Mark Nottingham, BEA
2145	Tom Freund, IBM	2157	David Orchard, BEA
2146	Mary Ann Hondo, IBM	2158	Jamie Roots, IBM
2147	John Ibbotson, IBM	2159	Shivajee Samdarshi, TIBCO Software
2148	Lei Jin, BEA	2160	John Shewchuk, Microsoft
2149	Chris Kaler, Microsoft	2161	Tony Storey, IBM
2150	David Langworthy-Editor, Microsoft		

2162 The following individuals have provided invaluable input into the initial contribution:

2163	Keith Ballinger, Microsoft	2177	David Ingham, Microsoft
2164	Stefan Batres, Microsoft	2178	Gopal Kakivaya, Microsoft
2165	Rebecca Bergersen, Iona	2179	Johannes Klein, Microsoft
2166	Allen Brown, Microsoft	2180	Frank Leymann, IBM
2167	Michael Conner, IBM	2181	Martin Nally, IBM
2168	George Copeland, Microsoft	2182	Peter Niblett, IBM
2169	Francisco Curbera, IBM	2183	Jeffrey Schlimmer, Microsoft
2170	Paul Fremantle, IBM	2184	James Snell, IBM
2171	Steve Graham, IBM	2185	Keith Stobie, Microsoft
2172	Pat Helland, Microsoft	2186	Satish Thatte, Microsoft
2173	Rick Hill, Microsoft	2187	Stephen Todd, IBM
2174	Scott Hinkelman, IBM	2188	Sanjiva Weerawarana, IBM
2175	Tim Holloway, IBM	2189	Roger Wolter, Microsoft
2176	Efim Hudis, Microsoft		

2190 The following individuals were members of the committee during the development of this specification:

2191	Abbie Barbir, Nortel	2210	Robert Freund, Hitachi
2192	Charlton Barreto, Adobe	2211	Peter Furniss, Erebore
2193	Stefan Batres, Microsoft	2212	Marc Goodner, Microsoft
2194	Hamid Ben Malek, Fujitsu	2213	Alastair Green, Choreology
2195	Andreas Bjarlestam, Ericsson	2214	Mike Grogan, Sun
2196	Toufic Boubez, Layer 7	2215	Ondrej Hrebicek, Microsoft
2197	Doug Bunting, Sun	2216	Kazunori Iwasa, Fujitsu
2198	Lloyd Burch, Novell	2217	Chamikara Jayalath, WSO2
2199	Steve Carter, Novell	2218	Lei Jin, BEA
2200	Martin Chapman, Oracle	2219	Ian Jones, BTplc
2201	Dave Chappell, Sonic	2220	Anish Karmarkar, Oracle
2202	Paul Cotton, Microsoft	2221	Paul Knight, Nortel
2203	Glen Daniels, Sonic	2222	Dan Leshchiner, Tibco
2204	Doug Davis, IBM	2223	Mark Little, JBoss
2205	Blake Dournaee, Intel	2224	Lily Liu, webMethods
2206	Jacques Durand, Fujitsu	2225	Matt Lovett, IBM
2207	Colleen Evans, Microsoft	2226	Ashok Malhotra, Oracle
2208	Christopher Ferris, IBM	2227	Jonathan Marsh, Microsoft
2209	Paul Fremantle, WSO2	2228	Daniel Millwood, IBM

2229	Jeff Mischkinsky, Oracle	2240	Stefan Rossmannith, SAP
2230	Nilo Mitra, Ericsson	2241	Tom Rutt, Fujitsu
2231	Peter Niblett, IBM	2242	Rich Salz, IBM
2232	Duane Nickull, Adobe	2243	Shivajee Samdarshi, Tibco
2233	Eisaku Nishiyama, Hitachi	2244	Vladimir Videlov, SAP
2234	Dave Orchard, BEA	2245	Claus von Riegen, SAP
2235	Chouthri Palanisamy, NEC	2246	Pete Wenzel, Sun
2236	Sanjay Patil, SAP	2247	Steve Winkler, SAP
2237	Gilbert Pilz, BEA	2248	Ümit Yalçinalp, SAP
2238	Martin Raepfle, SAP	2249	Nobuyuki Yamamoto, Hitachi
2239	Eric Rajkovic, Oracle		
2250			