



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

3 Committee Draft 02

4 20 November 2008

5 Specification URIs:

6 This Version:

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

10 Previous Version:

- 11 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.pdf>
- 12 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.html>
- 13 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01-e1.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® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

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

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

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

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

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

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

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

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

109 Table of Contents

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

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

158 1 Introduction

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

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

168 1.1 Terminology

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

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

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

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

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

197 1.2 Normative References

- 198 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC
199 2119, Harvard University, March 1997
200 <http://www.ietf.org/rfc/rfc2119.txt>
- 201 **[WS-RM Policy]** OASIS WS-RX Technical Committee Draft, "Web Services Reliable Messaging
202 Policy Assertion(WS-RM Policy)," November 2008
203 <http://docs.oasis-open.org/ws-rx/wsrmp/v1.2/wsrmp.pdf>
- 204 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
205 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 206 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June
207 2003.
208 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 209 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
210 Generic Syntax," RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January
211 2005.
212 <http://ietf.org/rfc/rfc3986>
- 213 **[UUID]** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN
214 Namespace," RFC 4122, Microsoft, Refactored Networks - LLC, DataPower
215 Technology Inc, July 2005
216 <http://www.ietf.org/rfc/rfc4122.txt>
- 217 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth
218 Edition)", September 2006.
219 <http://www.w3.org/TR/REC-xml/>
- 220 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.
221 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 222 **[XML-Schema Part1]** W3C Recommendation, "XML Schema Part 1: Structures," October 2004.
223 <http://www.w3.org/TR/xmlschema-1/>
- 224 **[XML-Schema Part2]** W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
225 <http://www.w3.org/TR/xmlschema-2/>
- 226 **[XPath 1.0]** W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16
227 November 1999.
228 <http://www.w3.org/TR/xpath>
- 229 **[WSDL 1.1]** W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
230 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 231 **[WS-Addressing]** W3C Recommendation, "Web Services Addressing 1.0 – Core," May 2006.
232 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
233 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding," May
234 2006
235 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

236 1.3 Non-Normative References

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

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

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

250 **[WS-Policy]** W3C Recommendation, "Web Services Policy 1.5 - Framework," September
 251 2007.
 252 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>

253 **[WS-PolicyAttachment]** W3C Recommendation, "Web Services Policy 1.5 - Attachment,"
 254 September 2007.
 255 <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904>

256 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 257 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",
 258 OASIS Standard 200401, March 2004.
 259 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
 260

261
 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]** OASIS WS-SX Technical Committee Editor Draft, "WS-
 272 SecureConversation 1.4"
 273 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512>

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.

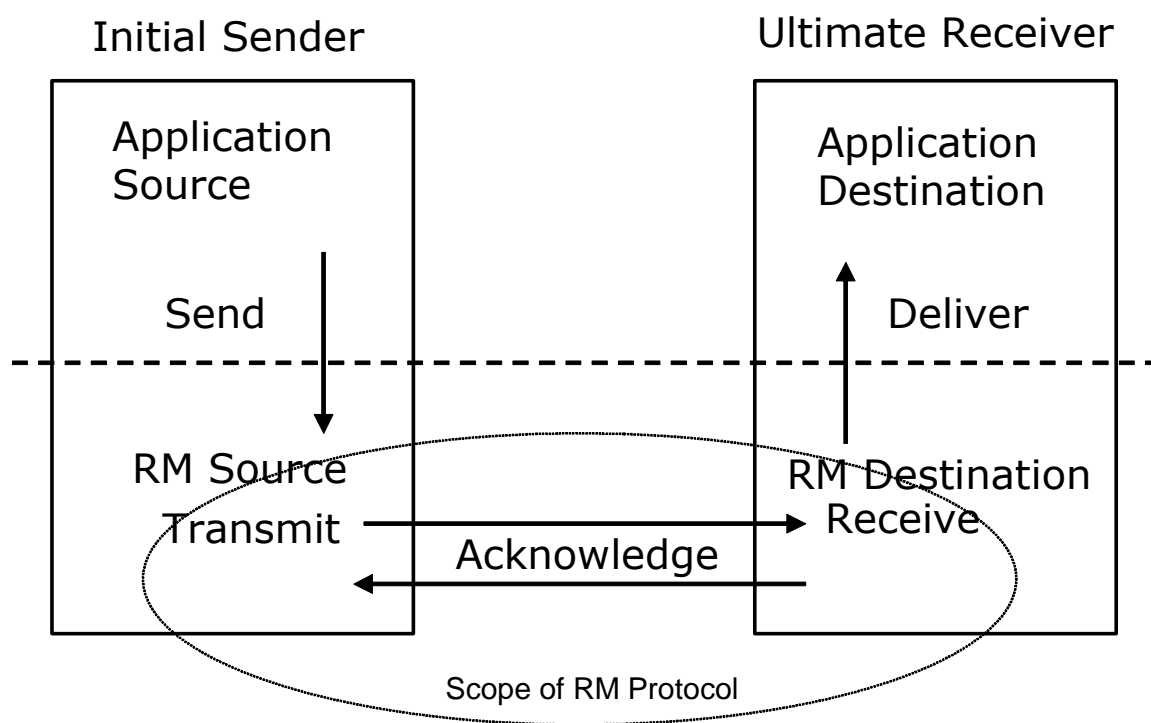
294 **2 Reliable Messaging Model**

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.

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

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

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

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

391 2.2 Protocol Preconditions

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

- 394 • For any single message exchange the RM Source **MUST** have an endpoint reference that
395 uniquely identifies the RM Destination Endpoint.
- 396 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 397 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's
398 policies.
- 399 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**
400 have a security context.

401 2.3 Protocol Invariants

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

- 403 • The RM Source **MUST** assign each message within a `Sequence` a message number (defined
404 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers
405 **MUST** be assigned in the same order in which messages are sent by the Application Source.
- 406 • Within every `Acknowledgement Message` it issues, the RM Destination **MUST** include one or
407 more `AcknowledgementRange` child elements that contain, in their collective ranges, the
408 message number of every message accepted by the RM Destination. The RM Destination **MUST**
409 exclude, in the `AcknowledgementRange` elements, the message numbers of any messages it
410 has not accepted. If no messages have been received the RM Destination **MUST** return `None`
411 instead of an `AcknowledgementRange(s)`. The RM Destination **MAY** transmit a `Nack` for a
412 specific message or messages instead of an `AcknowledgementRange(s)`.
- 413 • While the `Sequence` is not closed or terminated, the RM Source **SHOULD** retransmit
414 unacknowledged messages.

415 2.4 Delivery Assurances

416 This section defines a number of Delivery Assurance assertions, which can be supported by RM Sources
417 and RM Destinations. These assertions can be specified as policy assertions using the WS-Policy
418 framework [[WS-Policy](#)]. For details on this see the WSRM Policy specification [[WS-RM](#)
419 [Policy](#)].

420 `AtLeastOnce`

421 Each message is to be delivered at least once, or else an error **MUST** be raised by the RM
422 Source and/or RM Destination. The requirement on an RM Source is that it **SHOULD** retry

423 transmission of every message sent by the Application Source until it receives an
424 acknowledgement from the RM Destination. The requirement on the RM Destination is that it
425 SHOULD retry the transfer to the Application Destination of any message that it accepts from the
426 RM Source, until that message has been successfully delivered. There is no requirement for the
427 RM Destination to apply duplicate message filtering.

428 AtMostOnce

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

433 ExactlyOnce

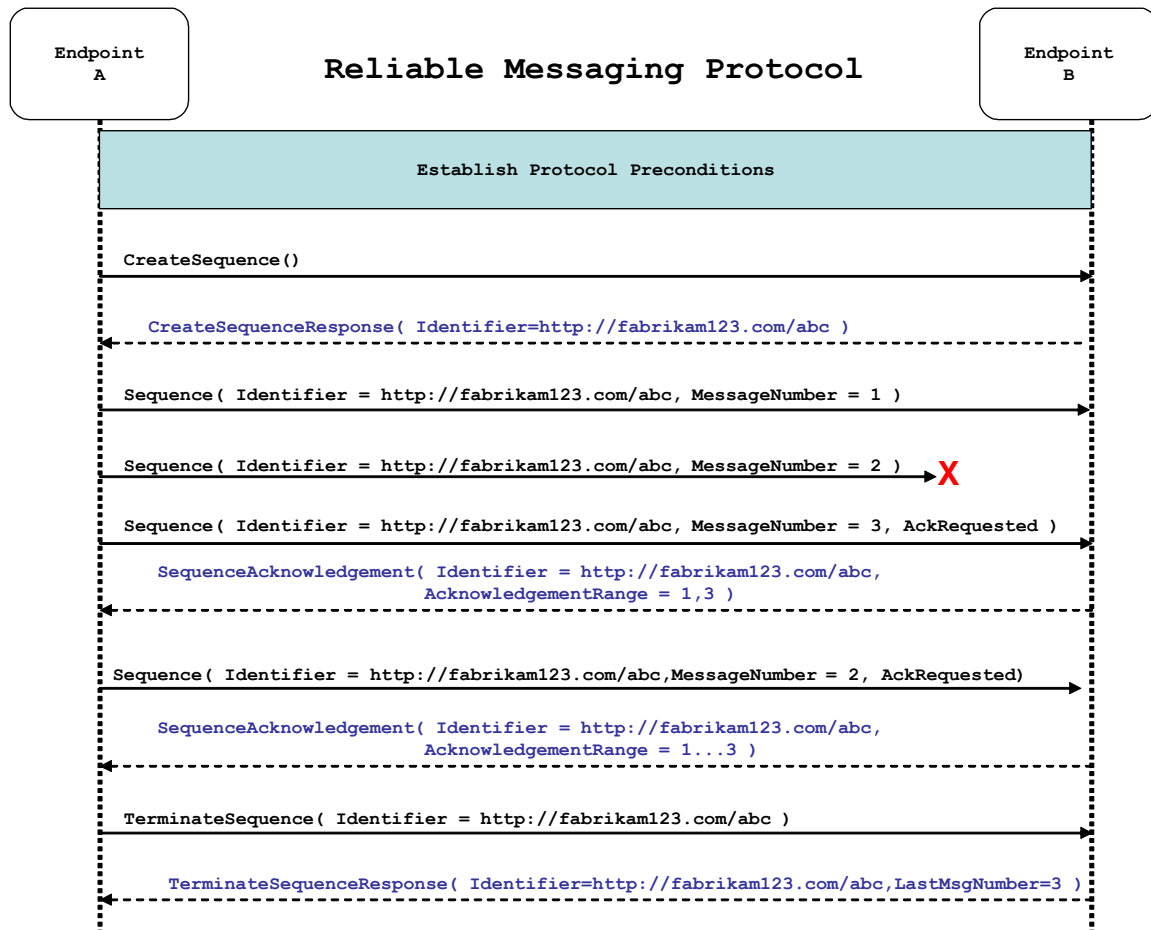
434 Each message is to be delivered exactly once; if a message cannot be delivered then an error
435 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is
436 that it SHOULD retry transmission of every message sent by the Application Source until it
437 receives an acknowledgement from the RM Destination. The requirement on the RM Destination
438 is that it SHOULD retry the transfer to the Application Destination of any message that it accepts
439 from the RM Source until that message has been successfully delivered, and that it MUST NOT
440 deliver a duplicate of a message that has already been delivered.

441 InOrder

442 Messages from each individual Sequence are to be delivered in the same order they have been
443 sent by the Application Source. The requirement on an RM Source is that it MUST ensure that the
444 ordinal position of each message in the Sequence (as indicated by a message Sequence number)
445 is consistent with the order in which the messages have been sent from the Application Source.
446 The requirement on the RM Destination is that it MUST deliver received messages for each
447 Sequence in the order indicated by the message numbering. This DeliveryAssurance can be used
448 in combination with any of the AtLeastOnce, AtMostOnce or ExactlyOnce assertions, and the
449 requirements of those assertions MUST also be met. In particular if the AtLeastOnce or
450 ExactlyOnce assertion applies and the RM Destination detects a gap in the Sequence then the
451 RM Destination MUST NOT deliver any subsequent messages from that Sequence until the
452 missing messages are received or until the Sequence is closed.

453 2.5 Example Message Exchange

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



455 Figure 2: The WS-ReliableMessaging Protocol

- 456 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
457 and establishing trust.
- 458 4.2. The RM Source requests creation of a new Sequence.
- 459 4.3. The RM Destination creates a new Sequence and returns its unique Identifier.
- 460 4.4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber
461 1. In the figure above, the RM Source sends 3 messages in the Sequence.
- 462 4.5. The 2nd message in the Sequence is lost in transit.
- 463 4.6. The 3rd message is the last in this Sequence and the RM Source includes an AckRequested
464 header to ensure that it gets a timely SequenceAcknowledgement for the Sequence.
- 465 4.7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving
466 the RM Source's AckRequested header.
- 467 4.8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new
468 message from the perspective of the underlying transport, but it has the same Sequence
469 Identifier and MessageNumber so the RM Destination can recognize it as a duplicate of the
470 earlier message, in case the original and retransmitted messages are both Received. The RM
471 Source includes an AckRequested header in the retransmitted message so the RM Destination
472 will expedite an acknowledgement.

473 | 4.9. The RM Destination Receives the second transmission of the message with `MessageNumber` 2
474 | and acknowledges receipt of message numbers 1, 2, and 3.

475 | 4.10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence`
476 | message to the RM Destination indicating that the Sequence is completed. The
477 | `TerminateSequence` message indicates that message number 3 was the last message in the
478 | Sequence. The RM Destination then reclaims any resources associated with the Sequence.

479 | 4.11. The RM Destination Receives the `TerminateSequence` message indicating that the RM
480 | Source will not be sending any more messages. The RM Destination sends a
481 | `TerminateSequenceResponse` message to the RM Source and reclaims any resources
482 | associated with the Sequence.

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

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

496 3 RM Protocol Elements

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

499 3.1 Considerations on the Use of Extensibility Points

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

504 3.2 Considerations on the Use of "Piggy-Backing"

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

514 3.3 Composition with WS-Addressing

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

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

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

- 524 4.2. When an Endpoint generates an Acknowledgement Message that has no element content in the
525 SOAP body, then the value of the `wsa:Action` IRI MUST be:

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

- 527 4.3. When an Endpoint generates an Acknowledgement Request that has no element content in the
528 SOAP body, then the value of the `wsa:Action` IRI MUST be:

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

- 530 2.4. When an Endpoint generates an RM fault as defined in section 4 below, the value of the
531 `wsa:Action` IRI MUST be as defined in section 4 below.

532 3.4 Sequence Creation

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

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

540 The following exemplar defines the `CreateSequence` syntax:

```
541 <wsrm:CreateSequence ...>
542   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
543   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
544   <wsrm:Offer ...>
545     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
546     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
547     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
548     <wsrm:IncompleteSequenceBehavior>
549       wsrml:IncompleteSequenceBehaviorType
550     </wsrm:IncompleteSequenceBehavior> ?
551     ...
552   </wsrm:Offer> ?
553   ...
554 </wsrm:CreateSequence>
```

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

556 `/wsrm:CreateSequence`

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

561 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

571 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

631 /wsmr:CreateSequence/{any}

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

634 /wsmr:CreateSequence/@{any}

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

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

641 The following exemplar defines the *CreateSequenceResponse* syntax:

```
642 <wsmr:CreateSequenceResponse ...>  
643   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
644   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
645   <wsmr:IncompleteSequenceBehavior>  
646     wsmr:IncompleteSequenceBehaviorType  
647   </wsmr:IncompleteSequenceBehavior> ?  
648   <wsmr:Accept ...>  
649     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
650     ...  
651   </wsmr:Accept> ?  
652   ...  
653 </wsmr:CreateSequenceResponse>
```

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

655 /wsmr:CreateSequenceResponse

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

659 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

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

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

667 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

680 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

693 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

699 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

714 /wsrm:CreateSequenceResponse/{any}

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

717 /wsrm:CreateSequenceResponse/@{any}

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

720 3.5 Closing A Sequence

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

726 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of
727 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept
728 any new messages for the specified Sequence, other than those already accepted at the time the
729 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or
730 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST
731 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`
732 element) header block on any messages associated with the Sequence destined to the RM Source,
733 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM
734 Source.

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

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

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

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

754 The following exemplar defines the `CloseSequence` syntax:

```
755 <wsrm:CloseSequence ...>  
756   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
```

```

757     <wsrm:LastMsgNumber> wsrm:MessageType </wsrm:LastMsgNumber> ?
758     ...
759 </wsrm:CloseSequence>

```

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

761 `/wsrm:CloseSequence`

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

765 `/wsrm:CloseSequence/wsrm:Identifier`

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

769 `/wsrm:CloseSequence/wsrm:LastMsgNumber`

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

773 `/wsrm:CloseSequence/wsrm:Identifier/@{any}`

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

776 `/wsrm:CloseSequence/{any}`

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

779 `/wsrm:CloseSequence/@{any}`

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

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

784 The following exemplar defines the `CloseSequenceResponse` syntax:

```

785 <wsrm:CloseSequenceResponse ...>
786   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
787   ...
788 </wsrm:CloseSequenceResponse>

```

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

790 `/wsrm:CloseSequenceResponse`

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

793 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

797 `/wsrm:CloseSequenceResponse/wsrm:Identifier/@{any}`

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

800 /wsmr:CloseSequenceResponse/{any}

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

803 /wsmr:CloseSequenceResponse/@{any}

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

806 3.6 Sequence Termination

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

814 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM
815 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.
816 The RM Destination can use this information, for example, to implement the behavior indicated by
817 `/wsmr:CreateSequenceResponse/wsmr:IncompleteSequenceBehavior`. The value of the
818 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the
819 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same
820 Sequence.

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

825 The following exemplar defines the `TerminateSequence` syntax:

```
826 <wsmr:TerminateSequence ...>  
827   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
828   <wsmr>LastMsgNumber> wsmr:MessageNumberType </wsmr>LastMsgNumber> ?  
829   ...  
830 </wsmr:TerminateSequence>
```

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

832 /wsmr:TerminateSequence

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

838 This element MAY also be sent by the RM Destination to indicate that it has unilaterally
839 terminated the Sequence. Upon sending this message the RM Destination MUST NOT accept
840 any additional messages (with the exception of the corresponding
841 `TerminateSequenceResponse`) for this Sequence. Upon receipt of a `TerminateSequence`
842 the RM Source MUST NOT send any additional messages (with the exception of the
843 corresponding `TerminateSequenceResponse`) for this Sequence.

844 /wsmr:TerminateSequence/wsmr:Identifier

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

848 /wsmr:TerminateSequence/wsmr:LastMsgNumber

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

852 /wsmr:TerminateSequence/wsmr:Identifier/@{any}

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

855 /wsmr:TerminateSequence/{any}

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

858 /wsmr:TerminateSequence/@{any}

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

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

863 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
864 <wsmr:TerminateSequenceResponse ...>  
865   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
866   ...  
867 </wsmr:TerminateSequenceResponse>
```

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

869 /wsmr:TerminateSequenceResponse

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

873 /wsmr:TerminateSequenceResponse/wsmr:Identifier

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

877 /wsmr:TerminateSequenceResponse/wsmr:Identifier/@{any}

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

880 /wsmr:TerminateSequenceResponse/{any}

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

883 /wsmr:TerminateSequenceResponse/@{any}

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

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

889 3.7 Sequences

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

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

897 A following exemplar defines its syntax:

```
898 <wsm:Sequence ...>  
899   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
900   <wsm:MessageNumber> wsm:MessageNumberType </wsm:MessageNumber>  
901   ...  
902 </wsm:Sequence>
```

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

904 `/wsm:Sequence`

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

911 `/wsm:Sequence/wsm:Identifier`

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

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

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

918 `/wsm:Sequence/wsm:MessageNumber`

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

923 `/wsm:Sequence/{any}`

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

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

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

929 The following example illustrates a Sequence header block.

```
930 <wsrm:Sequence>
931   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
932   <wsrm:MessageNumber>10</wsrm:MessageNumber>
933 </wsrm:Sequence>
```

934 3.8 Request Acknowledgement

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

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

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

949 The following exemplar defines its syntax:

```
950 <wsrm:AckRequested ...>
951   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
952   ...
953 </wsrm:AckRequested>
```

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

955 /wsrm:AckRequested

956 This element requests an Acknowledgement for the identified Sequence.

957 /wsrm:AckRequested/wsrm:Identifier

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

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

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

965 /wsrm:AckRequested/{any}

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

968 /wsrm:AckRequested/@{any}

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

971 3.9 Sequence Acknowledgement

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

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

982 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
983 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
984 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
985 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
986 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
987 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
988 header block for that same Sequence Identifier. When the RM Destination receives an
989 `AckRequested` header, and the `AcksTo` EPR for that Sequence is the WS-Addressing anonymous IRI,
990 the RM Destination SHOULD respond on the protocol binding-specific back-channel provided by the
991 Received message containing the `AckRequested` header block.

992 The following exemplar defines its syntax:

```
993 <wsm:SequenceAcknowledgement ...>  
994   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
995   [ [ [ <wsm:AcknowledgementRange ...  
996         Upper="wsm:MessageNumberType"  
997         Lower="wsm:MessageNumberType"/> +  
998         | <wsm:None/> ]  
999         <wsm:Final/> ? ]  
1000     | <wsm:Nack> wsm:MessageNumberType </wsm:Nack> + ]  
1001  
1002     ...  
1003 </wsm:SequenceAcknowledgement>
```

1004 The following describes the content model of the `SequenceAcknowledgement` header block.

1005 `/wsm:SequenceAcknowledgement`

1006 This element contains the Sequence Acknowledgement information.

1007 `/wsm:SequenceAcknowledgement/wsm:Identifier`

1008 An RM Destination that includes a `SequenceAcknowledgement` header block in a SOAP
1009 envelope MUST include this element in that header block. The RM Destination MUST set the
1010 value of this element to the absolute URI (conformant with RFC3986) that uniquely identifies the
1011 Sequence. The RM Destination MUST NOT include multiple `SequenceAcknowledgement`
1012 header blocks that share the same value for `Identifier` within the same SOAP envelope.

1013 `/wsm:SequenceAcknowledgement/wsm:Identifier/@{any}`

1014 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1015 to the element.

1016 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange`

- 1017 The RM Destination MAY include one or more instances of this element within a
1018 `SequenceAcknowledgement` header block. It contains a range of Sequence message numbers
1019 successfully accepted by the RM Destination. The ranges MUST NOT overlap. The RM
1020 Destination MUST NOT include this element if a sibling `Nack` or `None` element is also present as
1021 a child of `SequenceAcknowledgement`.
- 1022 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper`
1023 The RM Destination MUST set the value of this attribute equal to the message number of the
1024 highest contiguous message in a Sequence range accepted by the RM Destination.
- 1025 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower`
1026 The RM Destination MUST set the value of this attribute equal to the message number of the
1027 lowest contiguous message in a Sequence range accepted by the RM Destination.
- 1028 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}`
1029 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1030 to the element.
- 1031 `/wsrm:SequenceAcknowledgement/wsrm:None`
1032 The RM Destination MUST include this element within a `SequenceAcknowledgement` header
1033 block if the RM Destination has not accepted any messages for the specified Sequence. The RM
1034 Destination MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack`
1035 element is also present as a child of the `SequenceAcknowledgement`.
- 1036 `/wsrm:SequenceAcknowledgement/wsrm:Final`
1037 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1038 block. This element indicates that the RM Destination is not receiving new messages for the
1039 specified Sequence. The RM Source can be assured that the ranges of messages acknowledged
1040 by this `SequenceAcknowledgement` header block will not change in the future. The RM
1041 Destination MUST include this element when the Sequence is closed. The RM Destination MUST
1042 NOT include this element when sending a `Nack`; it can only be used when sending
1043 `AcknowledgementRange` elements or a `None`.
- 1044 `/wsrm:SequenceAcknowledgement/wsrm:Nack`
1045 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1046 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`
1047 representing the `MessageNumber` of an unreceived message in a Sequence. The RM Destination
1048 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is
1049 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM
1050 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT
1051 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously
1052 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a
1053 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been
1054 acknowledged within an `AcknowledgementRange`.
- 1055 `/wsrm:SequenceAcknowledgement/{any}`
1056 This is an extensibility mechanism to allow different (extensible) types of information, based on a
1057 schema, to be passed.
- 1058 `/wsrm:SequenceAcknowledgement/@{any}`
1059 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1060 to the element.

1061 The following examples illustrate `SequenceAcknowledgement` elements:

- 1062 • Message numbers 1..10 inclusive in a Sequence have been accepted by the RM Destination.

```
1063 <wsrm:SequenceAcknowledgement>  
1064   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1065   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
1066 </wsrm:SequenceAcknowledgement>
```

- 1067 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM
1068 Destination, messages 3 and 7 have not been accepted.

```
1069 <wsrm:SequenceAcknowledgement>  
1070   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1071   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1072   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1073   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1074 </wsrm:SequenceAcknowledgement>
```

- 1075 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1076 <wsrm:SequenceAcknowledgement>  
1077   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1078   <wsrm:Nack>3</wsrm:Nack>  
1079 </wsrm:SequenceAcknowledgement>
```

1080 4 Faults

1081 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. `Create`
1082 `Sequence Refused` is a possible fault reply for this operation. `Unknown Sequence` is a fault generated by
1083 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences
1084 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM
1085 on a Received message that did not use the protocol. All other faults in this section relate to known
1086 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those faults.
1087 If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

1088 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault
1089 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1090 `http://docs.oasis-open.org/ws-rx/wsr/200702/fault`

1091 The faults defined in this section are generated if the condition stated in the preamble is met. Fault
1092 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1093 The definitions of faults use the following properties:

1094 [Code] The fault code.

1095 [Subcode] The fault subcode.

1096 [Reason] The English language reason element.

1097 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail
1098 element is defined for a fault, implementations MUST include the elements in the order that they are
1099 specified.

1100 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
1101 "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

1102 The properties above bind to a SOAP 1.2 fault as follows:

```
1103 <S:Envelope>  
1104   <S:Header>  
1105     <wsa:Action>  
1106       http://docs.oasis-open.org/ws-rx/wsr/200702/fault  
1107     </wsa:Action>  
1108     <!-- Headers elided for brevity. -->  
1109   </S:Header>  
1110   <S:Body>  
1111     <S:Fault>  
1112       <S:Code>  
1113         <S:Value> [Code] </S:Value>  
1114         <S:Subcode>  
1115           <S:Value> [Subcode] </S:Value>  
1116         </S:Subcode>  
1117       </S:Code>  
1118       <S:Reason>  
1119         <S:Text xml:lang="en"> [Reason] </S:Text>  
1120       </S:Reason>  
1121     <S:Detail>
```

```
1122     [Detail]
1123     ...
1124     </S:Detail>
1125     </S:Fault>
1126     </S:Body>
1127     </S:Envelope>
```

1128 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM
1129 header block:

```
1130 <S11:Envelope>
1131   <S11:Header>
1132     <wsrm:SequenceFault>
1133       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1134       <wsrm:Detail> [Detail] </wsrm:Detail>
1135       ...
1136     </wsrm:SequenceFault>
1137     <!-- Headers elided for brevity. -->
1138   </S11:Header>
1139   <S11:Body>
1140     <S11:Fault>
1141       <faultcode> [Code] </faultcode>
1142       <faultstring> [Reason] </faultstring>
1143     </S11:Fault>
1144   </S11:Body>
1145 </S11:Envelope>
```

1146 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
1147 CreateSequence request message:

```
1148 <S11:Envelope>
1149   <S11:Body>
1150     <S11:Fault>
1151       <faultcode> [Subcode] </faultcode>
1152       <faultstring> [Reason] </faultstring>
1153     </S11:Fault>
1154   </S11:Body>
1155 </S11:Envelope>
```

1156 4.1 SequenceFault Element

1157 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the
1158 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging
1159 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.
1160 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the
1161 SOAP 1.2 binding.

1162 The following exemplar defines its syntax:

```
1163 <wsrm:SequenceFault ...>
1164   <wsrm:FaultCode> wsrm:FaultCode </wsrm:FaultCode>
1165   <wsrm:Detail> ... </wsrm:Detail> ?
1166   ...
1167 </wsrm:SequenceFault>
```

1168 The following describes the content model of the `SequenceFault` element.

1169 /wsrm:SequenceFault

1170 This is the element containing Sequence fault information for WS-ReliableMessaging

1171 /wsrm:SequenceFault/wsrm:FaultCode

- 1172 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this
 1173 element to a qualified name from the set of faults [Subcodes] defined below.
- 1174 `/wsrm:SequenceFault/wsrm:Detail`
 1175 This element, if present, carries application specific error information related to the fault being
 1176 described.
- 1177 `/wsrm:SequenceFault/wsrm:Detail/{any}`
 1178 The application specific error information related to the fault being described.
- 1179 `/wsrm:SequenceFault/wsrm:Detail/@{any}`
 1180 The application specific error information related to the fault being described.
- 1181 `/wsrm:SequenceFault/{any}`
 1182 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 1183 schema, to be passed.
- 1184 `/wsrm:SequenceFault/@{any}`
 1185 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 1186 to the element.

1187 4.2 Sequence Terminated

1188 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 1189 Endpoint of this decision.

1190 Properties:

1191 [Code] Sender or Receiver

1192 [Subcode] `wrm:SequenceTerminated`

1193 [Reason] The Sequence has been terminated due to an unrecoverable error.

1194 [Detail]

1195 `<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.

1196 4.3 Unknown Sequence

1197 Properties:

1198 [Code] Sender

1199 [Subcode] `wrm:UnknownSequence`

1200 [Reason] The value of `wsrc:Identifier` is not a known Sequence identifier.

1201 [Detail]

1202 `<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.

1203 4.4 Invalid Acknowledgement

1204 An example of when this fault is generated is when a message is Received by the RM Source containing
1205 a `SequenceAcknowledgement` covering messages that have not been sent.

1206 [Code] Sender

1207 [Subcode] `wsrc:InvalidAcknowledgement`

1208 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1209 [Detail]

1210 `<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.

1211 4.5 Message Number Rollover

1212 If the condition listed below is reached, the RM Destination MUST generate this fault.

1213 Properties:

1214 [Code] Sender

1215 [Subcode] wsrn:MessageNumberRollover

1216 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1217 [Detail]

1218 `<wsrn:Identifier ...> xs:anyURI </wsrn:Identifier>`

1219 `<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.

1220 4.6 Create Sequence Refused

1221 Properties:

1222 [Code] Sender or Receiver

1223 [Subcode] wsrn:CreateSequenceRefused

1224 [Reason] The Create Sequence request has been refused by the RM Destination.

1225 [Detail]

1226 `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.

1227 4.7 Sequence Closed

1228 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1229 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1230 is closed.

1231 Properties:

1232 [Code] Sender

1233 [Subcode] wsrn:SequenceClosed

1234 [Reason] The Sequence is closed and cannot accept new messages.

1235 [Detail]

1236 `<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.

1237 **4.8 WSRM Required**

1238 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1239 message that did not use this protocol.

1240 Properties:

1241 [Code] Sender

1242 [Subcode] wsrn:WSRMRequired

1243 [Reason] The RM Destination requires the use of WSRM.

1244 [Detail]

1245 `xs:any`

1246 5 Security Threats and Countermeasures

1247 This specification considers two sets of security requirements, those of the applications that use the WS-
1248 RM protocol and those of the protocol itself.

1249 This specification makes no assumptions about the security requirements of the applications that use WS-
1250 RM. However, once those requirements have been satisfied within a given operational context, the
1251 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1252 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1253 There are many other security concerns that one may need to consider when implementing or using this
1254 protocol. The material below should not be considered as a "check list". Implementers and users of this
1255 protocol are urged to perform a security analysis to determine their particular threat profile and the
1256 appropriate responses to those threats.

1257 Implementers are also advised that there is a core tension between security and reliable messaging that
1258 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1259 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1260 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1261 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1262 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1263 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1264 avoid and prevent this condition.

1265 5.1 Threats and Countermeasures

1266 The primary security requirement of this protocol is to protect the specified semantics and protocol
1267 invariants against various threats. The following sections describe several threats to the integrity and
1268 operation of this protocol and provide some general outlines of countermeasures to those threats.
1269 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1270 to all operational contexts.

1271 5.1.1 Integrity Threats

1272 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1273 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1274 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1275 to its intended message represents a threat to the WS-RM protocol.

1276 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM
1277 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1278 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be
1279 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1280 5.1.1.1 Countermeasures

1281 Integrity threats are generally countered via the use of digital signatures some level of the communication
1282 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1283 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers
1284 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in
1285 which they occur, implementations MUST allow for signatures that cover only these headers.

1286 5.1.2 Resource Consumption Threats

1287 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1288 implement that RM Destination. These resources can include network connections, database tables,
1289 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1290 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1291 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1292 message Delivery and use this Sequence to send a stream of very large messages to that service, making
1293 sure to omit message number “1” from that stream.

1294 5.1.2.1 Countermeasures

1295 There are a number of countermeasures against the described resource consumption threats. The
1296 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1297 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1298 some cases, allows the identity of any attackers to be determined.

1299 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and
1300 authenticate the RM Source that issued the `CreateSequence` message.

1301 5.1.3 Sequence Spoofing Threats

1302 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1303 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1304 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1305 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the
1306 current `MessageNumber` for their target Sequence.

1307 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP
1308 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence
1309 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to
1310 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a
1311 message that it sends to the `AcksTo` EPR of an RM Source.

1312 5.1.3.1 Sequence Hijacking

1313 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject
1314 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1315 messages.

1316 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a
1317 Sequence may be bound to some form of a security session in order to counter the threats described in
1318 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1319 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1320 established by the security infrastructure to make such determinations. Failure to observe this rule
1321 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1322 the ability to authenticate its peers even though the necessary security processing has taken place.

1323 5.1.3.2 Countermeasures

1324 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by
1325 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM
1326 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination
1327 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

1328 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it
1329 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the
1330 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it
1331 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the
1332 referenced Sequence.

1333 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and
1334 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify
1335 its Sequence peer it MUST be able to identify and authenticate the entity that sent the
1336 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1337 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that
1338 message and, if necessary, correlate this identity with the Sequence peer identity established at
1339 Sequence creation time.

1340 **5.2 Security Solutions and Technologies**

1341 The security threats described in the previous sections are neither new nor unique. The solutions that
1342 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1343 section maps the facilities provided by common web services security solutions against countermeasures
1344 described in the previous sections.

1345 Before continuing this discussion, however, some examination of the underlying requirements of the
1346 previously described countermeasures is necessary. Specifically it should be noted that the technique
1347 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1348 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization
1349 check against this authenticated identity and determines if the RM Source is permitted to create
1350 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1351 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1352 discussion of such facilities is considered to be beyond the scope of this specification.

1353 **5.2.1 Transport Layer Security**

1354 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1355 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1356 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1357 The description provided here is general in nature and is not intended to serve as a complete definition on
1358 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1359 choice of features as well as the manner in which they will be used. The mechanisms described in the
1360 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1361 requirements and constraints of the use of SSL/TLS.

1362 **5.2.1.1 Model**

1363 The basic model for using SSL/TLS is as follows:

- 1364 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1365 ~~4.2.~~ 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1366 Destination.
- 1367 ~~4.3.~~ 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1368 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1369 synchronous `CreateSequenceResponse` using the session established in (1).

1370 4.4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1371 any and all messages or faults that refer to that Sequence.

1372 4.5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1373 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1374 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1375 5.2.1.2 Countermeasure Implementation

1376 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1377 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the
1378 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1379 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1380 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1381 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification
1382 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1383 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1384 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

1385 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1386 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1387 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1388 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1389 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1390 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1391 Acknowledgement) using BasicAuth.

1392 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1393 connection authenticates itself to the party accepting the connection using an X.509 certificate
1394 that is exchanged during the SSL/TLS handshake.

1395 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1396 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1397 Source is authorized to create a Sequence with the RM Destination.

1398 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1399 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1400 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1401 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1402 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1403 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1404 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1405 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1406 to protect that Sequence.

1407 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1408 countermeasures (such as associating specific authentication information with a Sequence) although such
1409 methods are not covered by this document.

1410 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1411 session) are outside the scope of this specification.

1412 5.2.2 SOAP Message Security

1413 The mechanisms described in WS-Security may be used in various ways to implement the
1414 countermeasures described in the previous sections. This specification advocates using the protocol
1415 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1416 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1417 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1418 The description provided here is general in nature and is not intended to serve as a complete definition on
1419 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1420 need to agree on the choice of features as well as the manner in which they will be used. The
1421 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1422 describe the requirements and constraints of the use of WS-SecureConversation.

1423 **5.2.2.1 Model**

1424 The basic model for using WS-SecureConversation is as follows:

1425 1 The RM Source and the RM Destination create a WS-SecureConversation security context. This
1426 may involve the participation of third parties such as a security token service. The tokens
1427 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service
1428 tickets).

1429 ~~1-2~~ During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1430 context that will be used to protect the Sequence. This is done so that, in cases where the
1431 `CreateSequence` message is signed by more than one security context, the RM Source can
1432 indicate which security context should be used to protect the newly created Sequence.

1433 ~~1-3~~ For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1434 associated with the security context to sign (as defined by WS-Security) at least the body and
1435 any relevant WS-RM-defined headers of any and all messages or faults that refer to that
1436 Sequence.

1437 **5.2.2.2 Countermeasure Implementation**

1438 Without relying upon any authentication information, the per-message signatures provide the necessary
1439 integrity qualities to counter the threats described in section 5.1.1.

1440 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1441 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1442 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1443 create a Sequence with the RM Destination.

1444 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1445 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1446 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1447 context rather than on any authentication claims that may have been established during security context
1448 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1449 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1450 document.

1451 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits
1452 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the
1453 association between a Sequence and its protecting security context cannot always be established
1454 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1455 `CreateSequenceResponse` messages may be signed by more than one security context.

1456 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1457 amending or renewing contexts) are outside the scope of this specification.

1458 6 Securing Sequences

1459 As noted in section 5, the RM Source and RM Destination should be able to protect their shared
1460 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1461 achieving this objective depending upon the underlying security infrastructure.

1462 6.1 Securing Sequences Using WS-Security

1463 One mechanism for protecting a Sequence is to include a security token using a
1464 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1465 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1466 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1467 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1468 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1469 there may be more than one token on a `CreateSequence` message or inferred from the communication
1470 context (e.g. transport protection).

1471 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1472 if the token being referenced supports such mechanism.

1473 The following exemplar defines the `CreateSequence` syntax when extended to include a
1474 `wsse:SecurityTokenReference`:

```
1475 <wsrm:CreateSequence ...>  
1476   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1477   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1478   <wsrm:Offer ...>  
1479     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1480     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1481     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1482     <wsrm:IncompleteSequenceBehavior>  
1483       wsrm:IncompleteSequenceBehaviorType  
1484     </wsrm:IncompleteSequenceBehavior> ?  
1485     ...  
1486   </wsrm:Offer> ?  
1487   ...  
1488   <wsse:SecurityTokenReference>  
1489     ...  
1490   </wsse:SecurityTokenReference> ?  
1491   ...  
1492 </wsrm:CreateSequence>
```

1493 The following describes the content model of the additional `CreateSequence` elements.

1494 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1495 This element uses the extensibility mechanism defined for the `CreateSequence` element
1496 (defined in section 3.4) to communicate an explicit reference to the security token, using a
1497 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and
1498 Destination MUST use to authorize messages for the created (and, if present, the offered)
1499 Sequence(s). All subsequent messages related to the created (and, if present, the offered)
1500 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token
1501 (e.g., by using or deriving from a private or secret key).

1502 When a RM Source transmits a `CreateSequence` that has been extended to include a
1503 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1504 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1505 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1506 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1507 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1508 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1509 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1510 in WS-Security still applies.

1511 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1512 <wsrm:UsesSequenceSTR ... />
```

1513 The following describes the content model of the `UsesSequenceSTR` header block.

1514 `/wsrm:UsesSequenceSTR`

1515 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that
1516 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`
1517 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly
1518 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus
1519 aborting the requested Sequence creation.

1520 The following is an example of a `CreateSequence` message using the

1521 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1522 <soap:Envelope ...>  
1523   <soap:Header>  
1524     ...  
1525     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1526     ...  
1527   </soap:Header>  
1528   <soap:Body>  
1529     <wsrm:CreateSequence>  
1530       <wsrm:AcksTo>  
1531         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1532       </wsrm:AcksTo>  
1533       <wsse:SecurityTokenReference>  
1534         ...  
1535       </wsse:SecurityTokenReference>  
1536     </wsrm:CreateSequence>  
1537   </soap:Body>  
1538 </soap:Envelope>
```

1539 6.2 Securing Sequences Using SSL/TLS

1540 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1541 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1542 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1543 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1544 SOAP header block within the `CreateSequence` message.

1545 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1546 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1547 The following describes the content model of the `UsesSequenceSSL` header block.

1548 `/wsrm:UsesSequenceSSL`

1549 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`
1550 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1551 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM
1552 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.
1553 The receiving RM Destination MUST understand and correctly implement the functionality
1554 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the
1555 requested Sequence creation.

1556 Note that the inclusion of the above header by the RM Source implies that all Sequence-related
1557 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1558 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1559 `CreateSequenceResponse` message.

1560 ~~Appendix G.~~ Appendix A. Schema

1561 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1562 Schema Part2] is located at:

1563 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1564 The following copy is provided for reference.

```
1565 <?xml version="1.0" encoding="UTF-8"?>
1566 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1567      OASIS trademark, IPR and other policies apply. -->
1568 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1569   xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1570   open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1571   rx/wsrn/200702" elementFormDefault="qualified"
1572   attributeFormDefault="unqualified">
1573   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1574   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1575   <!-- Protocol Elements -->
1576   <xs:complexType name="SequenceType">
1577     <xs:sequence>
1578       <xs:element ref="wsrm:Identifier"/>
1579       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1580       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1581   maxOccurs="unbounded"/>
1582     </xs:sequence>
1583     <xs:anyAttribute namespace="##other" processContents="lax"/>
1584   </xs:complexType>
1585   <xs:element name="Sequence" type="wsrm:SequenceType"/>
1586   <xs:element name="SequenceAcknowledgement">
1587     <xs:complexType>
1588       <xs:sequence>
1589         <xs:element ref="wsrm:Identifier"/>
1590         <xs:choice>
1591           <xs:sequence>
1592             <xs:choice>
1593               <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1594                 <xs:complexType>
1595                   <xs:sequence/>
1596                   <xs:attribute name="Upper" type="xs:unsignedLong"
1597   use="required"/>
1598                   <xs:attribute name="Lower" type="xs:unsignedLong"
1599   use="required"/>
1600                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1601               </xs:complexType>
1602             </xs:choice>
1603             <xs:element name="None">
1604               <xs:complexType>
1605                 <xs:sequence/>
1606               </xs:complexType>
1607             </xs:element>
1608           </xs:choice>
1609           <xs:element name="Final" minOccurs="0">
1610             <xs:complexType>
1611               <xs:sequence/>
1612             </xs:complexType>
1613           </xs:element>
1614         </xs:sequence>
1615       <xs:element name="Nack" type="xs:unsignedLong"
```

```

1616 maxOccurs="unbounded"/>
1617     </xs:choice>
1618     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1619 maxOccurs="unbounded"/>
1620     </xs:sequence>
1621     <xs:anyAttribute namespace="##other" processContents="lax"/>
1622 </xs:complexType>
1623 </xs:element>
1624 <xs:complexType name="AckRequestedType">
1625     <xs:sequence>
1626         <xs:element ref="wsrm:Identifier"/>
1627         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1628 maxOccurs="unbounded"/>
1629     </xs:sequence>
1630     <xs:anyAttribute namespace="##other" processContents="lax"/>
1631 </xs:complexType>
1632 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1633 <xs:element name="Identifier">
1634     <xs:complexType>
1635         <xs:annotation>
1636             <xs:documentation>
1637                 This type is for elements whose [children] is an anyURI and can have
1638 arbitrary attributes.
1639             </xs:documentation>
1640         </xs:annotation>
1641         <xs:simpleContent>
1642             <xs:extension base="xs:anyURI">
1643                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1644             </xs:extension>
1645         </xs:simpleContent>
1646     </xs:complexType>
1647 </xs:element>
1648 <xs:element name="Address">
1649     <xs:complexType>
1650         <xs:simpleContent>
1651             <xs:extension base="xs:anyURI">
1652                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1653             </xs:extension>
1654         </xs:simpleContent>
1655     </xs:complexType>
1656 </xs:element>
1657 <xs:simpleType name="MessageNumberType">
1658     <xs:restriction base="xs:unsignedLong">
1659         <xs:minInclusive value="1"/>
1660         <xs:maxInclusive value="9223372036854775807"/>
1661     </xs:restriction>
1662 </xs:simpleType>
1663 <!-- Fault Container and Codes -->
1664 <xs:simpleType name="FaultCodes">
1665     <xs:restriction base="xs:QName">
1666         <xs:enumeration value="wsrm:SequenceTerminated"/>
1667         <xs:enumeration value="wsrm:UnknownSequence"/>
1668         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1669         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1670         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1671         <xs:enumeration value="wsrm:SequenceClosed"/>
1672         <xs:enumeration value="wsrm:WSRMRequired"/>
1673     </xs:restriction>
1674 </xs:simpleType>
1675 <xs:complexType name="SequenceFaultType">
1676     <xs:sequence>
1677         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1678         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1679         <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1680 maxOccurs="unbounded"/>
1681 </xs:sequence>
1682 <xs:anyAttribute namespace="##other" processContents="lax"/>
1683 </xs:complexType>
1684 <xs:complexType name="DetailType">
1685 <xs:sequence>
1686 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1687 maxOccurs="unbounded"/>
1688 </xs:sequence>
1689 <xs:anyAttribute namespace="##other" processContents="lax"/>
1690 </xs:complexType>
1691 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1692 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1693 <xs:element name="CreateSequenceResponse"
1694 type="wsrm:CreateSequenceResponseType"/>
1695 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1696 <xs:element name="CloseSequenceResponse"
1697 type="wsrm:CloseSequenceResponseType"/>
1698 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1699 <xs:element name="TerminateSequenceResponse"
1700 type="wsrm:TerminateSequenceResponseType"/>
1701 <xs:complexType name="CreateSequenceType">
1702 <xs:sequence>
1703 <xs:element ref="wsrm:AcksTo"/>
1704 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1705 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1706 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1707 maxOccurs="unbounded">
1708 <xs:annotation>
1709 <xs:documentation>
1710 It is the authors intent that this extensibility be used to
1711 transfer a Security Token Reference as defined in WS-Security.
1712 </xs:documentation>
1713 </xs:annotation>
1714 </xs:any>
1715 </xs:sequence>
1716 <xs:anyAttribute namespace="##other" processContents="lax"/>
1717 </xs:complexType>
1718 <xs:complexType name="CreateSequenceResponseType">
1719 <xs:sequence>
1720 <xs:element ref="wsrm:Identifier"/>
1721 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1722 <xs:element name="IncompleteSequenceBehavior"
1723 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1724 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1725 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1726 maxOccurs="unbounded"/>
1727 </xs:sequence>
1728 <xs:anyAttribute namespace="##other" processContents="lax"/>
1729 </xs:complexType>
1730 <xs:complexType name="CloseSequenceType">
1731 <xs:sequence>
1732 <xs:element ref="wsrm:Identifier"/>
1733 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1734 minOccurs="0"/>
1735 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1736 maxOccurs="unbounded"/>
1737 </xs:sequence>
1738 <xs:anyAttribute namespace="##other" processContents="lax"/>
1739 </xs:complexType>
1740 <xs:complexType name="CloseSequenceResponseType">
1741 <xs:sequence>
1742 <xs:element ref="wsrm:Identifier"/>
1743 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1744 maxOccurs="unbounded"/>
1745 </xs:sequence>
1746 <xs:anyAttribute namespace="##other" processContents="lax"/>
1747 </xs:complexType>
1748 <xs:complexType name="TerminateSequenceType">
1749 <xs:sequence>
1750 <xs:element ref="wsrm:Identifier"/>
1751 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1752 minOccurs="0"/>
1753 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1754 maxOccurs="unbounded"/>
1755 </xs:sequence>
1756 <xs:anyAttribute namespace="##other" processContents="lax"/>
1757 </xs:complexType>
1758 <xs:complexType name="TerminateSequenceResponseType">
1759 <xs:sequence>
1760 <xs:element ref="wsrm:Identifier"/>
1761 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1762 maxOccurs="unbounded"/>
1763 </xs:sequence>
1764 <xs:anyAttribute namespace="##other" processContents="lax"/>
1765 </xs:complexType>
1766 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1767 <xs:complexType name="OfferType">
1768 <xs:sequence>
1769 <xs:element ref="wsrm:Identifier"/>
1770 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1771 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1772 <xs:element name="IncompleteSequenceBehavior"
1773 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1774 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1775 maxOccurs="unbounded"/>
1776 </xs:sequence>
1777 <xs:anyAttribute namespace="##other" processContents="lax"/>
1778 </xs:complexType>
1779 <xs:complexType name="AcceptType">
1780 <xs:sequence>
1781 <xs:element ref="wsrm:AcksTo"/>
1782 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1783 maxOccurs="unbounded"/>
1784 </xs:sequence>
1785 <xs:anyAttribute namespace="##other" processContents="lax"/>
1786 </xs:complexType>
1787 <xs:element name="Expires">
1788 <xs:complexType>
1789 <xs:simpleContent>
1790 <xs:extension base="xs:duration">
1791 <xs:anyAttribute namespace="##other" processContents="lax"/>
1792 </xs:extension>
1793 </xs:simpleContent>
1794 </xs:complexType>
1795 </xs:element>
1796 <xs:simpleType name="IncompleteSequenceBehaviorType">
1797 <xs:restriction base="xs:string">
1798 <xs:enumeration value="DiscardEntireSequence"/>
1799 <xs:enumeration value="DiscardFollowingFirstGap"/>
1800 <xs:enumeration value="NoDiscard"/>
1801 </xs:restriction>
1802 </xs:simpleType>
1803 <xs:element name="UsesSequenceSTR">
1804 <xs:complexType>
1805 <xs:sequence/>
1806 <xs:anyAttribute namespace="##other" processContents="lax"/>

```

```
1807     </xs:complexType>
1808 </xs:element>
1809 <xs:element name="UsesSequenceSSL">
1810   <xs:complexType>
1811     <xs:sequence/>
1812     <xs:anyAttribute namespace="##other" processContents="lax"/>
1813   </xs:complexType>
1814 </xs:element>
1815 <xs:element name="UnsupportedElement">
1816   <xs:simpleType>
1817     <xs:restriction base="xs:QName"/>
1818   </xs:simpleType>
1819 </xs:element>
1820 </xs:schema>
```

1821 Appendix H. Appendix B. WSDL

1822 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1823 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1824 present in exchanges with that endpoint.

1825 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1826 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [[WS-RM Policy](#)]
1827 for a higher-level mechanism to indicate that WS-RM is engaged.

1828 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1829 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

1830 The following non-normative copy is provided for reference.

```
1831 <?xml version="1.0" encoding="utf-8"?>
1832 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1833 OASIS trademark, IPR and other policies apply. -->
1834 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1835 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1836 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1837 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1838 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1839 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsdl"
1840 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsdl">
1841
1842   <wsdl:types>
1843     <xs:schema
1844       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1845       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1846       200702.xsd"/>
1847     </xs:schema>
1848   </wsdl:types>
1849
1850   <wsdl:message name="CreateSequence">
1851     <wsdl:part name="create" element="rm:CreateSequence"/>
1852   </wsdl:message>
1853   <wsdl:message name="CreateSequenceResponse">
1854     <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1855   </wsdl:message>
1856   <wsdl:message name="CloseSequence">
1857     <wsdl:part name="close" element="rm:CloseSequence"/>
1858   </wsdl:message>
1859   <wsdl:message name="CloseSequenceResponse">
1860     <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1861   </wsdl:message>
1862   <wsdl:message name="TerminateSequence">
1863     <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1864   </wsdl:message>
1865   <wsdl:message name="TerminateSequenceResponse">
1866     <wsdl:part name="terminateResponse"
1867     element="rm:TerminateSequenceResponse"/>
1868   </wsdl:message>
1869
1870   <wsdl:portType name="SequenceAbstractPortType">
1871     <wsdl:operation name="CreateSequence">
1872       <wsdl:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1873       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1874       <wsdl:output message="tns:CreateSequenceResponse"
```

```
1875 wsam:Action="http://docs.oasis-open.org/ws-
1876 rx/wsrn/200702/CreateSequenceResponse"/>
1877 </wsdl:operation>
1878 <wsdl:operation name="CloseSequence">
1879 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1880 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1881 <wsdl:output message="tns:CloseSequenceResponse"
1882 wsam:Action="http://docs.oasis-open.org/ws-
1883 rx/wsrn/200702/CloseSequenceResponse"/>
1884 </wsdl:operation>
1885 <wsdl:operation name="TerminateSequence">
1886 <wsdl:input message="tns:TerminateSequence"
1887 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1888 <wsdl:output message="tns:TerminateSequenceResponse"
1889 wsam:Action="http://docs.oasis-open.org/ws-
1890 rx/wsrn/200702/TerminateSequenceResponse"/>
1891 </wsdl:operation>
1892 </wsdl:portType>
1893
1894 </wsdl:definitions>
```

1895 **Appendix I. Appendix C. Message Examples**

1896 **Appendix A.0 Appendix C.1 Create Sequence**

1897 **Create Sequence**

```
1898 <?xml version="1.0" encoding="UTF-8"?>
1899 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1900 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1901 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1902   <S:Header>
1903     <wsa:MessageID>
1904       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1905     </wsa:MessageID>
1906     <wsa:To>http://example.com/serviceB/123</wsa:To>
1907     <wsa:Action>http://docs.oasis-open.org/ws-
1908 rx/wsmr/200702/CreateSequence</wsa:Action>
1909     <wsa:ReplyTo>
1910       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1911     </wsa:ReplyTo>
1912   </S:Header>
1913   <S:Body>
1914     <wsmr:CreateSequence>
1915       <wsmr:AcksTo>
1916         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1917       </wsmr:AcksTo>
1918     </wsmr:CreateSequence>
1919   </S:Body>
1920 </S:Envelope>
```

1921 **Create Sequence Response**

```
1922 <?xml version="1.0" encoding="UTF-8"?>
1923 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1924 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1925 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1926   <S:Header>
1927     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1928     <wsa:RelatesTo>
1929       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1930     </wsa:RelatesTo>
1931     <wsa:Action>
1932       http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1933     </wsa:Action>
1934   </S:Header>
1935   <S:Body>
1936     <wsmr:CreateSequenceResponse>
1937       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1938     </wsmr:CreateSequenceResponse>
1939   </S:Body>
1940 </S:Envelope>
```

1941 **Appendix A.0 Appendix C.2 Initial Transmission**

1942 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1943 figure. The three messages have the following headers; the third message is identified as the last
1944 message in the Sequence:

1945 Message 1

```
1946 <?xml version="1.0" encoding="UTF-8"?>
1947 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1948 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1949 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1950   <S:Header>
1951     <wsa:MessageID>
1952       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1953     </wsa:MessageID>
1954     <wsa:To>http://example.com/serviceB/123</wsa:To>
1955     <wsa:From>
1956       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1957     </wsa:From>
1958     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1959     <wsmr:Sequence>
1960       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1961       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1962     </wsmr:Sequence>
1963   </S:Header>
1964   <S:Body>
1965     <!-- Some Application Data -->
1966   </S:Body>
1967 </S:Envelope>
```

1968 Message 2

```
1969 <?xml version="1.0" encoding="UTF-8"?>
1970 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1971 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1972 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1973   <S:Header>
1974     <wsa:MessageID>
1975       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1976     </wsa:MessageID>
1977     <wsa:To>http://example.com/serviceB/123</wsa:To>
1978     <wsa:From>
1979       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1980     </wsa:From>
1981     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1982     <wsmr:Sequence>
1983       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1984       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1985     </wsmr:Sequence>
1986   </S:Header>
1987   <S:Body>
1988     <!-- Some Application Data -->
1989   </S:Body>
1990 </S:Envelope>
```

1991 Message 3

```
1992 <?xml version="1.0" encoding="UTF-8"?>
1993 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1994 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1995 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1996   <S:Header>
1997     <wsa:MessageID>
1998       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1999     </wsa:MessageID>
2000     <wsa:To>http://example.com/serviceB/123</wsa:To>
2001     <wsa:From>
2002       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2003     </wsa:From>
```

```

2004 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2005 <wsrm:Sequence>
2006 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2007 <wsrm:MessageNumber>3</wsrm:MessageNumber>
2008 </wsrm:Sequence>
2009 <wsrm:AckRequested>
2010 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2011 </wsrm:AckRequested>
2012 </S:Header>
2013 <S:Body>
2014 <!-- Some Application Data -->
2015 </S:Body>
2016 </S:Envelope>

```

2017 **Appendix A.0 Appendix C.3 First Acknowledgement**

2018 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
2019 responds with an Acknowledgement for messages 1 and 3:

```

2020 <?xml version="1.0" encoding="UTF-8"?>
2021 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2022 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2023 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2024 <S:Header>
2025 <wsa:MessageID>
2026 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
2027 </wsa:MessageID>
2028 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2029 <wsa:From>
2030 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2031 </wsa:From>
2032 <wsa:Action>
2033 http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
2034 </wsa:Action>
2035 <wsrm:SequenceAcknowledgement>
2036 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2037 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2038 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2039 </wsrm:SequenceAcknowledgement>
2040 </S:Header>
2041 <S:Body/>
2042 </S:Envelope>

```

2043 **Appendix A.0 Appendix C.4 Retransmission**

2044 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2045 requests an Acknowledgement:

```

2046 <?xml version="1.0" encoding="UTF-8"?>
2047 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2048 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2049 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2050 <S:Header>
2051 <wsa:MessageID>
2052 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2053 </wsa:MessageID>
2054 <wsa:To>http://example.com/serviceB/123</wsa:To>
2055 <wsa:From>
2056 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2057 </wsa:From>
2058 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>

```

```

2059 <wsm:Sequence>
2060 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2061 <wsm:MessageNumber>2</wsm:MessageNumber>
2062 </wsm:Sequence>
2063 <wsm:AckRequested>
2064 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2065 </wsm:AckRequested>
2066 </S:Header>
2067 <S:Body>
2068 <!-- Some Application Data -->
2069 </S:Body>
2070 </S:Envelope>

```

2071 **Appendix A.0 Appendix C.5 Termination**

2072 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2073 be terminated:

```

2074 <?xml version="1.0" encoding="UTF-8"?>
2075 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2076 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2077 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2078 <S:Header>
2079 <wsa:MessageID>
2080 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2081 </wsa:MessageID>
2082 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2083 <wsa:From>
2084 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2085 </wsa:From>
2086 <wsa:Action>
2087 http://docs.oasis-open.org/ws-rx/wsm/200702/SequenceAcknowledgement
2088 </wsa:Action>
2089 <wsm:SequenceAcknowledgement>
2090 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2091 <wsm:AcknowledgementRange Upper="3" Lower="1"/>
2092 </wsm:SequenceAcknowledgement>
2093 </S:Header>
2094 <S:Body/>
2095 </S:Envelope>

```

2096 **Terminate Sequence**

```

2097 <?xml version="1.0" encoding="UTF-8"?>
2098 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2099 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2100 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2101 <S:Header>
2102 <wsa:MessageID>
2103 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2104 </wsa:MessageID>
2105 <wsa:To>http://example.com/serviceB/123</wsa:To>
2106 <wsa:Action>
2107 http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequence
2108 </wsa:Action>
2109 <wsa:From>
2110 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2111 </wsa:From>
2112 </S:Header>
2113 <S:Body>
2114 <wsm:TerminateSequence>
2115 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2116 <wsm:LastMsgNumber> 3 </wsm:LastMsgNumber>

```

```
2117     </wsrm:TerminateSequence>
2118 </S:Body>
2119 </S:Envelope>
```

2120 Terminate Sequence Response

```
2121 <?xml version="1.0" encoding="UTF-8"?>
2122 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2123 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
2124 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2125   <S:Header>
2126     <wsa:MessageID>
2127       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2128     </wsa:MessageID>
2129     <wsa:To>http://example.com/serviceA/789</wsa:To>
2130     <wsa:Action>
2131       http://docs.oasis-open.org/ws-rx/wsmr/200702/TerminateSequenceResponse
2132     </wsa:Action>
2133     <wsa:RelatesTo>
2134       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2135     </wsa:RelatesTo>
2136     <wsa:From>
2137       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2138     </wsa:From>
2139   </S:Header>
2140   <S:Body>
2141     <wsmr:TerminateSequenceResponse>
2142       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
2143     </wsmr:TerminateSequenceResponse>
2144   </S:Body>
2145 </S:Envelope>
```

2146 **Appendix J. Appendix D. State Tables**

2147 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2148 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

2149 Legend:

2150 The first column of these tables contains the motivating event and has the following format:

Event 2151
<i>Event name</i> [source]
{ref}

2152 Where:

- 2153 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2154 described by the specification.
- 2155 • [source]: indicates the source of the event; one of:
 - 2156 ○ [msg] a Received message
 - 2157 ○ [int]: an internal event such as the firing of a timer
 - 2158 ○ [app]: the application
 - 2159 ○ [unspec]: the source is unspecified

2160 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}

2161 Where:

- 2162 • action to take: indicates that the state machine performs the following action. Actions surrounded
2163 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2164 "Transmit"
- 2165 • [next state]: indicates the state to which the state machine will advance upon the performance of
2166 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2167 • {ref} is a reference to the document section describing the behavior in this cell

2168 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2169 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2170 described in this specification and does not indicate normal protocol operation. Implementations MAY
2171 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2172 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2173 combinations.

2174 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}

2175 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}	

2176

Appendix K. ~~Appendix E.~~ Acknowledgments

2177 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2178 authors:

2179	Ruslan Bilursets, BEA	2191	Amelia Lewis, TIBCO Software
2180	Don Box, Microsoft	2192	Rodney Limprecht, Microsoft
2181	Luis Felipe Cabrera, Microsoft	2193	Steve Lucco, Microsoft
2182	Doug Davis, IBM	2194	Don Mullen, TIBCO Software
2183	Donald Ferguson, IBM	2195	Anthony Nadalin, IBM
2184	Christopher Ferris, IBM	2196	Mark Nottingham, BEA
2185	Tom Freund, IBM	2197	David Orchard, BEA
2186	Mary Ann Hondo, IBM	2198	Jamie Roots, IBM
2187	John Ibbotson, IBM	2199	Shivajee Samdarshi, TIBCO Software
2188	Lei Jin, BEA	2200	John Shewchuk, Microsoft
2189	Chris Kaler, Microsoft	2201	Tony Storey, IBM
2190	David Langworthy-Editor, Microsoft		

2202 The following individuals have provided invaluable input into the initial contribution:

2203	Keith Ballinger, Microsoft	2217	David Ingham, Microsoft
2204	Stefan Batres, Microsoft	2218	Gopal Kakivaya, Microsoft
2205	Rebecca Bergersen, Iona	2219	Johannes Klein, Microsoft
2206	Allen Brown, Microsoft	2220	Frank Leymann, IBM
2207	Michael Conner, IBM	2221	Martin Nally, IBM
2208	George Copeland, Microsoft	2222	Peter Niblett, IBM
2209	Francisco Curbera, IBM	2223	Jeffrey Schlimmer, Microsoft
2210	Paul Fremantle, IBM	2224	James Snell, IBM
2211	Steve Graham, IBM	2225	Keith Stobie, Microsoft
2212	Pat Helland, Microsoft	2226	Satish Thatte, Microsoft
2213	Rick Hill, Microsoft	2227	Stephen Todd, IBM
2214	Scott Hinkelman, IBM	2228	Sanjiva Weerawarana, IBM
2215	Tim Holloway, IBM	2229	Roger Wolter, Microsoft
2216	Efim Hudis, Microsoft		

2230 The following individuals were members of the committee during the development of this specification:

2231	Abbie Barbir, Nortel	2250	Robert Freund, Hitachi
2232	Charlton Barreto, Adobe	2251	Peter Furniss, Erebor
2233	Stefan Batres, Microsoft	2252	Marc Goodner, Microsoft
2234	Hamid Ben Malek, Fujitsu	2253	Alastair Green, Choreology
2235	Andreas Bjarlestam, Ericsson	2254	Mike Grogan, Sun
2236	Toufic Boubez, Layer 7	2255	Ondrej Hrebicek, Microsoft
2237	Doug Bunting, Sun	2256	Kazunori Iwasa, Fujitsu
2238	Lloyd Burch, Novell	2257	Chamikara Jayalath, WSO2
2239	Steve Carter, Novell	2258	Lei Jin, BEA
2240	Martin Chapman, Oracle	2259	Ian Jones, BTplc
2241	Dave Chappell, Sonic	2260	Anish Karmarkar, Oracle
2242	Paul Cotton, Microsoft	2261	Paul Knight, Nortel
2243	Glen Daniels, Sonic	2262	Dan Leshchiner, Tibco
2244	Doug Davis, IBM	2263	Mark Little, JBoss
2245	Blake Dournaee, Intel	2264	Lily Liu, webMethods
2246	Jacques Durand, Fujitsu	2265	Matt Lovett, IBM
2247	Colleen Evans, Microsoft	2266	Ashok Malhotra, Oracle
2248	Christopher Ferris, IBM	2267	Jonathan Marsh, Microsoft
2249	Paul Fremantle, WSO2	2268	Daniel Millwood, IBM

2269	Jeff Mischkinsky, Oracle	2280	Stefan Rossmannith, SAP
2270	Nilo Mitra, Ericsson	2281	Tom Rutt, Fujitsu
2271	Peter Niblett, IBM	2282	Rich Salz, IBM
2272	Duane Nickull, Adobe	2283	Shivajee Samdarshi, Tibco
2273	Eisaku Nishiyama, Hitachi	2284	Vladimir Videlov, SAP
2274	Dave Orchard, BEA	2285	Claus von Riegen, SAP
2275	Chouthri Palanisamy, NEC	2286	Pete Wenzel, Sun
2276	Sanjay Patil, SAP	2287	Steve Winkler, SAP
2277	Gilbert Pilz, BEA	2288	Ümit Yalçinalp, SAP
2278	Martin Raepfle, SAP	2289	Nobuyuki Yamamoto, Hitachi
2279	Eric Rajkovic, Oracle		
2290			