



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

3 OASIS Standard

4 14 June 2007

5 **Specification URIs:**

6 **This Version:**

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

10 **Previous Version:**

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

14 **Latest Version:**

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

18 **Technical Committee:**

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

20 **Chairs:**

21 Paul Fremantle <paul@wso2.com>
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 v1.0

32 **Declared XML Namespaces:**

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

34 **Abstract:**

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

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

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

50 **Status:**

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

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

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

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

64 Notices

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

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

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

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

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

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

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

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

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

109 **Table of Contents**

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

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

158 1 Introduction

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

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

168 1.1 Terminology

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

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

- 173 • The syntax appears as an XML instance, but values in italics indicate data types instead of
174 values.
 - 175 • Characters are appended to elements and attributes to indicate cardinality:
 - 176 ○ "?" (0 or 1)
 - 177 ○ "*" (0 or more)
 - 178 ○ "+" (1 or more)
 - 179 • The character "|" is used to indicate a choice between alternatives.
 - 180 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group
181 with respect to cardinality or choice.
 - 182 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content
183 specified in this document. Additional children elements and/or attributes MAY be added at the
184 indicated extension points but they MUST NOT contradict the semantics of the parent and/or
185 owner, respectively. If an extension is not recognized it SHOULD be ignored.
 - 186 • XML namespace prefixes (see section 1.4) are used to indicate the namespace of the element
187 being defined.
- 188 Elements and Attributes defined by this specification are referred to in the text of this document using
189 XPath 1.0 [XPath_10] expressions. Extensibility points are referred to using an extended version of this
190 syntax:
- 191 • An element extensibility point is referred to using {any} in place of the element name. This
192 indicates that any element name can be used, from any namespace other than the wsrn:
193 namespace.
 - 194 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
195 indicates that any attribute name can be used, from any namespace other than the wsrn:
196 namespace.

197 1.2 Normative References

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

236 1.3 Non-Normative References

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

- 245 1999.
 246 <http://www.ietf.org/rfc/rfc2617.txt>
- 247 **[RFC 4346]** T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version
 248 1.1," April 2006.
 249 <http://www.ietf.org/rfc/rfc4346.txt>
- 250 **[WS-Policy]** W3C Member Submission "Web Services Policy 1.2 - Framework", April 2006
 251 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
 252
 253 W3C ~~Candidate~~ Recommendation, "Web Services Policy 1.5 - Framework,"
 254 ~~September~~February 2007.
 255 <http://www.w3.org/TR/2007/RECGR-ws-policy-20070920428>
- 256 **[WS-PolicyAttachment]** W3C Member Submission "Web Services Policy 1.2 - Attachment", April
 257 2006
 258 <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/>
 259
 260 W3C ~~Candidate~~ Recommendation, "Web Services Policy 1.5 - Attachment,"
 261 ~~September~~February 2007.
 262 <http://www.w3.org/TR/2007/RECGR-ws-policy-attach-20070904228>
- 263 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 264 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",
 265 OASIS Standard 200401, March 2004.
 266 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
 267 [security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
- 268
 269 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
 270 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)",
 271 OASIS Standard 200602, February 2006.
 272 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- 273 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",
 274 RFC 1323, May 1992.
 275 <http://www.rfc-editor.org/rfc/rfc1323.txt>
- 276 **[SecurityPolicy]** G. Della-Libra, et. al. "Web Services Security Policy Language (WS-
 277 SecurityPolicy)", July 2005
 278 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
- 279 **[SecureConversation]** S. Anderson, et al, "Web Services Secure Conversation Language (WS-
 280 SecureConversation)," February 2005.
 281 <http://schemas.xmlsoap.org/ws/2004/04/sc/>
- 282 **[Trust]** S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.
 283 <http://schemas.xmlsoap.org/ws/2005/02/trust>

Field Code Changed

Field Code Changed

284 1.4 Namespace

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

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

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

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

291 Table 1

Prefix	Namespace
--------	-----------

S	(Either SOAP 1.1 or 1.2)
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
wsm	http://docs.oasis-open.org/ws-rx/wsm/200702
wsa	http://www.w3.org/2005/08/addressing
wsam	http://www.w3.org/2007/052/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

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

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

295 **1.5 Conformance**

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

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

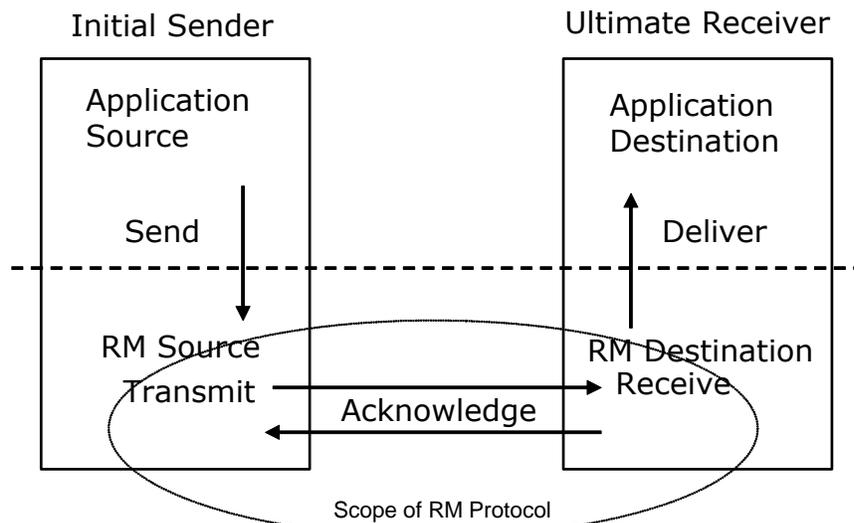
302 2 Reliable Messaging Model

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

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

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

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



328 Figure 1: Reliable Messaging Model

329 2.1 Glossary

330 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

360 2.2 Protocol Preconditions

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

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

370 2.3 Protocol Invariants

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

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

384 2.4 Delivery Assurances

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

389 **AtLeastOnce**

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

Formatted: Font: Not Bold

Formatted: Default Paragraph Font

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

397 AtMostOnce

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

402 ExactlyOnce

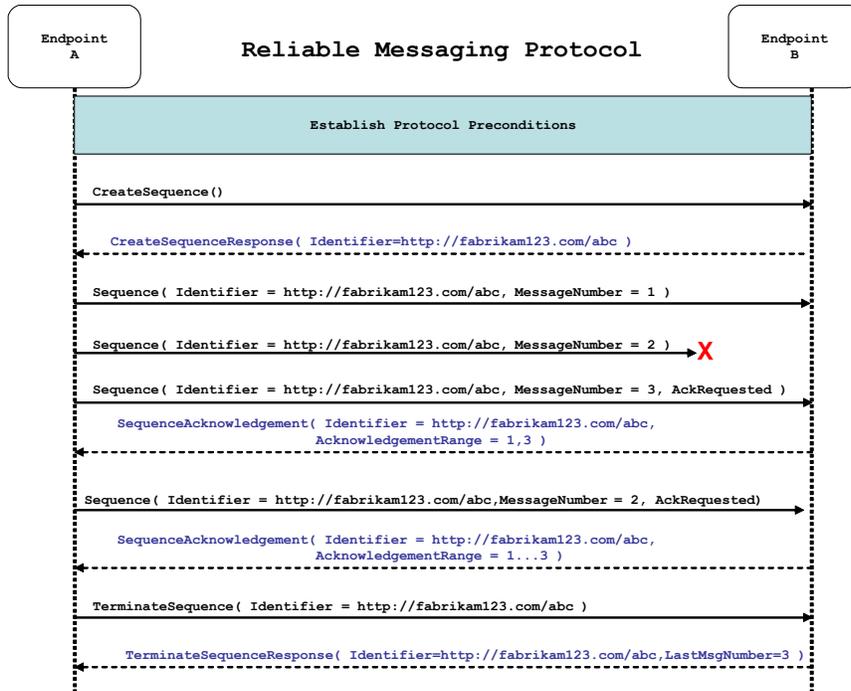
403 Each message is to be delivered exactly once; if a message cannot be delivered then an error
404 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is
405 that it SHOULD retry transmission of every message sent by the Application Source until it
406 receives an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

410 InOrder

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

422 2.5 Example Message Exchange

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



424 Figure 2: The WS-ReliableMessaging Protocol

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

448 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source
449 will not be sending any more messages. The RM Destination sends a
450 `TerminateSequenceResponse` message to the RM Source and reclaims any resources
451 associated with the Sequence.

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

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

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

465 3 RM Protocol Elements

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

468 3.1 Considerations on the Use of Extensibility Points

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

473 3.2 Considerations on the Use of "Piggy-Backing"

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

483 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

501 3.4 Sequence Creation

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

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

509 The following exemplar defines the `CreateSequence` syntax:

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

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

525 `/wsmr:CreateSequence`

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

530 `/wsmr:CreateSequence/wsmr:AcksTo`

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

540 `/wsmr:CreateSequence/wsmr:Expires`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

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

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

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

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

600 /wsmr:CreateSequence/{any}

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

603 /wsmr:CreateSequence/@{any}

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

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

610 The following exemplar defines the `CreateSequenceResponse` syntax:

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

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

624 /wsmr:CreateSequenceResponse

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

628 /wsmr:CreateSequenceResponse/wsmr:Identifier

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

Formatted: Font: (Default) Courier New

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

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

636 /wsmr:CreateSequenceResponse/wsmr:Expires

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

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

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

649 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

662 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

668 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

669 The RM Destination MUST include this element, of type `wsa:EndpointReferenceType` (as
670 specified by WS-Addressing). It specifies the endpoint reference to which messages containing
671 `SequenceAcknowledgement` header blocks and faults related to the created Sequence are to
672 be sent, unless otherwise noted in this specification (for example, see section 3.5).

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

Formatted: Font: (Default) Courier New

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

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

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

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

683 /wsm:CreateSequenceResponse/{any}

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

686 /wsm:CreateSequenceResponse/@{any}

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

689 3.5 Closing A Sequence

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

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

723 The following exemplar defines the `CloseSequence` syntax:

Formatted: Font: (Default) Courier New

```
724 <wsm:CloseSequence ...>  
725   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
726   <wsm>LastMsgNumber> wsm:MessageNumberType </wsm>LastMsgNumber> ?
```

```
727     ...
728 </wsrm:CloseSequence>
```

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

730 `/wsrm:CloseSequence`

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

734 `/wsrm:CloseSequence/wsrm:Identifier`

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

Formatted: Font: (Default) Courier New

738 `/wsrm:CloseSequence/wsrm:LastMessageNumber`

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

Formatted: Font: (Default) Courier New

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

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

745 `/wsrm:CloseSequence/{any}`

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

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

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

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

753 The following exemplar defines the `CloseSequenceResponse` syntax:

```
754 <wsrm:CloseSequenceResponse ...>
755   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
756   ...
757 </wsrm:CloseSequenceResponse>
```

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

759 `/wsrm:CloseSequenceResponse`

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

762 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

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

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

769 /wsm:CloseSequenceResponse/{any}

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

772 /wsm:CloseSequenceResponse/@{any}

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

775 3.6 Sequence Termination

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

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

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

794 The following exemplar defines the `TerminateSequence` syntax:

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

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

801 /wsm:TerminateSequence

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

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

813 /wsm:TerminateSequence/wsm:Identifier

Formatted: Font: (Default) Courier New

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

Formatted: Font: (Default) Courier New

817 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

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

Formatted: Font: (Default) Courier New

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

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

824 `/wsrm:TerminateSequence/{any}`

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

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

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

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

832 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
833 <wsrm:TerminateSequenceResponse ...>  
834   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
835   ...  
836 </wsrm:TerminateSequenceResponse>
```

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

838 `/wsrm:TerminateSequenceResponse`

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

842 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

849 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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

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

858 3.7 Sequences

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

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

866 A following exemplar defines its syntax:

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

872 The following describes the content model of the Sequence header block.

873 /wsrm:Sequence

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

Formatted: Font: (Default) Courier New

880 /wsrm:Sequence/wsrm:Identifier

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

884 /wsrm:Sequence/wsrm:Identifier/@{any}

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

887 /wsrm:Sequence/wsrm:MessageNumber

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

Formatted: Font: (Default) Courier New

892 /wsrm:Sequence/{any}

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

895 /wsrm:Sequence/@{any}

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

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

Formatted: Font: (Default) Courier New

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

903 3.8 Request Acknowledgement

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

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

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

918 The following exemplar defines its syntax:

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

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

924 `/wsrm:AckRequested`

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

926 `/wsrm:AckRequested/wsrm:Identifier`

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

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

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

934 `/wsrm:AckRequested/{any}`

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

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

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

940 3.9 Sequence Acknowledgement

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

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

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

961 The following exemplar defines its syntax:

```
962 <wsrm:SequenceAcknowledgement ...>  
963   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
964   [ [ [ <wsrm:AcknowledgementRange ...  
965     Upper="wsrm:MessageNumberType"  
966     Lower="wsrm:MessageNumberType"/> +  
967     | <wsrm:None/> ]  
968     <wsrm:Final/> ? ]  
969     | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]  
970   ...  
971   ...  
972 </wsrm:SequenceAcknowledgement>
```

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

974 `/wsrm:SequenceAcknowledgement`

975 This element contains the Sequence Acknowledgement information.

976 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

985 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

Formatted: Font: (Default) Courier New

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

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

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

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

1000 `/wsrm:SequenceAcknowledgement/wsrm:None`
1001 The RM Destination MUST include this element within a `SequenceAcknowledgement` header
1002 block if the RM Destination has not accepted any messages for the specified `Sequence`. The RM
1003 Destination MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack`
1004 element is also present as a child of the `SequenceAcknowledgement`.

1005 `/wsrm:SequenceAcknowledgement/wsrm:Final`
1006 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1007 block. This element indicates that the RM Destination is not receiving new messages for the
1008 specified `Sequence`. The RM Source can be assured that the ranges of messages acknowledged
1009 by this `SequenceAcknowledgement` header block will not change in the future. The RM
1010 Destination MUST include this element when the `Sequence` is closed. The RM Destination MUST
1011 NOT include this element when sending a `Nack`; it can only be used when sending
1012 `AcknowledgementRange` elements or a `None`.

1013 `/wsrm:SequenceAcknowledgement/wsrm:Nack`
1014 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
1015 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`
1016 representing the `MessageNumber` of an unreceived message in a `Sequence`. The RM Destination
1017 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is
1018 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM
1019 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT
1020 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously
1021 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a
1022 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been
1023 acknowledged within an `AcknowledgementRange`.

1024 `/wsrm:SequenceAcknowledgement/{any}`
1025 This is an extensibility mechanism to allow different (extensible) types of information, based on a
1026 schema, to be passed.

1027 `/wsrm:SequenceAcknowledgement/@{any}`
1028 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
1029 to the element.

Formatted: Font: (Default) Courier New

1030 The following examples illustrate SequenceAcknowledgement elements:

- 1031 • Message numbers 1...10 inclusive in a Sequence have been accepted by the RM Destination.

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

- 1036 • Message numbers 1..2, 4..6, and 8..10 inclusive in a Sequence have been accepted by the RM
1037 Destination, messages 3 and 7 have not been accepted.

```
1038 <wsrm:SequenceAcknowledgement>  
1039   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1040   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
1041   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
1042   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
1043 </wsrm:SequenceAcknowledgement>
```

- 1044 • Message number 3 in a Sequence has not been accepted by the RM Destination.

```
1045 <wsrm:SequenceAcknowledgement>  
1046   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
1047   <wsrm:Nack>3</wsrm:Nack>  
1048 </wsrm:SequenceAcknowledgement>
```

1049 4 Faults

1050 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. Create
1051 Sequence Refused is a possible fault reply for this operation. Unknown Sequence is a fault generated by
1052 Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences
1053 are detected. `WSRMRequired` is a fault generated by an RM Destination that requires the use of WS-RM
1054 on a Received message that did not use the protocol. All other faults in this section relate to known
1055 Sequences. Destinations that generate faults related to known Sequences SHOULD transmit those
1056 faults. If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement
1057 messages.

1058 Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault
1059 action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

1060 `http://docs.oasis-open.org/ws-rx/wsrn/200702/fault`

1061 The faults defined in this section are generated if the condition stated in the preamble is met. Fault
1062 handling rules are defined in section 6 of WS-Addressing SOAP Binding.

1063 The definitions of faults use the following properties:

1064 [Code] The fault code.

1065 [Subcode] The fault subcode.

1066 [Reason] The English language reason element.

1067 [Detail] The detail element(s). If absent, no detail element is defined for the fault. If more than one detail
1068 element is defined for a fault, implementations MUST include the elements in the order that they are
1069 specified.

1070 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
1071 "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

1072 The properties above bind to a SOAP 1.2 fault as follows:

```
1073 <S:Envelope>
1074 <S:Header>
1075   <wsa:Action>
1076     http://docs.oasis-open.org/ws-rx/wsrn/200702/fault
1077   </wsa:Action>
1078   <!-- Headers elided for brevity. -->
1079 </S:Header>
1080 <S:Body>
1081 <S:Fault>
1082   <S:Code>
1083     <S:Value> [Code] </S:Value>
1084     <S:Subcode>
1085       <S:Value> [Subcode] </S:Value>
1086     </S:Subcode>
1087   </S:Code>
1088   <S:Reason>
1089     <S:Text xml:lang="en"> [Reason] </S:Text>
1090   </S:Reason>
1091   <S:Detail>
```

```
1092     [Detail]
1093     ...
1094   </S:Detail>
1095 </S:Fault>
1096 </S:Body>
1097 </S:Envelope>
```

1098 The properties above bind to a SOAP 1.1 fault as follows when the fault is triggered by processing an RM
1099 header block:

```
1100 <S11:Envelope>
1101 <S11:Header>
1102   <wsrm:SequenceFault>
1103     <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1104     <wsrm:Detail> [Detail] </wsrm:Detail>
1105     ...
1106   </wsrm:SequenceFault>
1107   <!-- Headers elided for brevity. -->
1108 </S11:Header>
1109 <S11:Body>
1110 <S11:Fault>
1111   <faultcode> [Code] </faultcode>
1112   <faultstring> [Reason] </faultstring>
1113 </S11:Fault>
1114 </S11:Body>
1115 </S11:Envelope>
```

1116 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
1117 CreateSequence request message:

```
1118 <S11:Envelope>
1119 <S11:Body>
1120 <S11:Fault>
1121   <faultcode> [Subcode] </faultcode>
1122   <faultstring> [Reason] </faultstring>
1123 </S11:Fault>
1124 </S11:Body>
1125 </S11:Envelope>
```

1126 4.1 SequenceFault Element

1127 The purpose of the SequenceFault element is to carry the specific details of a fault generated during the
1128 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging
1129 nodes MUST use the SequenceFault container only in conjunction with the SOAP 1.1 fault mechanism.
1130 WS-ReliableMessaging nodes MUST NOT use the SequenceFault container in conjunction with the
1131 SOAP 1.2 binding.

1132 The following exemplar defines its syntax:

```
1133 <wsrm:SequenceFault ...>
1134   <wsrm:FaultCode> wsrm:FaultCode </wsrm:FaultCode>
1135   <wsrm:Detail> ... </wsrm:Detail> ?
1136   ...
1137 </wsrm:SequenceFault>
```

1138 The following describes the content model of the SequenceFault element.

1139 /wsrm:SequenceFault

1140 This is the element containing Sequence fault information for WS-ReliableMessaging

1141 /wsrm:SequenceFault/wsrm:FaultCode

- 1142 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this
 1143 element to a qualified name from the set of faults [Subcodes] defined below.
- 1144 `/wsm:SequenceFault/wsm:Detail`
 1145 This element, if present, carries application specific error information related to the fault being
 1146 described.
- 1147 `/wsm:SequenceFault/wsm:Detail/{any}`
 1148 The application specific error information related to the fault being described.
- 1149 `/wsm:SequenceFault/wsm:Detail/@{any}`
 1150 The application specific error information related to the fault being described.
- 1151 `/wsm:SequenceFault/{any}`
 1152 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 1153 schema, to be passed.
- 1154 `/wsm:SequenceFault/@{any}`
 1155 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
 1156 to the element.

1157 **4.2 Sequence Terminated**

- 1158 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
 1159 Endpoint of this decision.
- 1160 Properties:
- 1161 [Code] Sender or Receiver
- 1162 [Subcode] `wsm:SequenceTerminated`
- 1163 [Reason] The Sequence has been terminated due to an unrecoverable error.
- 1164 [Detail]
- 1165 `<wsm:Identifier ...> xs:anyURI </wsm: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.

1166 **4.3 Unknown Sequence**

- 1167 Properties:
- 1168 [Code] Sender
- 1169 [Subcode] `wsm:UnknownSequence`

1170 [Reason] The value of `wsrn:Identifier` is not a known Sequence identifier.

Formatted: Font: (Default) Courier New

1171 [Detail]

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

1173 4.4 Invalid Acknowledgement

1174 An example of when this fault is generated is when a message is Received by the RM Source containing
1175 a `SequenceAcknowledgement` covering messages that have not been sent.

1176 [Code] Sender

1177 [Subcode] `wsrn:InvalidAcknowledgement`

1178 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

Formatted: Font: (Default) Courier New

1179 [Detail]

```
<wsrm:SequenceAcknowledgement ...> ... </wsrm: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.

Formatted: Font: (Default) Courier New

Formatted: Font: (Default) Courier New

1181 4.5 Message Number Rollover

1182 If the condition listed below is reached, the RM Destination MUST generate this fault.

1183 Properties:

1184 [Code] Sender

1185 [Subcode] wsrn:MessageNumberRollover

1186 [Reason] The maximum value for `wsrn:MessageNumber` has been exceeded.

Formatted: Font: (Default) Courier New

1187 [Detail]

```
1188 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
1189 <wsrm:MaxMessageNumber> wsrn:MessageNumberType </wsrm:MaxMessageNumber>
```

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	Message number in <code>/wsrm:Sequence/wsrn:MessageNumber</code> 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.

1190 4.6 Create Sequence Refused

1191 Properties:

1192 [Code] Sender or Receiver

1193 [Subcode] wsrn:CreateSequenceRefused

1194 [Reason] The Create Sequence request has been refused by the RM Destination.

1195 [Detail]

```
1196 xs:any
```

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	In response to a <code>CreateSequence</code> message when the RM Destination does not wish to create a new Sequence.	Unspecified.	Sequence terminated.

1197 4.7 Sequence Closed

1198 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1199 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that is closed.

1201 Properties:

1202 [Code] Sender

1203 [Subcode] wsm:SequenceClosed

1204 [Reason] The Sequence is closed and cannot accept new messages.

1205 [Detail]

1206 `<wsm:Identifier...> xs:anyURI </wsm: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.

1207 **4.8 WSRM Required**

1208 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming message that did not use this protocol.

1210 Properties:

1211 [Code] Sender

1212 [Subcode] wsm:WSRMRequired

1213 [Reason] The RM Destination requires the use of WSRM.

1214 [Detail]

1215 `xs:any`

1216 5 Security Threats and Countermeasures

1217 This specification considers two sets of security requirements, those of the applications that use the WS-
1218 RM protocol and those of the protocol itself.

1219 This specification makes no assumptions about the security requirements of the applications that use WS-
1220 RM. However, once those requirements have been satisfied within a given operational context, the
1221 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1222 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1223 There are many other security concerns that one may need to consider when implementing or using this
1224 protocol. The material below should not be considered as a "check list". Implementers and users of this
1225 protocol are urged to perform a security analysis to determine their particular threat profile and the
1226 appropriate responses to those threats.

1227 Implementers are also advised that there is a core tension between security and reliable messaging that
1228 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1229 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1230 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1231 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1232 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1233 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1234 avoid and prevent this condition.

1235 5.1 Threats and Countermeasures

1236 The primary security requirement of this protocol is to protect the specified semantics and protocol
1237 invariants against various threats. The following sections describe several threats to the integrity and
1238 operation of this protocol and provide some general outlines of countermeasures to those threats.
1239 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1240 to all operational contexts.

1241 5.1.1 Integrity Threats

1242 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1243 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1244 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1245 to its intended message represents a threat to the WS-RM protocol.

1246 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM
1247 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1248 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be
1249 Delivered to the Application Destination in the same order that they were sent by the Application Source.

Formatted: Font: 10 pt

1250 5.1.1.1 Countermeasures

1251 Integrity threats are generally countered via the use of digital signatures some level of the communication
1252 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1253 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers
1254 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in
1255 which they occur, implementations MUST allow for signatures that cover only these headers.

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1256 5.1.2 Resource Consumption Threats

1257 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1258 implement that RM Destination. These resources can include network connections, database tables,
1259 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1260 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1261 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1262 message Delivery and use this Sequence to send a stream of very large messages to that service, making
1263 sure to omit message number "1" from that stream.

Formatted: Font: 10 pt

1264 5.1.2.1 Countermeasures

1265 There are a number of countermeasures against the described resource consumption threats. The
1266 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1267 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1268 some cases, allows the identity of any attackers to be determined.

1269 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and
1270 authenticate the RM Source that issued the `CreateSequence` message.

Formatted: Font: 10 pt

1271 5.1.3 Sequence Spoofing Threats

1272 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1273 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1274 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1275 appropriate RM Destination. Some `Sequence` spoofing attacks also require up-to-date knowledge of the
1276 current `MessageNumber` for their target Sequence.

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1277 In general any Sequence Lifecycle Message, RM Protocol Header Block, or `Sequence`-correlated SOAP
1278 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence
1279 `Identifier` to attack the Sequence. These attacks are "two-way" in that an attacker may choose to
1280 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a
1281 message that it sends to the `AcksTo` EPR of an RM Source.

Formatted: Font: 10 pt

Formatted: Font: (Default) Courier New

Formatted: Font: 10 pt

1282 5.1.3.1 Sequence Hijacking

1283 Sequence hijacking is a specific case of a `Sequence` spoofing attack. The attacker attempts to inject
1284 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1285 messages.

Formatted: Font: 10 pt

1286 Note that "`Sequence` hijacking" should not be equated with "security session hijacking". Although a
1287 Sequence may be bound to some form of a security session in order to counter the threats described in
1288 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1289 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1290 established by the security infrastructure to make such determinations. Failure to observe this rule
1291 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1292 the ability to authenticate its peers even though the necessary security processing has taken place.

1293 5.1.3.2 Countermeasures

1294 There are a number of countermeasures against `Sequence` spoofing threats. The technique advocated
1295 by this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM
1296 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination
1297 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1298 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it
1299 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the
1300 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it
1301 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the
1302 referenced Sequence.

1303 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and
1304 authenticate the entity that sent the CreateSequence message. Similarly for the RM Source to identify
1305 its Sequence peer it MUST be able to identify and authenticate the entity that sent the
1306 CreateSequenceResponse message. For either the RM Destination or the RM Source to determine if a
1307 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that
1308 message and, if necessary, correlate this identity with the Sequence peer identity established at
1309 Sequence creation time.

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1310 5.2 Security Solutions and Technologies

1311 The security threats described in the previous sections are neither new nor unique. The solutions that
1312 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1313 section maps the facilities provided by common web services security solutions against countermeasures
1314 described in the previous sections.

1315 Before continuing this discussion, however, some examination of the underlying requirements of the
1316 previously described countermeasures is necessary. Specifically it should be noted that the technique
1317 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1318 the issuer of a CreateSequence message. Secondly, the RM Destination performs an authorization
1319 check against this authenticated identity and determines if the RM Source is permitted to create
1320 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1321 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1322 discussion of such facilities is considered to be beyond the scope of this specification.

Formatted: Font: 10 pt

1323 5.2.1 Transport Layer Security

1324 This section describes how the facilities provided by SSL/TLS [RFC 4346] can be used to implement the
1325 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1326 defined in section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1327 The description provided here is general in nature and is not intended to serve as a complete definition on
1328 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1329 choice of features as well as the manner in which they will be used. The mechanisms described in the
1330 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the
1331 requirements and constraints of the use of SSL/TLS.

1332 5.2.1.1 Model

1333 The basic model for using SSL/TLS is as follows:

- 1334 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1335 2. The RM Source uses this SSL/TLS session to send a CreateSequence message to the RM
1336 Destination.
- 1337 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1338 asynchronous CreateSequenceResponse using this session. Alternately it may respond with a
1339 synchronous CreateSequenceResponse using the session established in (1).

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt

- 1340 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1341 any and all messages or faults that refer to that Sequence.
- 1342 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1343 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1344 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1345 5.2.1.2 Countermeasure Implementation

1346 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1347 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the
1348 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1349 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1350 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1351 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification
1352 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1353 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1354 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1355 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1356 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1357 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1358 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1359 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1360 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1361 Acknowledgement) using BasicAuth.
- 1362 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1363 connection authenticates itself to the party accepting the connection using an X.509 certificate
1364 that is exchanged during the SSL/TLS handshake.

1365 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1366 using one of the above mechanisms. The authenticated identity can then be used to determine if the RM
1367 Source is authorized to create a Sequence with the RM Destination.

1368 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1369 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1370 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1371 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1372 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1373 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1374 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1375 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1376 to protect that Sequence.

1377 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1378 countermeasures (such as associating specific authentication information with a Sequence) although such
1379 methods are not covered by this document.

1380 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1381 session) are outside the scope of this specification.

1382 5.2.2 SOAP Message Security

1383 The mechanisms described in WS-Security may be used in various ways to implement the
1384 countermeasures described in the previous sections. This specification advocates using the protocol
1385 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

Formatted: Font: 10 pt

1386 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1387 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1388 The description provided here is general in nature and is not intended to serve as a complete definition on
1389 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1390 need to agree on the choice of features as well as the manner in which they will be used. The
1391 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1392 describe the requirements and constraints of the use of WS-SecureConversation.

1393 5.2.2.1 Model

1394 The basic model for using WS-SecureConversation is as follows:

- 1395 1 The RM Source and the RM Destination create a WS-SecureConversation security context. This
1396 may involve the participation of third parties such as a security token service. The tokens
1397 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service
1398 tickets).
- 1399 2 During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1400 context that will be used to protect the Sequence. This is done so that, in cases where the
1401 `CreateSequence` message is signed by more than one security context, the RM Source can
1402 indicate which security context should be used to protect the newly created Sequence.
- 1403 3 For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1404 associated with the security context to sign (as defined by WS-Security) at least the body and
1405 any relevant WS-RM-defined headers of any and all messages or faults that refer to that
1406 Sequence.

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1407 5.2.2.2 Countermeasure Implementation

1408 Without relying upon any authentication information, the per-message signatures provide the necessary
1409 integrity qualities to counter the threats described in section 5.1.1.

1410 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1411 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1412 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1413 create a Sequence with the RM Destination.

1414 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1415 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1416 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1417 context rather than on any authentication claims that may have been established during security context
1418 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1419 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1420 document.

1421 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits
1422 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the
1423 association between a Sequence and its protecting security context cannot always be established
1424 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1425 `CreateSequenceResponse` messages may be signed by more than one security context.

Formatted: Font: 10 pt

Formatted: Font: 10 pt

1426 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1427 amending or renewing contexts) are outside the scope of this specification.

1428 6 Securing Sequences

1429 As noted in section 5, the RM Source and RM Destination should be able to protect their shared
1430 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1431 achieving this objective depending upon the underlying security infrastructure.

1432 6.1 Securing Sequences Using WS-Security

1433 One mechanism for protecting a Sequence is to include a security token using a
1434 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1435 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1436 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1437 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1438 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1439 there may be more than one token on a `CreateSequence` message or inferred from the communication
1440 context (e.g. transport protection).

1441 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1442 if the token being referenced supports such mechanism.

1443 The following exemplar defines the `CreateSequence` syntax when extended to include a
1444 `wsse:SecurityTokenReference`:

```
1445 <wsrm:CreateSequence ...>  
1446   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1447   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1448   <wsrm:Offer ...>  
1449     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1450     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1451     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1452     <wsrm:IncompleteSequenceBehavior>  
1453       wsrm:IncompleteSequenceBehaviorType  
1454     </wsrm:IncompleteSequenceBehavior> ?  
1455     ...  
1456   </wsrm:Offer> ?  
1457   ...  
1458   <wsse:SecurityTokenReference>  
1459     ...  
1460   </wsse:SecurityTokenReference> ?  
1461   ...  
1462 </wsrm:CreateSequence>
```

1463 The following describes the content model of the additional `CreateSequence` elements.

1464 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1465 This element uses the extensibility mechanism defined for the `CreateSequence` element
1466 (defined in section 3.4) to communicate an explicit reference to the security token, using a
1467 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and
1468 Destination MUST use to authorize messages for the created (and, if present, the offered)
1469 Sequence(s). All subsequent messages related to the created (and, if present, the offered)
1470 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token
1471 (e.g., by using or deriving from a private or secret key).

1472 When a RM Source transmits a `CreateSequence` that has been extended to include a
1473 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1474 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1475 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1476 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1477 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1478 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1479 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1480 in WS-Security still applies.

1481 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1482 <wsrm:UsesSequenceSTR ... />
```

1483 The following describes the content model of the `UsesSequenceSTR` header block.

1484 `/wsrm:UsesSequenceSTR`

1485 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that
1486 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`
1487 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly
1488 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus
1489 aborting the requested Sequence creation.

1490 The following is an example of a `CreateSequence` message using the

1491 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1492 <soap:Envelope ...>  
1493   <soap:Header>  
1494     ...  
1495     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1496     ...  
1497   </soap:Header>  
1498   <soap:Body>  
1499     <wsrm:CreateSequence>  
1500       <wsrm:AcksTo>  
1501         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1502       </wsrm:AcksTo>  
1503       <wsse:SecurityTokenReference>  
1504         ...  
1505       </wsse:SecurityTokenReference>  
1506     </wsrm:CreateSequence>  
1507   </soap:Body>  
1508 </soap:Envelope>
```

1509 6.2 Securing Sequences Using SSL/TLS

1510 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1511 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1512 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1513 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1514 SOAP header block within the `CreateSequence` message.

1515 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1516 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1517 The following describes the content model of the `UsesSequenceSSL` header block.

1518 `/wsrm:UsesSequenceSSL`

1519 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`
1520 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1521 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM
1522 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.
1523 The receiving RM Destination MUST understand and correctly implement the functionality
1524 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the
1525 requested Sequence creation.

1526 Note that the inclusion of the above header by the RM Source implies that all Sequence-related
1527 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1528 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1529 `CreateSequenceResponse` message.

1530 Appendix A. Schema

1531 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1532 Schema Part2] is located at:

1533 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1534 The following copy is provided for reference.

```
1535 <?xml version="1.0" encoding="UTF-8"?>
1536 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1537 OASIS trademark, IPR and other policies apply. -->
1538 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1539 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1540 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1541 rx/wsrn/200702" elementFormDefault="qualified"
1542 attributeFormDefault="unqualified">
1543 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1544 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1545 <!-- Protocol Elements -->
1546 <xs:complexType name="SequenceType">
1547 <xs:sequence>
1548 <xs:element ref="wsrm:Identifier"/>
1549 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1550 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1551 maxOccurs="unbounded"/>
1552 </xs:sequence>
1553 <xs:anyAttribute namespace="##other" processContents="lax"/>
1554 </xs:complexType>
1555 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1556 <xs:element name="SequenceAcknowledgement">
1557 <xs:complexType>
1558 <xs:sequence>
1559 <xs:element ref="wsrm:Identifier"/>
1560 <xs:choice>
1561 <xs:sequence>
1562 <xs:choice>
1563 <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1564 <xs:complexType>
1565 <xs:sequence/>
1566 <xs:attribute name="Upper" type="xs:unsignedLong"
1567 use="required"/>
1568 <xs:attribute name="Lower" type="xs:unsignedLong"
1569 use="required"/>
1570 <xs:anyAttribute namespace="##other" processContents="lax"/>
1571 </xs:complexType>
1572 </xs:element>
1573 <xs:element name="None">
1574 <xs:complexType>
1575 <xs:sequence/>
1576 </xs:complexType>
1577 </xs:element>
1578 </xs:choice>
1579 <xs:element name="Final" minOccurs="0">
1580 <xs:complexType>
1581 <xs:sequence/>
1582 </xs:complexType>
1583 </xs:element>
1584 </xs:sequence>
1585 <xs:element name="Nack" type="xs:unsignedLong"
```

```

1586 maxOccurs="unbounded"/>
1587   </xs:choice>
1588   <xs:any namespace="##other" processContents="lax" minOccurs="0"
1589 maxOccurs="unbounded"/>
1590 </xs:sequence>
1591 <xs:anyAttribute namespace="##other" processContents="lax"/>
1592 </xs:complexType>
1593 </xs:element>
1594 <xs:complexType name="AckRequestedType">
1595   <xs:sequence>
1596     <xs:element ref="wsrm:Identifier"/>
1597     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1598 maxOccurs="unbounded"/>
1599   </xs:sequence>
1600   <xs:anyAttribute namespace="##other" processContents="lax"/>
1601 </xs:complexType>
1602 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1603 <xs:element name="Identifier">
1604   <xs:complexType>
1605     <xs:annotation>
1606       <xs:documentation>
1607         This type is for elements whose [children] is an anyURI and can have
1608         arbitrary attributes.
1609       </xs:documentation>
1610     </xs:annotation>
1611     <xs:simpleContent>
1612       <xs:extension base="xs:anyURI">
1613         <xs:anyAttribute namespace="##other" processContents="lax"/>
1614       </xs:extension>
1615     </xs:simpleContent>
1616   </xs:complexType>
1617 </xs:element>
1618 <xs:element name="Address">
1619   <xs:complexType>
1620     <xs:simpleContent>
1621       <xs:extension base="xs:anyURI">
1622         <xs:anyAttribute namespace="##other" processContents="lax"/>
1623       </xs:extension>
1624     </xs:simpleContent>
1625   </xs:complexType>
1626 </xs:element>
1627 <xs:simpleType name="MessageNumberType">
1628   <xs:restriction base="xs:unsignedLong">
1629     <xs:minInclusive value="1"/>
1630     <xs:maxInclusive value="9223372036854775807"/>
1631   </xs:restriction>
1632 </xs:simpleType>
1633 <!-- Fault Container and Codes -->
1634 <xs:simpleType name="FaultCodes">
1635   <xs:restriction base="xs:QName">
1636     <xs:enumeration value="wsrm:SequenceTerminated"/>
1637     <xs:enumeration value="wsrm:UnknownSequence"/>
1638     <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1639     <xs:enumeration value="wsrm:MessageNumberRollover"/>
1640     <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1641     <xs:enumeration value="wsrm:SequenceClosed"/>
1642     <xs:enumeration value="wsrm:WSRMRequired"/>
1643   </xs:restriction>
1644 </xs:simpleType>
1645 <xs:complexType name="SequenceFaultType">
1646   <xs:sequence>
1647     <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1648     <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1649     <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1650 maxOccurs="unbounded"/>
1651 </xs:sequence>
1652 <xs:anyAttribute namespace="##other" processContents="lax"/>
1653 </xs:complexType>
1654 <xs:complexType name="DetailType">
1655 <xs:sequence>
1656 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1657 maxOccurs="unbounded"/>
1658 </xs:sequence>
1659 <xs:anyAttribute namespace="##other" processContents="lax"/>
1660 </xs:complexType>
1661 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1662 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1663 <xs:element name="CreateSequenceResponse"
1664 type="wsrm:CreateSequenceResponseType"/>
1665 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1666 <xs:element name="CloseSequenceResponse"
1667 type="wsrm:CloseSequenceResponseType"/>
1668 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1669 <xs:element name="TerminateSequenceResponse"
1670 type="wsrm:TerminateSequenceResponseType"/>
1671 <xs:complexType name="CreateSequenceType">
1672 <xs:sequence>
1673 <xs:element ref="wsrm:AcksTo"/>
1674 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1675 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1676 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1677 maxOccurs="unbounded">
1678 <xs:annotation>
1679 <xs:documentation>
1680 It is the authors intent that this extensibility be used to
1681 transfer a Security Token Reference as defined in WS-Security.
1682 </xs:documentation>
1683 </xs:annotation>
1684 </xs:any>
1685 </xs:sequence>
1686 <xs:anyAttribute namespace="##other" processContents="lax"/>
1687 </xs:complexType>
1688 <xs:complexType name="CreateSequenceResponseType">
1689 <xs:sequence>
1690 <xs:element ref="wsrm:Identifier"/>
1691 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1692 <xs:element name="IncompleteSequenceBehavior"
1693 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1694 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1695 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1696 maxOccurs="unbounded"/>
1697 </xs:sequence>
1698 <xs:anyAttribute namespace="##other" processContents="lax"/>
1699 </xs:complexType>
1700 <xs:complexType name="CloseSequenceType">
1701 <xs:sequence>
1702 <xs:element ref="wsrm:Identifier"/>
1703 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1704 minOccurs="0"/>
1705 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1706 maxOccurs="unbounded"/>
1707 </xs:sequence>
1708 <xs:anyAttribute namespace="##other" processContents="lax"/>
1709 </xs:complexType>
1710 <xs:complexType name="CloseSequenceResponseType">
1711 <xs:sequence>
1712 <xs:element ref="wsrm:Identifier"/>
1713 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1714 maxOccurs="unbounded"/>
1715 </xs:sequence>
1716 <xs:anyAttribute namespace="##other" processContents="lax"/>
1717 </xs:complexType>
1718 <xs:complexType name="TerminateSequenceType">
1719 <xs:sequence>
1720 <xs:element ref="wsrm:Identifier"/>
1721 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1722 minOccurs="0"/>
1723 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1724 maxOccurs="unbounded"/>
1725 </xs:sequence>
1726 <xs:anyAttribute namespace="##other" processContents="lax"/>
1727 </xs:complexType>
1728 <xs:complexType name="TerminateSequenceResponseType">
1729 <xs:sequence>
1730 <xs:element ref="wsrm:Identifier"/>
1731 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1732 maxOccurs="unbounded"/>
1733 </xs:sequence>
1734 <xs:anyAttribute namespace="##other" processContents="lax"/>
1735 </xs:complexType>
1736 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1737 <xs:complexType name="OfferType">
1738 <xs:sequence>
1739 <xs:element ref="wsrm:Identifier"/>
1740 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1741 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1742 <xs:element name="IncompleteSequenceBehavior"
1743 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1744 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1745 maxOccurs="unbounded"/>
1746 </xs:sequence>
1747 <xs:anyAttribute namespace="##other" processContents="lax"/>
1748 </xs:complexType>
1749 <xs:complexType name="AcceptType">
1750 <xs:sequence>
1751 <xs:element ref="wsrm:AcksTo"/>
1752 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1753 maxOccurs="unbounded"/>
1754 </xs:sequence>
1755 <xs:anyAttribute namespace="##other" processContents="lax"/>
1756 </xs:complexType>
1757 <xs:element name="Expires">
1758 <xs:complexType>
1759 <xs:simpleContent>
1760 <xs:extension base="xs:duration">
1761 <xs:anyAttribute namespace="##other" processContents="lax"/>
1762 </xs:extension>
1763 </xs:simpleContent>
1764 </xs:complexType>
1765 </xs:element>
1766 <xs:simpleType name="IncompleteSequenceBehaviorType">
1767 <xs:restriction base="xs:string">
1768 <xs:enumeration value="DiscardEntireSequence"/>
1769 <xs:enumeration value="DiscardFollowingFirstGap"/>
1770 <xs:enumeration value="NoDiscard"/>
1771 </xs:restriction>
1772 </xs:simpleType>
1773 <xs:element name="UsesSequenceSTR">
1774 <xs:complexType>
1775 <xs:sequence/>
1776 <xs:anyAttribute namespace="##other" processContents="lax"/>
1777 </xs:complexType>

```

```
1778 </xs:element>
1779 <xs:element name="UsesSequenceSSL">
1780 <xs:complexType>
1781 <xs:sequence/>
1782 <xs:anyAttribute namespace="##other" processContents="lax"/>
1783 </xs:complexType>
1784 </xs:element>
1785 <xs:element name="UnsupportedElement">
1786 <xs:simpleType>
1787 <xs:restriction base="xs:QName"/>
1788 </xs:simpleType>
1789 </xs:element>
1790 </xs:schema>
```

1791 Appendix B. WSDL

1792 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1793 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1794 present in exchanges with that endpoint.

1795 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1796 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1797 for a higher-level mechanism to indicate that WS-RM is engaged.

1798 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1799 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702.wsdl>

1800 The following non-normative copy is provided for reference.

```
1801 <?xml version="1.0" encoding="utf-8"?>
1802 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1803 OASIS trademark, IPR and other policies apply. -->
1804 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1805 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1806 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1807 xmlns:wsam="http://www.w3.org/2007/052/addressing/metadata"
1808 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1809 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsdl"
1810 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsdl">
1811
1812   <wsdl:types>
1813     <xs:schema
1814       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1815       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1816       200702.xsd"/>
1817     </xs:schema>
1818   </wsdl:types>
1819
1820   <wsdl:message name="CreateSequence">
1821     <wsdl:part name="create" element="rm:CreateSequence"/>
1822   </wsdl:message>
1823   <wsdl:message name="CreateSequenceResponse">
1824     <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1825   </wsdl:message>
1826   <wsdl:message name="CloseSequence">
1827     <wsdl:part name="close" element="rm:CloseSequence"/>
1828   </wsdl:message>
1829   <wsdl:message name="CloseSequenceResponse">
1830     <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1831   </wsdl:message>
1832   <wsdl:message name="TerminateSequence">
1833     <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1834   </wsdl:message>
1835   <wsdl:message name="TerminateSequenceResponse">
1836     <wsdl:part name="terminateResponse"
1837     element="rm:TerminateSequenceResponse"/>
1838   </wsdl:message>
1839
1840   <wsdl:portType name="SequenceAbstractPortType">
1841     <wsdl:operation name="CreateSequence">
1842       <wsdl:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1843       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1844       <wsdl:output message="tns:CreateSequenceResponse"
```

```
1845 wsam:Action="http://docs.oasis-open.org/ws-
1846 rx/wsrn/200702/CreateSequenceResponse"/>
1847 </wsdl:operation>
1848 <wsdl:operation name="CloseSequence">
1849 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1850 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1851 <wsdl:output message="tns:CloseSequenceResponse"
1852 wsam:Action="http://docs.oasis-open.org/ws-
1853 rx/wsrn/200702/CloseSequenceResponse"/>
1854 </wsdl:operation>
1855 <wsdl:operation name="TerminateSequence">
1856 <wsdl:input message="tns:TerminateSequence"
1857 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1858 <wsdl:output message="tns:TerminateSequenceResponse"
1859 wsam:Action="http://docs.oasis-open.org/ws-
1860 rx/wsrn/200702/TerminateSequenceResponse"/>
1861 </wsdl:operation>
1862 </wsdl:portType>
1863 </wsdl:definitions>
1864
```

1865 Appendix C. Message Examples

1866 Appendix C.1 Create Sequence

1867 Create Sequence

```
1868 <?xml version="1.0" encoding="UTF-8"?>
1869 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1870 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1871 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1872 <S:Header>
1873 <wsa:MessageID>
1874 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1875 </wsa:MessageID>
1876 <wsa:To>http://example.com/serviceB/123</wsa:To>
1877 <wsa:Action>http://docs.oasis-open.org/ws-
1878 rx/wsmr/200702/CreateSequence</wsa:Action>
1879 <wsa:ReplyTo>
1880 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1881 </wsa:ReplyTo>
1882 </S:Header>
1883 <S:Body>
1884 <wsmr:CreateSequence>
1885 <wsmr:AcksTo>
1886 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1887 </wsmr:AcksTo>
1888 </wsmr:CreateSequence>
1889 </S:Body>
1890 </S:Envelope>
```

1891 Create Sequence Response

```
1892 <?xml version="1.0" encoding="UTF-8"?>
1893 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1894 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1895 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1896 <S:Header>
1897 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1898 <wsa:RelatesTo>
1899 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1900 </wsa:RelatesTo>
1901 <wsa:Action>
1902 http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1903 </wsa:Action>
1904 </S:Header>
1905 <S:Body>
1906 <wsmr:CreateSequenceResponse>
1907 <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1908 </wsmr:CreateSequenceResponse>
1909 </S:Body>
1910 </S:Envelope>
```

1911 Appendix C.2 Initial Transmission

1912 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1913 figure. The three messages have the following headers; the third message is identified as the last
1914 message in the Sequence:

1915 Message 1

```
1916 <?xml version="1.0" encoding="UTF-8"?>
1917 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1918 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1919 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1920   <S:Header>
1921     <wsa:MessageID>
1922       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1923     </wsa:MessageID>
1924     <wsa:To>http://example.com/serviceB/123</wsa:To>
1925     <wsa:From>
1926       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1927     </wsa:From>
1928     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1929     <wsm:Sequence>
1930       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
1931       <wsm:MessageNumber>1</wsm:MessageNumber>
1932     </wsm:Sequence>
1933   </S:Header>
1934   <S:Body>
1935     <!-- Some Application Data -->
1936   </S:Body>
1937 </S:Envelope>
```

1938 Message 2

```
1939 <?xml version="1.0" encoding="UTF-8"?>
1940 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1941 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1942 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1943   <S:Header>
1944     <wsa:MessageID>
1945       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1946     </wsa:MessageID>
1947     <wsa:To>http://example.com/serviceB/123</wsa:To>
1948     <wsa:From>
1949       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1950     </wsa:From>
1951     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1952     <wsm:Sequence>
1953       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
1954       <wsm:MessageNumber>2</wsm:MessageNumber>
1955     </wsm:Sequence>
1956   </S:Header>
1957   <S:Body>
1958     <!-- Some Application Data -->
1959   </S:Body>
1960 </S:Envelope>
```

1961 Message 3

```
1962 <?xml version="1.0" encoding="UTF-8"?>
1963 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1964 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1965 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1966   <S:Header>
1967     <wsa:MessageID>
1968       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1969     </wsa:MessageID>
1970     <wsa:To>http://example.com/serviceB/123</wsa:To>
1971     <wsa:From>
1972       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1973     </wsa:From>
1974     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
```

```

1975 <wsrm:Sequence>
1976 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1977 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1978 </wsrm:Sequence>
1979 <wsrm:AckRequested>
1980 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1981 </wsrm:AckRequested>
1982 </S:Header>
1983 <S:Body>
1984 <!-- Some Application Data -->
1985 </S:Body>
1986 </S:Envelope>

```

1987 **Appendix C.3 First Acknowledgement**

1988 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1989 responds with an Acknowledgement for messages 1 and 3:

```

1990 <?xml version="1.0" encoding="UTF-8"?>
1991 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1992 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
1993 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1994 <S:Header>
1995 <wsa:MessageID>
1996 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1997 </wsa:MessageID>
1998 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1999 <wsa:From>
2000 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2001 </wsa:From>
2002 <wsa:Action>
2003 http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2004 </wsa:Action>
2005 <wsrm:SequenceAcknowledgement>
2006 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2007 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
2008 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
2009 </wsrm:SequenceAcknowledgement>
2010 </S:Header>
2011 <S:Body/>
2012 </S:Envelope>

```

2013 **Appendix C.4 Retransmission**

2014 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2015 requests an Acknowledgement:

```

2016 <?xml version="1.0" encoding="UTF-8"?>
2017 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2018 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2019 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2020 <S:Header>
2021 <wsa:MessageID>
2022 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2023 </wsa:MessageID>
2024 <wsa:To>http://example.com/serviceB/123</wsa:To>
2025 <wsa:From>
2026 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2027 </wsa:From>
2028 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2029 <wsrm:Sequence>

```

```

2030 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2031 <wsrm:MessageNumber>2</wsrm:MessageNumber>
2032 </wsrm:Sequence>
2033 <wsrm:AckRequested>
2034 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2035 </wsrm:AckRequested>
2036 </S:Header>
2037 <S:Body>
2038 <!-- Some Application Data -->
2039 </S:Body>
2040 </S:Envelope>

```

2041 Appendix C.5 Termination

2042 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2043 be terminated:

```

2044 <?xml version="1.0" encoding="UTF-8"?>
2045 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2046 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2047 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2048 <S:Header>
2049 <wsa:MessageID>
2050 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2051 </wsa:MessageID>
2052 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2053 <wsa:From>
2054 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2055 </wsa:From>
2056 <wsa:Action>
2057 http://docs.oasis-open.org/ws-rx/wsr/200702/SequenceAcknowledgement
2058 </wsa:Action>
2059 <wsrm:SequenceAcknowledgement>
2060 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2061 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2062 </wsrm:SequenceAcknowledgement>
2063 </S:Header>
2064 <S:Body/>
2065 </S:Envelope>

```

2066 Terminate Sequence

```

2067 <?xml version="1.0" encoding="UTF-8"?>
2068 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2069 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200702"
2070 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2071 <S:Header>
2072 <wsa:MessageID>
2073 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2074 </wsa:MessageID>
2075 <wsa:To>http://example.com/serviceB/123</wsa:To>
2076 <wsa:Action>
2077 http://docs.oasis-open.org/ws-rx/wsr/200702/TerminateSequence
2078 </wsa:Action>
2079 <wsa:From>
2080 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2081 </wsa:From>
2082 </S:Header>
2083 <S:Body>
2084 <wsrm:TerminateSequence>
2085 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2086 <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2087 </wsrm:TerminateSequence>

```

2088 </S:Body>
2089 </S:Envelope>

2090 **Terminate Sequence Response**

```
2091 <?xml version="1.0" encoding="UTF-8"?>
2092 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2093 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
2094 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2095 <S:Header>
2096 <wsa:MessageID>
2097 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2098 </wsa:MessageID>
2099 <wsa:To>http://example.com/serviceA/789</wsa:To>
2100 <wsa:Action>
2101 http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
2102 </wsa:Action>
2103 <wsa:RelatesTo>
2104 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2105 </wsa:RelatesTo>
2106 <wsa:From>
2107 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2108 </wsa:From>
2109 </S:Header>
2110 <S:Body>
2111 <wsm:TerminateSequenceResponse>
2112 <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2113 </wsm:TerminateSequenceResponse>
2114 </S:Body>
2115 </S:Envelope>
```

2116 Appendix D. State Tables

2117 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2118 The state tables describe the lifetime of a [Sequence](#) in both the RM Source and the RM Destination

2119 Legend:

2120 The first column of these tables contains the motivating event and has the following format:

Event 2121
<i>Event name</i> [source] {ref}

2122 Where:

2123 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
2124 described by the specification.

2125 • [source]: indicates the source of the event; one of:

2126 ○ [msg]: a Received message

2127 ○ [int]: an internal event such as the firing of a timer

2128 ○ [app]: the application

2129 ○ [unspec]: the source is unspecified

2130 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}

2131 Where:

2132 • action to take: indicates that the state machine performs the following action. Actions surrounded
2133 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
2134 "Transmit"

2135 • [next state]: indicates the state to which the state machine will advance upon the performance of
2136 the action. For ease of reading the next state "same" indicates that the state does not change.

2137 • {ref} is a reference to the document section describing the behavior in this cell

2138 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2139 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2140 described in this specification and does not indicate normal protocol operation. Implementations MAY
2141 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2142 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2143 combinations.

2144 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}			
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 [SameRollover]	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}			
Sequence Terminated Fault [msg] {4.2}	N/A		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] {3.47}	Terminate Sequence [None] {3.47}	Terminate Sequence [None] {3.47}	Terminate Sequence [None] {3.47}	Terminate Sequence [None] {3.47}
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}			

2145 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}	

2146 **Appendix E. Acknowledgments**

2147 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2148 authors:

2149	Ruslan Bilorusets, BEA	2161	Amelia Lewis, TIBCO Software
2150	Don Box, Microsoft	2162	Rodney Limprecht, Microsoft
2151	Luis Felipe Cabrera, Microsoft	2163	Steve Lucco, Microsoft
2152	Doug Davis, IBM	2164	Don Mullen, TIBCO Software
2153	Donald Ferguson, IBM	2165	Anthony Nadalin, IBM
2154	Christopher Ferris, IBM	2166	Mark Nottingham, BEA
2155	Tom Freund, IBM	2167	David Orchard, BEA
2156	Mary Ann Hondo, IBM	2168	Jamie Roots, IBM
2157	John Ibbotson, IBM	2169	Shivajee Samdarshi, TIBCO Software
2158	Lei Jin, BEA	2170	John Shewchuk, Microsoft
2159	Chris Kaler, Microsoft	2171	Tony Storey, IBM
2160	David Langworthy-Editor, Microsoft		

2172 The following individuals have provided invaluable input into the initial contribution:

2173	Keith Ballinger, Microsoft	2187	David Ingham, Microsoft
2174	Stefan Batres, Microsoft	2188	Gopal Kakivaya, Microsoft
2175	Rebecca Bergersen, Iona	2189	Johannes Klein, Microsoft
2176	Allen Brown, Microsoft	2190	Frank Leymann, IBM
2177	Michael Conner, IBM	2191	Martin Nally, IBM
2178	George Copeland, Microsoft	2192	Peter Niblett, IBM
2179	Francisco Curbera, IBM	2193	Jeffrey Schlimmer, Microsoft
2180	Paul Fremantle, IBM	2194	James Snell, IBM
2181	Steve Graham, IBM	2195	Keith Stobie, Microsoft
2182	Pat Helland, Microsoft	2196	Satish Thatte, Microsoft
2183	Rick Hill, Microsoft	2197	Stephen Todd, IBM
2184	Scott Hinkelman, IBM	2198	Sanjiva Weerawarana, IBM
2185	Tim Holloway, IBM	2199	Roger Wolter, Microsoft
2186	Efim Hudis, Microsoft		

2200 The following individuals were members of the committee during the development of this specification:

2201	Abbie Barbir, Nortel	2220	Robert Freund, Hitachi
2202	Charlton Barreto, Adobe	2221	Peter Furniss, Erebor
2203	Stefan Batres, Microsoft	2222	Marc Goodner, Microsoft
2204	Hamid Ben Malek, Fujitsu	2223	Alastair Green, Choreology
2205	Andreas Bjarlestam, Ericsson	2224	Mike Grogan, Sun
2206	Toufic Boubez, Layer 7	2225	Ondrej Hrebicek, Microsoft
2207	Doug Bunting, Sun	2226	Kazunori Iwasa, Fujitsu
2208	Lloyd Burch, Novell	2227	Chamikara Jayalath, WSO2
2209	Steve Carter, Novell	2228	Lei Jin, BEA
2210	Martin Chapman, Oracle	2229	Ian Jones, BTplc
2211	Dave Chappell, Sonic	2230	Anish Karmarkar, Oracle
2212	Paul Cotton, Microsoft	2231	Paul Knight, Nortel
2213	Glen Daniels, Sonic	2232	Dan Leshchiner, Tibco
2214	Doug Davis, IBM	2233	Mark Little, JBoss
2215	Blake Dournaee, Intel	2234	Lily Liu, webMethods
2216	Jacques Durand, Fujitsu	2235	Matt Lovett, IBM
2217	Colleen Evans, Microsoft	2236	Ashok Malhotra, Oracle
2218	Christopher Ferris, IBM	2237	Jonathan Marsh, Microsoft
2219	Paul Fremantle, WSO2	2238	Daniel Millwood, IBM

2239	Jeff Mischkinsky, Oracle	2250	Stefan Rossmannith, SAP
2240	Nilo Mitra, Ericsson	2251	Tom Rutt, Fujitsu
2241	Peter Niblett, IBM	2252	Rich Salz, IBM
2242	Duane Nickull, Adobe	2253	Shivajee Samdarshi, Tibco
2243	Eisaku Nishiyama, Hitachi	2254	Vladimir Videlov, SAP
2244	Dave Orchard, BEA	2255	Claus von Riegen, SAP
2245	Chouthri Palanisamy, NEC	2256	Pete Wenzel, Sun
2246	Sanjay Patil, SAP	2257	Steve Winkler, SAP
2247	Gilbert Pilz, BEA	2258	Ümit Yalçinalp, SAP
2248	Martin Raepfle, SAP	2259	Nobuyuki Yamamoto, Hitachi
2249	Eric Rajkovic, Oracle		
2260			