



Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2

OASIS Standard

2 February 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-os.pdf>
<http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-os.html>
<http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.2-spec-os.doc> (Authoritative)

Previous Version:

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

Latest Version:

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

Technical Committee:

OASIS Web Services Reliable Exchange (WS-RX) TC

Chairs:

Paul Fremantle <paul@wso2.com>
Sanjay Patil <sanjay.patil@sap.com>

Editors:

Doug Davis, IBM <dug@us.ibm.com>
Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>
Gilbert Pilz, BEA <gpilz@bea.com>
Steve Winkler, SAP <steve.winkler@sap.com>
Ümit Yalçınalp, SAP <umit.yalcinalp@sap.com>

Related Work:

This specification replaces or supercedes:

- WS-ReliableMessaging v1.1

Declared XML Namespaces:

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

Abstract:

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

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

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

Status:

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

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/ws-rx/>.

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

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/ws-rx/>.

Notices

Copyright © OASIS® 1993–2009. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

| | | |
|-----|--|----|
| 1 | Introduction..... | 6 |
| 1.1 | Terminology..... | 6 |
| 1.2 | Normative References..... | 7 |
| 1.3 | Non-Normative References..... | 7 |
| 1.4 | Namespace..... | 8 |
| 1.5 | Conformance..... | 9 |
| 2 | Reliable Messaging Model..... | 10 |
| 2.1 | Glossary..... | 11 |
| 2.2 | Protocol Preconditions..... | 12 |
| 2.3 | Protocol Invariants..... | 12 |
| 2.4 | Delivery Assurances..... | 12 |
| 2.5 | Example Message Exchange..... | 13 |
| 3 | RM Protocol Elements..... | 16 |
| 3.1 | Considerations on the Use of Extensibility Points..... | 16 |
| 3.2 | Considerations on the Use of "Piggy-Backing"..... | 16 |
| 3.3 | Composition with WS-Addressing..... | 16 |
| 3.4 | Sequence Creation..... | 17 |
| 3.5 | Closing A Sequence..... | 21 |
| 3.6 | Sequence Termination..... | 23 |
| 3.7 | Sequences..... | 25 |
| 3.8 | Request Acknowledgement..... | 26 |
| 3.9 | Sequence Acknowledgement..... | 27 |
| 4 | Faults..... | 30 |
| 4.1 | SequenceFault Element..... | 31 |
| 4.2 | Sequence Terminated..... | 32 |
| 4.3 | Unknown Sequence..... | 32 |
| 4.4 | Invalid Acknowledgement..... | 33 |
| 4.5 | Message Number Rollover..... | 33 |
| 4.6 | Create Sequence Refused..... | 34 |
| 4.7 | Sequence Closed..... | 34 |
| 4.8 | WSRM Required..... | 35 |
| 5 | Security Threats and Countermeasures..... | 36 |
| 5.1 | Threats and Countermeasures..... | 36 |
| 5.2 | Security Solutions and Technologies..... | 38 |
| 6 | Securing Sequences..... | 41 |

| | |
|---|----|
| 6.1 Securing Sequences Using WS-Security..... | 41 |
| 6.2 Securing Sequences Using SSL/TLS | 42 |
| Appendix A. Schema | 44 |
| Appendix B. WSDL | 49 |
| Appendix C. Message Examples | 51 |
| Appendix C.1 Create Sequence | 51 |
| Appendix C.2 Initial Transmission..... | 51 |
| Appendix C.3 First Acknowledgement | 53 |
| Appendix C.4 Retransmission | 53 |
| Appendix C.5 Termination | 54 |
| Appendix D. State Tables | 56 |
| Appendix E. Acknowledgments..... | 61 |

1 Introduction

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

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

1.1 Terminology

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

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

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

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

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

40 1.2 Normative References

- 41 **[KEYWORDS]** S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC
42 2119, Harvard University, March 1997
43 <http://www.ietf.org/rfc/rfc2119.txt>
- 44 **[WS-RM Policy]** OASIS Standard, "Web Services Reliable Messaging Policy Assertion(WS-RM
45 Policy)," February 2009
46 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.2-spec-os.doc>
- 47 **[SOAP 1.1]** W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
48 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 49 **[SOAP 1.2]** W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June
50 2003.
51 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
- 52 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
53 Generic Syntax," RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January
54 2005.
55 <http://ietf.org/rfc/rfc3986>
- 56 **[UUID]** P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN
57 Namespace," RFC 4122, Microsoft, Refactored Networks - LLC, DataPower
58 Technology Inc, July 2005
59 <http://www.ietf.org/rfc/rfc4122.txt>
- 60 **[XML]** W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth
61 Edition)", September 2006.
62 <http://www.w3.org/TR/REC-xml/>
- 63 **[XML-ns]** W3C Recommendation, "Namespaces in XML," 14 January 1999.
64 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
- 65 **[XML-Schema Part1]** W3C Recommendation, "XML Schema Part 1: Structures," October 2004.
66 <http://www.w3.org/TR/xmlschema-1/>
- 67 **[XML-Schema Part2]** W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
68 <http://www.w3.org/TR/xmlschema-2/>
- 69 **[XPath 1.0]** W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16
70 November 1999.
71 <http://www.w3.org/TR/xpath>
- 72 **[WSDL 1.1]** W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
73 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 74 **[WS-Addressing]** W3C Recommendation, "Web Services Addressing 1.0 – Core," May 2006.
75 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
76 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding," May
77 2006
78 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

79 1.3 Non-Normative References

- 80 **[BSP 1.0]** WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006
81 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
82
- 83 **[RDDL 2.0]** Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language
84 (RDDL) 2.0," January 2004
85 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>
- 86 **[RFC 2617]** J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L.
87 Stewart, "HTTP Authentication: Basic and Digest Access Authentication," June

88 1999.
89 <http://www.ietf.org/rfc/rfc2617.txt>

90 **[RFC 4346]** T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version
91 1.1," April 2006.
92 <http://www.ietf.org/rfc/rfc4346.txt>

93 **[WS-Policy]** W3C Recommendation, "Web Services Policy 1.5 - Framework," September
94 2007.
95 <http://www.w3.org/TR/2007/REC-ws-policy-20070904>

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

99 **[WS-Security]** Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
100 Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)",
101 OASIS Standard 200401, March 2004.
102 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
103

104 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS
105 Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)",
106 OASIS Standard 200602, February 2006.
107 <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

108 **[RTTM]** V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance",
109 RFC 1323, May 1992.
110 <http://www.rfc-editor.org/rfc/rfc1323.txt>

111 **[SecurityPolicy]** OASIS Standard, "WS-SecurityPolicy 1.3", February 2009
112 <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.doc>
113

114 **[SecureConversation]** OASIS Standard, "WS-SecureConversation 1.4", February 2009
115 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/os/ws-secureconversation-1.4-spec-os.doc>
116

117 **[Trust]** OASIS Standard, "WS-Trust 1.4", February 2009
118 <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.doc>

119 1.4 Namespace

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

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

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

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

126 Table 1

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

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

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

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

130 **1.5 Conformance**

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

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

137

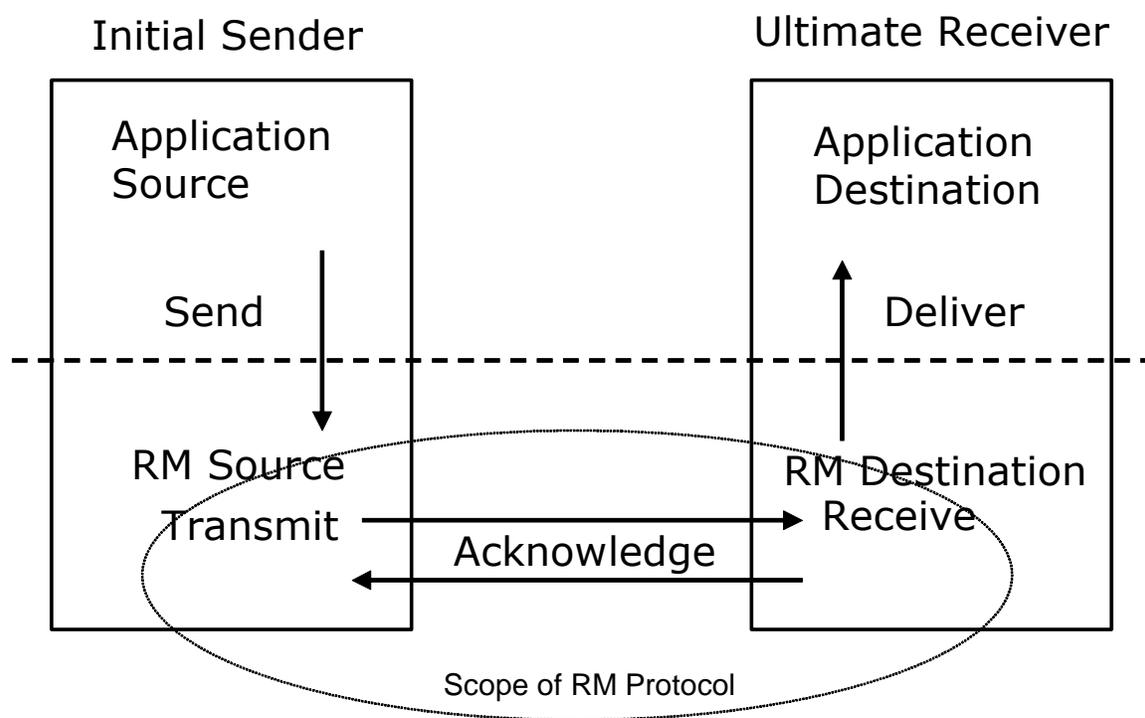
2 Reliable Messaging Model

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

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

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

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



163 Figure 1: Reliable Messaging Model

164 2.1 Glossary

165 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

195 2.2 Protocol Preconditions

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

- 198 • For any single message exchange the RM Source **MUST** have an endpoint reference that
199 uniquely identifies the RM Destination Endpoint.
- 200 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 201 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's
202 policies.
- 203 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**
204 have a security context.

205 2.3 Protocol Invariants

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

- 207 • The RM Source **MUST** assign each message within a `Sequence` a message number (defined
208 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers
209 **MUST** be assigned in the same order in which messages are sent by the Application Source.
- 210 • Within every `AcknowledgementMessage` it issues, the RM Destination **MUST** include one or
211 more `AcknowledgementRange` child elements that contain, in their collective ranges, the
212 message number of every message accepted by the RM Destination. The RM Destination **MUST**
213 exclude, in the `AcknowledgementRange` elements, the message numbers of any messages it
214 has not accepted. If no messages have been received the RM Destination **MUST** return `None`
215 instead of an `AcknowledgementRange(s)`. The RM Destination **MAY** transmit a `Nack` for a
216 specific message or messages instead of an `AcknowledgementRange(s)`.
- 217 • While the `Sequence` is not closed or terminated, the RM Source **SHOULD** retransmit
218 unacknowledged messages.

219 2.4 Delivery Assurances

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

223 **AtLeastOnce**

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

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

231 AtMostOnce

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

236 ExactlyOnce

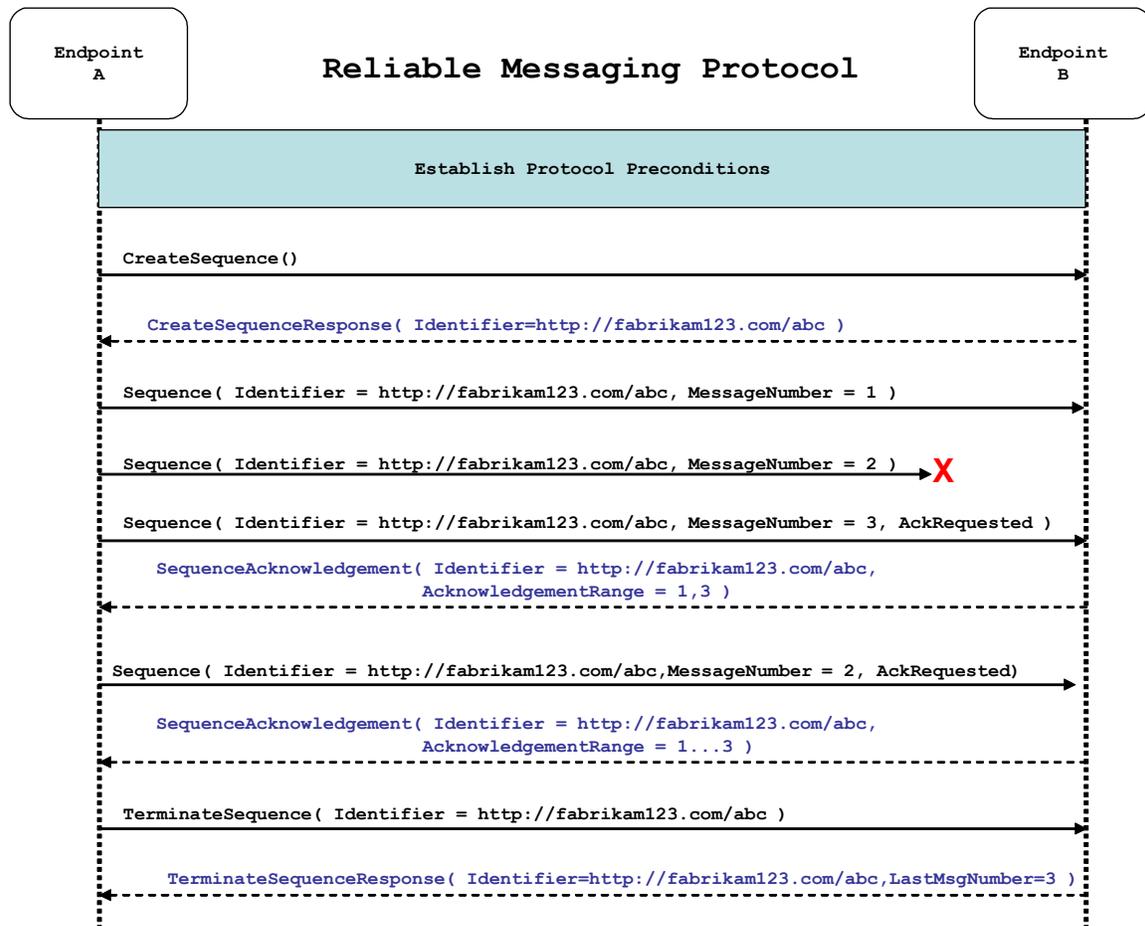
237 Each message is to be delivered exactly once; if a message cannot be delivered then an error
238 MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is
239 that it SHOULD retry transmission of every message sent by the Application Source until it
240 receives an acknowledgement from the RM Destination. The requirement on the RM Destination
241 is that it SHOULD retry the transfer to the Application Destination of any message that it accepts
242 from the RM Source until that message has been successfully delivered, and that it MUST NOT
243 deliver a duplicate of a message that has already been delivered.

244 InOrder

245 Messages from each individual Sequence are to be delivered in the same order they have been
246 sent by the Application Source. The requirement on an RM Source is that it MUST ensure that the
247 ordinal position of each message in the Sequence (as indicated by a message Sequence number)
248 is consistent with the order in which the messages have been sent from the Application Source.
249 The requirement on the RM Destination is that it MUST deliver received messages for each
250 Sequence in the order indicated by the message numbering. This DeliveryAssurance can be used
251 in combination with any of the AtLeastOnce, AtMostOnce or ExactlyOnce assertions, and the
252 requirements of those assertions MUST also be met. In particular if the AtLeastOnce or
253 ExactlyOnce assertion applies and the RM Destination detects a gap in the Sequence then the
254 RM Destination MUST NOT deliver any subsequent messages from that Sequence until the
255 missing messages are received or until the Sequence is closed.

256 2.5 Example Message Exchange

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



258 Figure 2: The WS-ReliableMessaging Protocol

- 259 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
260 and establishing trust.
- 261 2. The RM Source requests creation of a new Sequence.
- 262 3. The RM Destination creates a new Sequence and returns its unique Identifier.
- 263 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber
264 1. In the figure above, the RM Source sends 3 messages in the Sequence.
- 265 5. The 2nd message in the Sequence is lost in transit.
- 266 6. The 3rd message is the last in this Sequence and the RM Source includes an AckRequested
267 header to ensure that it gets a timely SequenceAcknowledgement for the Sequence.
- 268 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving
269 the RM Source's AckRequested header.
- 270 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new
271 message from the perspective of the underlying transport, but it has the same Sequence
272 Identifier and MessageNumber so the RM Destination can recognize it as a duplicate of the
273 earlier message, in case the original and retransmitted messages are both Received. The RM
274 Source includes an AckRequested header in the retransmitted message so the RM Destination
275 will expedite an acknowledgement.

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

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

282 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source
283 will not be sending any more messages. The RM Destination sends a
284 `TerminateSequenceResponse` message to the RM Source and reclaims any resources
285 associated with the Sequence.

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

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

299

3 RM Protocol Elements

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

302 3.1 Considerations on the Use of Extensibility Points

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

307 3.2 Considerations on the Use of "Piggy-Backing"

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

317 3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

335 3.4 Sequence Creation

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

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

343 The following exemplar defines the `CreateSequence` syntax:

```
344 <wsmr:CreateSequence ...>  
345   <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>  
346   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
347   <wsmr:Offer ...>  
348     <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
349     <wsmr:Endpoint> wsa:EndpointReferenceType </wsmr:Endpoint>  
350     <wsmr:Expires ...> xs:duration </wsmr:Expires> ?  
351     <wsmr:IncompleteSequenceBehavior>  
352       wsmr:IncompleteSequenceBehaviorType  
353     </wsmr:IncompleteSequenceBehavior> ?  
354     ...  
355   </wsmr:Offer> ?  
356   ...  
357 </wsmr:CreateSequence>
```

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

359 /wsmr:CreateSequence

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

364 /wsmr:CreateSequence/wsmr:AcksTo

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

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

374 /wsmr:CreateSequence/wsmr:Expires

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

379 /wsmr:CreateSequence/wsmr:Expires/@{any}

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

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

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

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

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

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

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

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

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

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

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

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

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

428 /wsrn:CreateSequence/wsrn:Offer/{any}

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

431 /wsrn:CreateSequence/wsrn:Offer/@{any}

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

434 /wsrn:CreateSequence/{any}

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

437 /wsrn:CreateSequence/@{any}

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

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

444 The following exemplar defines the *CreateSequenceResponse* syntax:

```
445   <wsrn:CreateSequenceResponse ...>  
446    <wsrn:Identifier ...> xs:anyURI </wsrn:Identifier>  
447    <wsrn:Expires ...> xs:duration </wsrn:Expires> ?  
448    <wsrn:IncompleteSequenceBehavior>  
449      wsrn:IncompleteSequenceBehaviorType  
450    </wsrn:IncompleteSequenceBehavior> ?  
451    <wsrn:Accept ...>  
452      <wsrn:AcksTo> wsa:EndpointReferenceType </wsrn:AcksTo>  
453      ...  
454    </wsrn:Accept> ?  
455    ...  
456   </wsrn:CreateSequenceResponse>
```

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

458 /wsrn:CreateSequenceResponse

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

462 /wsrn:CreateSequenceResponse/wsrn:Identifier

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

467 /wsrn:CreateSequenceResponse/wsrn:Identifier/@{any}

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

470 /wsrn:CreateSequenceResponse/wsrn:Expires

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

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

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

483 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`

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

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

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

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

496 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

502 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

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

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

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

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

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

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

517 /wsrm:CreateSequenceResponse/{any}

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

520 /wsrm:CreateSequenceResponse/@{any}

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

523 3.5 Closing A Sequence

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

529 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of
530 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept
531 any new messages for the specified Sequence, other than those already accepted at the time the
532 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or
533 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST
534 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`
535 element) header block on any messages associated with the Sequence destined to the RM Source,
536 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM
537 Source.

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

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

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

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

557 The following exemplar defines the `CloseSequence` syntax:

```
558 <wsrm:CloseSequence ...>  
559   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
```

560
561
562

```
<wsrm:LastMsgNumber> wsrm:MessageNumberType </wsrm:LastMsgNumber> ?  
...  
</wsrm:CloseSequence>
```

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

564 `/wsrm:CloseSequence`

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

568 `/wsrm:CloseSequence/wsrm:Identifier`

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

572 `/wsrm:CloseSequence/wsrm:LastMsgNumber`

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

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

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

579 `/wsrm:CloseSequence/{any}`

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

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

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

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

587 The following exemplar defines the `CloseSequenceResponse` syntax:

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

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

593 `/wsrm:CloseSequenceResponse`

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

596 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

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

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

603 /wsrm:CloseSequenceResponse/{any}

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

606 /wsrm:CloseSequenceResponse/@{any}

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

609 3.6 Sequence Termination

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

617 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM
618 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.
619 The RM Destination can use this information, for example, to implement the behavior indicated by
620 /wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior. The value of the
621 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the
622 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same
623 Sequence.

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

628 The following exemplar defines the `TerminateSequence` syntax:

```
629 <wsrm:TerminateSequence ...>  
630   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
631   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
632   ...  
633 </wsrm:TerminateSequence>
```

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

635 /wsrm:TerminateSequence

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

641 This element MAY also be sent by the RM Destination to indicate that it has unilaterally
642 terminated the Sequence. Upon sending this message the RM Destination MUST NOT accept
643 any additional messages (with the exception of the corresponding
644 `TerminateSequenceResponse`) for this Sequence. Upon receipt of a `TerminateSequence`
645 the RM Source MUST NOT send any additional messages (with the exception of the
646 corresponding `TerminateSequenceResponse`) for this Sequence.

647 /wsmr:TerminateSequence/wsmr:Identifier
648 The RM Source or RM Destination MUST include this element in any `TerminateSequence`
649 message it sends. The RM Source or RM Destination MUST set the value of this element to the
650 absolute URI (conformant with RFC3986) of the terminating Sequence.

651 /wsmr:TerminateSequence/wsmr:LastMsgNumber
652 The RM Source SHOULD include this element in any `TerminateSequence` message it sends.
653 The `LastMsgNumber` element specifies the highest assigned message number of all the
654 Sequence Traffic Messages for the terminating Sequence.

655 /wsmr:TerminateSequence/wsmr:Identifier/@{any}
656 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
657 to the element.

658 /wsmr:TerminateSequence/{any}
659 This is an extensibility mechanism to allow different (extensible) types of information, based on a
660 schema, to be passed.

661 /wsmr:TerminateSequence/@{any}
662 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
663 to the element.

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

666 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
667 <wsmr:TerminateSequenceResponse ...>  
668   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
669   ...  
670 </wsmr:TerminateSequenceResponse>
```

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

672 /wsmr:TerminateSequenceResponse
673 This element is sent in the body of a message in response to receipt of a `TerminateSequence`
674 request message. It indicates that the responder has terminated the Sequence. The responder
675 MUST NOT send this element as a header block.

676 /wsmr:TerminateSequenceResponse/wsmr:Identifier
677 The responder (RM Source or RM Destination) MUST include this element in any
678 `TerminateSequenceResponse` message it sends. The responder MUST set the value of this
679 element to the absolute URI (conformant with RFC3986) of the terminating Sequence.

680 /wsmr:TerminateSequenceResponse/wsmr:Identifier/@{any}
681 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
682 to the element.

683 /wsmr:TerminateSequenceResponse/{any}
684 This is an extensibility mechanism to allow different (extensible) types of information, based on a
685 schema, to be passed.

686 /wsmr:TerminateSequenceResponse/@{any}
687 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added
688 to the element.

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

692 3.7 Sequences

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

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

700 A following exemplar defines its syntax:

```
701 <wsrm:Sequence ...>  
702   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
703   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
704   ...  
705 </wsrm:Sequence>
```

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

707 `/wsrm:Sequence`

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

714 `/wsrm:Sequence/wsrm:Identifier`

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

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

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

721 `/wsrm:Sequence/wsrm:MessageNumber`

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

726 `/wsrm:Sequence/{any}`

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

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

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

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

```
733 <wsm:Sequence>
734   <wsm:Identifier>http://example.com/abc</wsm:Identifier>
735   <wsm:MessageNumber>10</wsm:MessageNumber>
736 </wsm:Sequence>
```

737 3.8 Request Acknowledgement

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

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

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

752 The following exemplar defines its syntax:

```
753 <wsm:AckRequested ...>
754   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>
755   ...
756 </wsm:AckRequested>
```

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

758 `/wsm:AckRequested`

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

760 `/wsm:AckRequested/wsm:Identifier`

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

765 `/wsm:AckRequested/wsm:Identifier/@{any}`

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

768 `/wsm:AckRequested/{any}`

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

771 `/wsm:AckRequested/@{any}`

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

774 3.9 Sequence Acknowledgement

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

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

785 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
786 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
787 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
788 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
789 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
790 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
791 header block for that same Sequence Identifier. When the RM Destination receives an
792 `AckRequested` header, and the `AcksTo` EPR for that Sequence is the WS-Addressing anonymous IRI,
793 the RM Destination SHOULD respond on the protocol binding-specific back-channel provided by the
794 Received message containing the `AckRequested` header block.

795 The following exemplar defines its syntax:

```
796 <wsm:SequenceAcknowledgement ...>  
797   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
798   [ [ [ <wsm:AcknowledgementRange ...  
799     Upper="wsm:MessageNumberType"  
800     Lower="wsm:MessageNumberType"/> +  
801     | <wsm:None/> ]  
802     <wsm:Final/> ? ]  
803     | <wsm:Nack> wsm:MessageNumberType </wsm:Nack> + ]  
804  
805     ...  
806 </wsm:SequenceAcknowledgement>
```

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

808 `/wsm:SequenceAcknowledgement`

809 This element contains the Sequence Acknowledgement information.

810 `/wsm:SequenceAcknowledgement/wsm:Identifier`

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

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

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

819 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange`

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

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

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

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

834 `/wsrm:SequenceAcknowledgement/wsrm:None`
835 The RM Destination MUST include this element within a `SequenceAcknowledgement` header
836 block if the RM Destination has not accepted any messages for the specified Sequence. The RM
837 Destination MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack`
838 element is also present as a child of the `SequenceAcknowledgement`.

839 `/wsrm:SequenceAcknowledgement/wsrm:Final`
840 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
841 block. This element indicates that the RM Destination is not receiving new messages for the
842 specified Sequence. The RM Source can be assured that the ranges of messages acknowledged
843 by this `SequenceAcknowledgement` header block will not change in the future. The RM
844 Destination MUST include this element when the Sequence is closed. The RM Destination MUST
845 NOT include this element when sending a `Nack`; it can only be used when sending
846 `AcknowledgementRange` elements or a `None`.

847 `/wsrm:SequenceAcknowledgement/wsrm:Nack`
848 The RM Destination MAY include this element within a `SequenceAcknowledgement` header
849 block. If used, the RM Destination MUST set the value of this element to a `MessageNumberType`
850 representing the `MessageNumber` of an unreceived message in a Sequence. The RM Destination
851 MUST NOT include a `Nack` element if a sibling `AcknowledgementRange` or `None` element is
852 also present as a child of `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM
853 Source SHOULD retransmit the message identified by the `Nack`. The RM Destination MUST NOT
854 issue a `SequenceAcknowledgement` containing a `Nack` for a message that it has previously
855 acknowledged within an `AcknowledgementRange`. The RM Source SHOULD ignore a
856 `SequenceAcknowledgement` containing a `Nack` for a message that has previously been
857 acknowledged within an `AcknowledgementRange`.

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

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

864 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
866 <wsrm:SequenceAcknowledgement>  
867   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
868   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
869 </wsrm:SequenceAcknowledgement>
```

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

```
872 <wsrm:SequenceAcknowledgement>  
873   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
874   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
875   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
876   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
877 </wsrm:SequenceAcknowledgement>
```

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

```
879 <wsrm:SequenceAcknowledgement>  
880   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
881   <wsrm:Nack>3</wsrm:Nack>  
882 </wsrm:SequenceAcknowledgement>
```

883 4 Faults

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

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

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

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

896 The definitions of faults use the following properties:

897 [Code] The fault code.

898 [Subcode] The fault subcode.

899 [Reason] The English language reason element.

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

903 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or
904 "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 |

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

```
906 <S:Envelope>
907   <S:Header>
908     <wsa:Action>
909       http://docs.oasis-open.org/ws-rx/wsrn/200702/fault
910     </wsa:Action>
911     <!-- Headers elided for brevity. -->
912   </S:Header>
913   <S:Body>
914     <S:Fault>
915       <S:Code>
916         <S:Value> [Code] </S:Value>
917         <S:Subcode>
918           <S:Value> [Subcode] </S:Value>
919         </S:Subcode>
920       </S:Code>
921       <S:Reason>
922         <S:Text xml:lang="en"> [Reason] </S:Text>
923       </S:Reason>
924     <S:Detail>
```

```
925     [Detail]
926     ...
927     </S:Detail>
928     </S:Fault>
929     </S:Body>
930 </S:Envelope>
```

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

```
933 <S11:Envelope>
934   <S11:Header>
935     <wsrm:SequenceFault>
936       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
937       <wsrm:Detail> [Detail] </wsrm:Detail>
938       ...
939     </wsrm:SequenceFault>
940     <!-- Headers elided for brevity. -->
941   </S11:Header>
942   <S11:Body>
943     <S11:Fault>
944       <faultcode> [Code] </faultcode>
945       <faultstring> [Reason] </faultstring>
946     </S11:Fault>
947   </S11:Body>
948 </S11:Envelope>
```

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

```
951 <S11:Envelope>
952   <S11:Body>
953     <S11:Fault>
954       <faultcode> [Subcode] </faultcode>
955       <faultstring> [Reason] </faultstring>
956     </S11:Fault>
957   </S11:Body>
958 </S11:Envelope>
```

959 4.1 SequenceFault Element

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

965 The following exemplar defines its syntax:

```
966 <wsrm:SequenceFault ...>
967   <wsrm:FaultCode> wsrm:FaultCode </wsrm:FaultCode>
968   <wsrm:Detail> ... </wsrm:Detail> ?
969   ...
970 </wsrm:SequenceFault>
```

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

972 /wsrm:SequenceFault

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

974 /wsrm:SequenceFault/wsrm:FaultCode

975 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this
 976 element to a qualified name from the set of faults [Subcodes] defined below.

977 `/wsrm:SequenceFault/wsrn:Detail`
 978 This element, if present, carries application specific error information related to the fault being
 979 described.

980 `/wsrm:SequenceFault/wsrn:Detail/{any}`
 981 The application specific error information related to the fault being described.

982 `/wsrm:SequenceFault/wsrn:Detail/@{any}`
 983 The application specific error information related to the fault being described.

984 `/wsrm:SequenceFault/{any}`
 985 This is an extensibility mechanism to allow different (extensible) types of information, based on a
 986 schema, to be passed.

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

4.2 Sequence Terminated

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

993 Properties:

994 [Code] Sender or Receiver

995 [Subcode] `wsrn:SequenceTerminated`

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

997 [Detail]

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

| Generated by | Condition | Action Upon Generation | Action Upon Receipt |
|------------------------------|--|------------------------|--|
| RM Source or RM Destination. | Encountering an unrecoverable condition or detection of violation of the protocol. | Sequence termination. | MUST terminate the Sequence if not otherwise terminated. |

4.3 Unknown Sequence

1000 Properties:

1001 [Code] Sender

1002 [Subcode] `wsrn:UnknownSequence`

1003 [Reason] The value of `wsrn:Identifier` is not a known Sequence identifier.

1004 [Detail]

1005 `<wsrn:Identifier ...> xs:anyURI </wsrn: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. |

1006 4.4 Invalid Acknowledgement

1007 An example of when this fault is generated is when a message is Received by the RM Source containing
1008 a `SequenceAcknowledgement` covering messages that have not been sent.

1009 [Code] Sender

1010 [Subcode] `wsrn:InvalidAcknowledgement`

1011 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1012 [Detail]

1013 `<wsrn:SequenceAcknowledgement ...> ... </wsrn: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. |

1014 4.5 Message Number Rollover

1015 If the condition listed below is reached, the RM Destination MUST generate this fault.

1016 Properties:

1017 [Code] Sender

1018 [Subcode] wsrM:MessageNumberRollover
 1019 [Reason] The maximum value for wsrM:MessageNumber has been exceeded.
 1020 [Detail]

```
1021 <wsrM:Identifier ...> xs:anyURI </wsrM:Identifier>
1022 <wsrM:MaxMessageNumber> wsrM:MessageNumberType </wsrM:MaxMessageNumber>
```

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

1023 **4.6 Create Sequence Refused**

1024 Properties:
 1025 [Code] Sender or Receiver
 1026 [Subcode] wsrM:CreateSequenceRefused
 1027 [Reason] The Create Sequence request has been refused by the RM Destination.
 1028 [Detail]

```
1029 xs:any
```

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

1030 **4.7 Sequence Closed**

1031 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.
 1032 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
 1033 is closed.
 1034 Properties:
 1035 [Code] Sender

- 1036 [Subcode] wsrn:SequenceClosed
- 1037 [Reason] The Sequence is closed and cannot accept new messages.
- 1038 [Detail]

1039 `<wsrm:Identifier...> xs:anyURI </wsrm:Identifier>`

| Generated by | Condition | Action Upon Generation | Action Upon Receipt |
|-----------------|---|------------------------|---------------------|
| RM Destination. | In response to a message that belongs to a Sequence that is already closed. | Unspecified. | Sequence closed. |

1040 **4.8 WSRM Required**

- 1041 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
- 1042 message that did not use this protocol.
- 1043 Properties:
- 1044 [Code] Sender
- 1045 [Subcode] wsrn:WSRMRequired
- 1046 [Reason] The RM Destination requires the use of WSRM.
- 1047 [Detail]

1048 `xs:any`

1049

5 Security Threats and Countermeasures

1050 This specification considers two sets of security requirements, those of the applications that use the WS-
1051 RM protocol and those of the protocol itself.

1052 This specification makes no assumptions about the security requirements of the applications that use WS-
1053 RM. However, once those requirements have been satisfied within a given operational context, the
1054 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;
1055 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1056 There are many other security concerns that one may need to consider when implementing or using this
1057 protocol. The material below should not be considered as a "check list". Implementers and users of this
1058 protocol are urged to perform a security analysis to determine their particular threat profile and the
1059 appropriate responses to those threats.

1060 Implementers are also advised that there is a core tension between security and reliable messaging that
1061 can be problematic if not addressed by implementations; one aspect of security is to prevent message
1062 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.
1063 Consequently, if the security sub-system processes a message but a failure occurs before the reliable
1064 messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system
1065 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-
1066 system will likely continue to expect and even solicit the missing message(s). Care should be taken to
1067 avoid and prevent this condition.

1068 5.1 Threats and Countermeasures

1069 The primary security requirement of this protocol is to protect the specified semantics and protocol
1070 invariants against various threats. The following sections describe several threats to the integrity and
1071 operation of this protocol and provide some general outlines of countermeasures to those threats.
1072 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable
1073 to all operational contexts.

1074 5.1.1 Integrity Threats

1075 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic
1076 Message, Sequence Lifecycle Message, Acknowledgement Messages, Acknowledgement Request, or
1077 Sequence-related fault, or which allows an attacker to alter the correlation of a RM Protocol Header Block
1078 to its intended message represents a threat to the WS-RM protocol.

1079 For example, if an attacker is able to swap *Sequence* headers on messages in transit between the RM
1080 Source and RM Destination then they have undermined the implementation's ability to guarantee the first
1081 invariant described in section 2.3. The result is that there is no way of guaranteeing that messages will be
1082 Delivered to the Application Destination in the same order that they were sent by the Application Source.

1083 5.1.1.1 Countermeasures

1084 Integrity threats are generally countered via the use of digital signatures some level of the communication
1085 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include
1086 both the SOAP body and any relevant SOAP headers (e.g. *Sequence* header). Because some headers
1087 (*AckRequested*, *SequenceAcknowledgement*) are independent of the body of the SOAP message in
1088 which they occur, implementations MUST allow for signatures that cover only these headers.

1089 **5.1.2 Resource Consumption Threats**

1090 The creation of a Sequence with an RM Destination consumes various resources on the systems used to
1091 implement that RM Destination. These resources can include network connections, database tables,
1092 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM
1093 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM
1094 Destination. Another attack is to create a Sequence for a service that is known to require in-order
1095 message Delivery and use this Sequence to send a stream of very large messages to that service, making
1096 sure to omit message number “1” from that stream.

1097 **5.1.2.1 Countermeasures**

1098 There are a number of countermeasures against the described resource consumption threats. The
1099 technique advocated by this specification is for the RM Destination to restrict the ability to create a
1100 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in
1101 some cases, allows the identity of any attackers to be determined.

1102 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability to identify and
1103 authenticate the RM Source that issued the `CreateSequence` message.

1104 **5.1.3 Sequence Spoofing Threats**

1105 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a
1106 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a
1107 fake `TerminateSequence` message that references the target Sequence and sends this message to the
1108 appropriate RM Destination. Some Sequence spoofing attacks also require up-to-date knowledge of the
1109 current `MessageNumber` for their target Sequence.

1110 In general any Sequence Lifecycle Message, RM Protocol Header Block, or Sequence-correlated SOAP
1111 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence
1112 `Identifier` to attack the Sequence. These attacks are “two-way” in that an attacker may choose to
1113 target the RM Source by, for example, inserting a fake `SequenceAcknowledgement` header into a
1114 message that it sends to the `AcksTo` EPR of an RM Source.

1115 **5.1.3.1 Sequence Hijacking**

1116 Sequence hijacking is a specific case of a Sequence spoofing attack. The attacker attempts to inject
1117 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those
1118 messages.

1119 Note that “Sequence hijacking” should not be equated with “security session hijacking”. Although a
1120 Sequence may be bound to some form of a security session in order to counter the threats described in
1121 this section, applications MUST NOT rely on WS-RM-related information to make determinations about
1122 the identity of the entity that created a message; applications SHOULD rely only upon information that is
1123 established by the security infrastructure to make such determinations. Failure to observe this rule
1124 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of
1125 the ability to authenticate its peers even though the necessary security processing has taken place.

1126 **5.1.3.2 Countermeasures**

1127 There are a number of countermeasures against Sequence spoofing threats. The technique advocated by
1128 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM
1129 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination
1130 that serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter

1131 Sequence spoofing attempts the RM Destination SHOULD ensure that every message or fault that it
1132 Receives that refers to a particular Sequence originated from the RM Source that jointly owns the
1133 referenced Sequence. For its part the RM Source SHOULD ensure that every message or fault that it
1134 Receives that refers to a particular Sequence originated from the RM Destination that jointly owns the
1135 referenced Sequence.

1136 For the RM Destination to be able to identify its Sequence peer it MUST be able to identify and
1137 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify
1138 its Sequence peer it MUST be able to identify and authenticate the entity that sent the
1139 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1140 message was sent by its Sequence peer it MUST be able to identify and authenticate the initiator of that
1141 message and, if necessary, correlate this identity with the Sequence peer identity established at
1142 Sequence creation time.

1143 **5.2 Security Solutions and Technologies**

1144 The security threats described in the previous sections are neither new nor unique. The solutions that
1145 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1146 section maps the facilities provided by common web services security solutions against countermeasures
1147 described in the previous sections.

1148 Before continuing this discussion, however, some examination of the underlying requirements of the
1149 previously described countermeasures is necessary. Specifically it should be noted that the technique
1150 described in section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1151 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization
1152 check against this authenticated identity and determines if the RM Source is permitted to create
1153 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1154 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1155 discussion of such facilities is considered to be beyond the scope of this specification.

1156 **5.2.1 Transport Layer Security**

1157 This section describes how the facilities provided by SSL/TLS [[RFC 4346](#)] can be used to implement the
1158 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1159 defined in section 4 of the Basic Security Profile 1.0 [[BSP 1.0](#)].

1160 The description provided here is general in nature and is not intended to serve as a complete definition on
1161 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1162 choice of features as well as the manner in which they will be used. The mechanisms described in the
1163 Web Services Security Policy Language [[SecurityPolicy](#)] MAY be used by services to describe the
1164 requirements and constraints of the use of SSL/TLS.

1165 **5.2.1.1 Model**

1166 The basic model for using SSL/TLS is as follows:

- 1167 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1168 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1169 Destination.
- 1170 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1171 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1172 synchronous `CreateSequenceResponse` using the session established in (1).

- 1173 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1174 any and all messages or faults that refer to that Sequence.
- 1175 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1176 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1177 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1178 5.2.1.2 Countermeasure Implementation

1179 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1180 necessary integrity qualities to counter the threats described in section 5.1.1. Note, however, that the
1181 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1182 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1183 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1184 As noted, the technique described in sections 5.1.2.1 involves the use of authentication. This specification
1185 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1186 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1187 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1188 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1189 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1190 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1191 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1192 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1193 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1194 Acknowledgement) using BasicAuth.
- 1195 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1196 connection authenticates itself to the party accepting the connection using an X.509 certificate
1197 that is exchanged during the SSL/TLS handshake.

1198 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1199 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1200 Source is authorized to create a Sequence with the RM Destination.

1201 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1202 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1203 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1204 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1205 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1206 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1207 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1208 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1209 to protect that Sequence.

1210 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1211 countermeasures (such as associating specific authentication information with a Sequence) although such
1212 methods are not covered by this document.

1213 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1214 session) are outside the scope of this specification.

1215 5.2.2 SOAP Message Security

1216 The mechanisms described in WS-Security may be used in various ways to implement the
1217 countermeasures described in the previous sections. This specification advocates using the protocol
1218 described by WS-SecureConversation [SecureConversation] (optionally in conjunction with WS-Trust

1219 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1220 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1221 The description provided here is general in nature and is not intended to serve as a complete definition on
1222 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1223 need to agree on the choice of features as well as the manner in which they will be used. The
1224 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1225 describe the requirements and constraints of the use of WS-SecureConversation.

1226 **5.2.2.1 Model**

1227 The basic model for using WS-SecureConversation is as follows:

- 1228 1 The RM Source and the RM Destination create a WS-SecureConversation security context. This
1229 may involve the participation of third parties such as a security token service. The tokens
1230 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service
1231 tickets).
- 1232 2 During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1233 context that will be used to protect the Sequence. This is done so that, in cases where the
1234 `CreateSequence` message is signed by more than one security context, the RM Source can
1235 indicate which security context should be used to protect the newly created Sequence.
- 1236 3 For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1237 associated with the security context to sign (as defined by WS-Security) at least the body and
1238 any relevant WS-RM-defined headers of any and all messages or faults that refer to that
1239 Sequence.

1240 **5.2.2.2 Countermeasure Implementation**

1241 Without relying upon any authentication information, the per-message signatures provide the necessary
1242 integrity qualities to counter the threats described in section 5.1.1.

1243 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1244 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1245 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1246 create a Sequence with the RM Destination.

1247 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1248 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1249 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1250 context rather than on any authentication claims that may have been established during security context
1251 initiation. Note that other methods of using WS-SecureConversation to implement the countermeasures
1252 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1253 document.

1254 As with transport security, the requisite equivalence of a security context peer with a Sequence peer limits
1255 the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security, the
1256 association between a Sequence and its protecting security context cannot always be established
1257 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1258 `CreateSequenceResponse` messages may be signed by more than one security context.

1259 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1260 amending or renewing contexts) are outside the scope of this specification.

6 Securing Sequences

1261
1262 As noted in section 5, the RM Source and RM Destination should be able to protect their shared
1263 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of
1264 achieving this objective depending upon the underlying security infrastructure.

6.1 Securing Sequences Using WS-Security

1265
1266 One mechanism for protecting a Sequence is to include a security token using a
1267 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-
1268 SecureConversation) in the `CreateSequence` element. This establishes an association between the
1269 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source
1270 and Destination MUST use the security token as the basis for authorization of all subsequent interactions
1271 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as
1272 there may be more than one token on a `CreateSequence` message or inferred from the communication
1273 context (e.g. transport protection).

1274 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,
1275 if the token being referenced supports such mechanism.

1276 The following exemplar defines the `CreateSequence` syntax when extended to include a
1277 `wsse:SecurityTokenReference`:

```
1278 <wsrm:CreateSequence ...>  
1279   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
1280   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1281   <wsrm:Offer ...>  
1282     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1283     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
1284     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1285     <wsrm:IncompleteSequenceBehavior>  
1286       wsrml:IncompleteSequenceBehaviorType  
1287     </wsrm:IncompleteSequenceBehavior> ?  
1288     ...  
1289   </wsrm:Offer> ?  
1290   ...  
1291   <wsse:SecurityTokenReference>  
1292     ...  
1293   </wsse:SecurityTokenReference> ?  
1294   ...  
1295 </wsrm:CreateSequence>
```

1296 The following describes the content model of the additional `CreateSequence` elements.

`/wsrm:CreateSequence/wsse:SecurityTokenReference`

1298 This element uses the extensibility mechanism defined for the `CreateSequence` element
1299 (defined in section 3.4) to communicate an explicit reference to the security token, using a
1300 `wsse:SecurityTokenReference` as documented in WS-Security, that the RM Source and
1301 Destination MUST use to authorize messages for the created (and, if present, the offered)
1302 Sequence(s). All subsequent messages related to the created (and, if present, the offered)
1303 Sequence(s) MUST demonstrate proof-of-possession of the secret associated with the token
1304 (e.g., by using or deriving from a private or secret key).

1305 When a RM Source transmits a `CreateSequence` that has been extended to include a
1306 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and

1307 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include
1308 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1309 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1310 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1311 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1312 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1313 in WS-Security still applies.

1314 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1315 <wsmr:UsesSequenceSTR ... />
```

1316 The following describes the content model of the `UsesSequenceSTR` header block.

1317 `/wsmr:UsesSequenceSTR`

1318 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that
1319 use the extensibility mechanism described above in this section. The `soap:mustUnderstand`
1320 attribute value MUST be 'true'. The receiving RM Destination MUST understand and correctly
1321 implement the extension described above or else generate a `soap:MustUnderstand` fault, thus
1322 aborting the requested Sequence creation.

1323 The following is an example of a `CreateSequence` message using the

1324 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1325 <soap:Envelope ...>  
1326   <soap:Header>  
1327     ...  
1328     <wsmr:UsesSequenceSTR soap:mustUnderstand='true' />  
1329     ...  
1330   </soap:Header>  
1331   <soap:Body>  
1332     <wsmr:CreateSequence>  
1333       <wsmr:AcksTo>  
1334         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1335       </wsmr:AcksTo>  
1336       <wsse:SecurityTokenReference>  
1337         ...  
1338       </wsse:SecurityTokenReference>  
1339     </wsmr:CreateSequence>  
1340   </soap:Body>  
1341 </soap:Envelope>
```

1342 6.2 Securing Sequences Using SSL/TLS

1343 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1344 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1345 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1346 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1347 SOAP header block within the `CreateSequence` message.

1348 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1349 <wsmr:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1350 The following describes the content model of the `UsesSequenceSSL` header block.

1351 `/wsmr:UsesSequenceSSL`

1352 The RM Source MAY include this element as a SOAP header block of a `CreateSequence`
1353 message to indicate to the RM Destination that the resulting Sequence is to be bound to the

1354 SSL/TLS session that was used to carry the `CreateSequence` message. If included, the RM
1355 Source MUST mark this header with a `soap:mustUnderstand` attribute with a value of 'true'.
1356 The receiving RM Destination MUST understand and correctly implement the functionality
1357 described in section 5.2.1 or else generate a `soap:MustUnderstand` fault, thus aborting the
1358 requested Sequence creation.

1359 Note that the inclusion of the above header by the RM Source implies that all Sequence-related
1360 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1361 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1362 `CreateSequenceResponse` message.

1363

Appendix A. Schema

1364 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1365 Schema Part2] is located at:

1366 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1367 The following copy is provided for reference.

```
1368 <?xml version="1.0" encoding="UTF-8"?>
1369 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1370 OASIS trademark, IPR and other policies apply. -->
1371 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1372 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1373 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1374 rx/wsrn/200702" elementFormDefault="qualified"
1375 attributeFormDefault="unqualified">
1376 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1377 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1378 <!-- Protocol Elements -->
1379 <xs:complexType name="SequenceType">
1380 <xs:sequence>
1381 <xs:element ref="wsrm:Identifier"/>
1382 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
1383 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1384 maxOccurs="unbounded"/>
1385 </xs:sequence>
1386 <xs:anyAttribute namespace="##other" processContents="lax"/>
1387 </xs:complexType>
1388 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1389 <xs:element name="SequenceAcknowledgement">
1390 <xs:complexType>
1391 <xs:sequence>
1392 <xs:element ref="wsrm:Identifier"/>
1393 <xs:choice>
1394 <xs:sequence>
1395 <xs:choice>
1396 <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1397 <xs:complexType>
1398 <xs:sequence/>
1399 <xs:attribute name="Upper" type="xs:unsignedLong"
1400 use="required"/>
1401 <xs:attribute name="Lower" type="xs:unsignedLong"
1402 use="required"/>
1403 <xs:anyAttribute namespace="##other" processContents="lax"/>
1404 </xs:complexType>
1405 </xs:sequence>
1406 <xs:element name="None">
1407 <xs:complexType>
1408 <xs:sequence/>
1409 </xs:complexType>
1410 </xs:choice>
1411 </xs:choice>
1412 <xs:element name="Final" minOccurs="0">
1413 <xs:complexType>
1414 <xs:sequence/>
1415 </xs:complexType>
1416 </xs:element>
1417 </xs:sequence>
1418 <xs:element name="Nack" type="xs:unsignedLong"
```

```

1419 maxOccurs="unbounded"/>
1420     </xs:choice>
1421     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1422 maxOccurs="unbounded"/>
1423     </xs:sequence>
1424     <xs:anyAttribute namespace="##other" processContents="lax"/>
1425 </xs:complexType>
1426 </xs:element>
1427 <xs:complexType name="AckRequestedType">
1428     <xs:sequence>
1429         <xs:element ref="wsrm:Identifier"/>
1430         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1431 maxOccurs="unbounded"/>
1432     </xs:sequence>
1433     <xs:anyAttribute namespace="##other" processContents="lax"/>
1434 </xs:complexType>
1435 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1436 <xs:element name="Identifier">
1437     <xs:complexType>
1438         <xs:annotation>
1439             <xs:documentation>
1440                 This type is for elements whose [children] is an anyURI and can have
1441 arbitrary attributes.
1442             </xs:documentation>
1443         </xs:annotation>
1444         <xs:simpleContent>
1445             <xs:extension base="xs:anyURI">
1446                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1447             </xs:extension>
1448         </xs:simpleContent>
1449     </xs:complexType>
1450 </xs:element>
1451 <xs:element name="Address">
1452     <xs:complexType>
1453         <xs:simpleContent>
1454             <xs:extension base="xs:anyURI">
1455                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1456             </xs:extension>
1457         </xs:simpleContent>
1458     </xs:complexType>
1459 </xs:element>
1460 <xs:simpleType name="MessageNumberType">
1461     <xs:restriction base="xs:unsignedLong">
1462         <xs:minInclusive value="1"/>
1463         <xs:maxInclusive value="9223372036854775807"/>
1464     </xs:restriction>
1465 </xs:simpleType>
1466 <!-- Fault Container and Codes -->
1467 <xs:simpleType name="FaultCodes">
1468     <xs:restriction base="xs:QName">
1469         <xs:enumeration value="wsrm:SequenceTerminated"/>
1470         <xs:enumeration value="wsrm:UnknownSequence"/>
1471         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1472         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1473         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1474         <xs:enumeration value="wsrm:SequenceClosed"/>
1475         <xs:enumeration value="wsrm:WSRMRequired"/>
1476     </xs:restriction>
1477 </xs:simpleType>
1478 <xs:complexType name="SequenceFaultType">
1479     <xs:sequence>
1480         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1481         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1482     <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1483 maxOccurs="unbounded"/>
1484 </xs:sequence>
1485 <xs:anyAttribute namespace="##other" processContents="lax"/>
1486 </xs:complexType>
1487 <xs:complexType name="DetailType">
1488 <xs:sequence>
1489 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1490 maxOccurs="unbounded"/>
1491 </xs:sequence>
1492 <xs:anyAttribute namespace="##other" processContents="lax"/>
1493 </xs:complexType>
1494 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1495 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1496 <xs:element name="CreateSequenceResponse"
1497 type="wsrm:CreateSequenceResponseType"/>
1498 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1499 <xs:element name="CloseSequenceResponse"
1500 type="wsrm:CloseSequenceResponseType"/>
1501 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1502 <xs:element name="TerminateSequenceResponse"
1503 type="wsrm:TerminateSequenceResponseType"/>
1504 <xs:complexType name="CreateSequenceType">
1505 <xs:sequence>
1506 <xs:element ref="wsrm:AcksTo"/>
1507 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1508 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1509 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1510 maxOccurs="unbounded">
1511 <xs:annotation>
1512 <xs:documentation>
1513 It is the authors intent that this extensibility be used to
1514 transfer a Security Token Reference as defined in WS-Security.
1515 </xs:documentation>
1516 </xs:annotation>
1517 </xs:any>
1518 </xs:sequence>
1519 <xs:anyAttribute namespace="##other" processContents="lax"/>
1520 </xs:complexType>
1521 <xs:complexType name="CreateSequenceResponseType">
1522 <xs:sequence>
1523 <xs:element ref="wsrm:Identifier"/>
1524 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1525 <xs:element name="IncompleteSequenceBehavior"
1526 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1527 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1528 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1529 maxOccurs="unbounded"/>
1530 </xs:sequence>
1531 <xs:anyAttribute namespace="##other" processContents="lax"/>
1532 </xs:complexType>
1533 <xs:complexType name="CloseSequenceType">
1534 <xs:sequence>
1535 <xs:element ref="wsrm:Identifier"/>
1536 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1537 minOccurs="0"/>
1538 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1539 maxOccurs="unbounded"/>
1540 </xs:sequence>
1541 <xs:anyAttribute namespace="##other" processContents="lax"/>
1542 </xs:complexType>
1543 <xs:complexType name="CloseSequenceResponseType">
1544 <xs:sequence>
1545 <xs:element ref="wsrm:Identifier"/>
1546 <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1547 maxOccurs="unbounded"/>
1548 </xs:sequence>
1549 <xs:anyAttribute namespace="##other" processContents="lax"/>
1550 </xs:complexType>
1551 <xs:complexType name="TerminateSequenceType">
1552 <xs:sequence>
1553 <xs:element ref="wsrm:Identifier"/>
1554 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1555 minOccurs="0"/>
1556 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1557 maxOccurs="unbounded"/>
1558 </xs:sequence>
1559 <xs:anyAttribute namespace="##other" processContents="lax"/>
1560 </xs:complexType>
1561 <xs:complexType name="TerminateSequenceResponseType">
1562 <xs:sequence>
1563 <xs:element ref="wsrm:Identifier"/>
1564 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1565 maxOccurs="unbounded"/>
1566 </xs:sequence>
1567 <xs:anyAttribute namespace="##other" processContents="lax"/>
1568 </xs:complexType>
1569 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1570 <xs:complexType name="OfferType">
1571 <xs:sequence>
1572 <xs:element ref="wsrm:Identifier"/>
1573 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1574 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1575 <xs:element name="IncompleteSequenceBehavior"
1576 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1577 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1578 maxOccurs="unbounded"/>
1579 </xs:sequence>
1580 <xs:anyAttribute namespace="##other" processContents="lax"/>
1581 </xs:complexType>
1582 <xs:complexType name="AcceptType">
1583 <xs:sequence>
1584 <xs:element ref="wsrm:AcksTo"/>
1585 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1586 maxOccurs="unbounded"/>
1587 </xs:sequence>
1588 <xs:anyAttribute namespace="##other" processContents="lax"/>
1589 </xs:complexType>
1590 <xs:element name="Expires">
1591 <xs:complexType>
1592 <xs:simpleContent>
1593 <xs:extension base="xs:duration">
1594 <xs:anyAttribute namespace="##other" processContents="lax"/>
1595 </xs:extension>
1596 </xs:simpleContent>
1597 </xs:complexType>
1598 </xs:element>
1599 <xs:simpleType name="IncompleteSequenceBehaviorType">
1600 <xs:restriction base="xs:string">
1601 <xs:enumeration value="DiscardEntireSequence"/>
1602 <xs:enumeration value="DiscardFollowingFirstGap"/>
1603 <xs:enumeration value="NoDiscard"/>
1604 </xs:restriction>
1605 </xs:simpleType>
1606 <xs:element name="UsesSequenceSTR">
1607 <xs:complexType>
1608 <xs:sequence/>
1609 <xs:anyAttribute namespace="##other" processContents="lax"/>

```

```
1610     </xs:complexType>
1611 </xs:element>
1612 <xs:element name="UsesSequenceSSL">
1613   <xs:complexType>
1614     <xs:sequence/>
1615     <xs:anyAttribute namespace="##other" processContents="lax"/>
1616   </xs:complexType>
1617 </xs:element>
1618 <xs:element name="UnsupportedElement">
1619   <xs:simpleType>
1620     <xs:restriction base="xs:QName"/>
1621   </xs:simpleType>
1622 </xs:element>
1623 </xs:schema>
```

Appendix B. WSDL

1624

1625 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1626 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1627 present in exchanges with that endpoint.

1628 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1629 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1630 for a higher-level mechanism to indicate that WS-RM is engaged.

1631 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1632 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-wsdl-200702e1.wsdl>

1633 The following non-normative copy is provided for reference.

```
1634 <?xml version="1.0" encoding="utf-8"?>
1635 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1636 OASIS trademark, IPR and other policies apply. -->
1637 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1638 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1639 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1640 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
1641 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1642 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL"
1643 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wSDL">
1644
1645   <wSDL:types>
1646     <xs:schema
1647       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1648       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1649       200702.xsd"/>
1650     </xs:schema>
1651   </wSDL:types>
1652
1653   <wSDL:message name="CreateSequence">
1654     <wSDL:part name="create" element="rm:CreateSequence"/>
1655   </wSDL:message>
1656   <wSDL:message name="CreateSequenceResponse">
1657     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1658   </wSDL:message>
1659   <wSDL:message name="CloseSequence">
1660     <wSDL:part name="close" element="rm:CloseSequence"/>
1661   </wSDL:message>
1662   <wSDL:message name="CloseSequenceResponse">
1663     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1664   </wSDL:message>
1665   <wSDL:message name="TerminateSequence">
1666     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1667   </wSDL:message>
1668   <wSDL:message name="TerminateSequenceResponse">
1669     <wSDL:part name="terminateResponse"
1670     element="rm:TerminateSequenceResponse"/>
1671   </wSDL:message>
1672
1673   <wSDL:portType name="SequenceAbstractPortType">
1674     <wSDL:operation name="CreateSequence">
1675       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1676       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1677       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1678 wsam:Action="http://docs.oasis-open.org/ws-
1679 rx/wsrn/200702/CreateSequenceResponse"/>
1680 </wsdl:operation>
1681 <wsdl:operation name="CloseSequence">
1682 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1683 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1684 <wsdl:output message="tns:CloseSequenceResponse"
1685 wsam:Action="http://docs.oasis-open.org/ws-
1686 rx/wsrn/200702/CloseSequenceResponse"/>
1687 </wsdl:operation>
1688 <wsdl:operation name="TerminateSequence">
1689 <wsdl:input message="tns:TerminateSequence"
1690 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1691 <wsdl:output message="tns:TerminateSequenceResponse"
1692 wsam:Action="http://docs.oasis-open.org/ws-
1693 rx/wsrn/200702/TerminateSequenceResponse"/>
1694 </wsdl:operation>
1695 </wsdl:portType>
1696
1697 </wsdl:definitions>
```

1698

Appendix C. Message Examples

1699

Appendix C.1 Create Sequence

1700

Create Sequence

1701

1702

1703

1704

1705

1706

1707

1708

1709

1710

1711

1712

1713

1714

1715

1716

1717

1718

1719

1720

1721

1722

1723

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
    </wsa:MessageID>
    <wsa:To>http://example.com/serviceB/123</wsa:To>
    <wsa:Action>http://docs.oasis-open.org/ws-
rx/wsmr/200702/CreateSequence</wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body>
    <wsmr:CreateSequence>
      <wsmr:AcksTo>
        <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
      </wsmr:AcksTo>
    </wsmr:CreateSequence>
  </S:Body>
</S:Envelope>
```

1724

Create Sequence Response

1725

1726

1727

1728

1729

1730

1731

1732

1733

1734

1735

1736

1737

1738

1739

1740

1741

1742

1743

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://Business456.com/serviceA/789</wsa:To>
    <wsa:RelatesTo>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
    </wsa:RelatesTo>
    <wsa:Action>
      http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
    </wsa:Action>
  </S:Header>
  <S:Body>
    <wsmr:CreateSequenceResponse>
      <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
    </wsmr:CreateSequenceResponse>
  </S:Body>
</S:Envelope>
```

1744

Appendix C.2 Initial Transmission

1745

1746

1747

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

1748 **Message 1**

```
1749 <?xml version="1.0" encoding="UTF-8"?>
1750 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1751 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1752 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1753   <S:Header>
1754     <wsa:MessageID>
1755       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1756     </wsa:MessageID>
1757     <wsa:To>http://example.com/serviceB/123</wsa:To>
1758     <wsa:From>
1759       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1760     </wsa:From>
1761     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1762     <wsmr:Sequence>
1763       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1764       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1765     </wsmr:Sequence>
1766   </S:Header>
1767   <S:Body>
1768     <!-- Some Application Data -->
1769   </S:Body>
1770 </S:Envelope>
```

1771 **Message 2**

```
1772 <?xml version="1.0" encoding="UTF-8"?>
1773 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1774 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1775 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1776   <S:Header>
1777     <wsa:MessageID>
1778       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1779     </wsa:MessageID>
1780     <wsa:To>http://example.com/serviceB/123</wsa:To>
1781     <wsa:From>
1782       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1783     </wsa:From>
1784     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1785     <wsmr:Sequence>
1786       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1787       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1788     </wsmr:Sequence>
1789   </S:Header>
1790   <S:Body>
1791     <!-- Some Application Data -->
1792   </S:Body>
1793 </S:Envelope>
```

1794 **Message 3**

```
1795 <?xml version="1.0" encoding="UTF-8"?>
1796 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1797 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1798 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1799   <S:Header>
1800     <wsa:MessageID>
1801       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1802     </wsa:MessageID>
1803     <wsa:To>http://example.com/serviceB/123</wsa:To>
1804     <wsa:From>
1805       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1806     </wsa:From>
```

```

1807 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1808 <wsrm:Sequence>
1809 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1810 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1811 </wsrm:Sequence>
1812 <wsrm:AckRequested>
1813 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1814 </wsrm:AckRequested>
1815 </S:Header>
1816 <S:Body>
1817 <!-- Some Application Data -->
1818 </S:Body>
1819 </S:Envelope>

```

1820 **Appendix C.3 First Acknowledgement**

1821 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1822 responds with an Acknowledgement for messages 1 and 3:

```

1823 <?xml version="1.0" encoding="UTF-8"?>
1824 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1825 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1826 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1827 <S:Header>
1828 <wsa:MessageID>
1829 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1830 </wsa:MessageID>
1831 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1832 <wsa:From>
1833 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1834 </wsa:From>
1835 <wsa:Action>
1836 http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1837 </wsa:Action>
1838 <wsrm:SequenceAcknowledgement>
1839 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1840 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1841 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1842 </wsrm:SequenceAcknowledgement>
1843 </S:Header>
1844 <S:Body/>
1845 </S:Envelope>

```

1846 **Appendix C.4 Retransmission**

1847 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
1848 requests an Acknowledgement:

```

1849 <?xml version="1.0" encoding="UTF-8"?>
1850 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1851 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1852 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1853 <S:Header>
1854 <wsa:MessageID>
1855 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1856 </wsa:MessageID>
1857 <wsa:To>http://example.com/serviceB/123</wsa:To>
1858 <wsa:From>
1859 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1860 </wsa:From>
1861 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>

```

```

1862 <wsrm:Sequence>
1863   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1864   <wsrm:MessageNumber>2</wsrm:MessageNumber>
1865 </wsrm:Sequence>
1866 <wsrm:AckRequested>
1867   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1868 </wsrm:AckRequested>
1869 </S:Header>
1870 <S:Body>
1871   <!-- Some Application Data -->
1872 </S:Body>
1873 </S:Envelope>

```

1874 Appendix C.5 Termination

1875 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
 1876 be terminated:

```

1877 <?xml version="1.0" encoding="UTF-8"?>
1878 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1879 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1880 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1881   <S:Header>
1882     <wsa:MessageID>
1883       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
1884     </wsa:MessageID>
1885     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1886     <wsa:From>
1887       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1888     </wsa:From>
1889     <wsa:Action>
1890       http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1891     </wsa:Action>
1892     <wsrm:SequenceAcknowledgement>
1893       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1894       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
1895     </wsrm:SequenceAcknowledgement>
1896   </S:Header>
1897   <S:Body/>
1898 </S:Envelope>

```

1899 Terminate Sequence

```

1900 <?xml version="1.0" encoding="UTF-8"?>
1901 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1902 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1903 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1904   <S:Header>
1905     <wsa:MessageID>
1906       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
1907     </wsa:MessageID>
1908     <wsa:To>http://example.com/serviceB/123</wsa:To>
1909     <wsa:Action>
1910       http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence
1911     </wsa:Action>
1912     <wsa:From>
1913       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1914     </wsa:From>
1915   </S:Header>
1916   <S:Body>
1917     <wsrm:TerminateSequence>
1918       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1919       <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>

```

1920 </wsm:TerminateSequence>
1921 </S:Body>
1922 </S:Envelope>

1923 Terminate Sequence Response

```
1924 <?xml version="1.0" encoding="UTF-8"?>
1925 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1926 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200702"
1927 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1928   <S:Header>
1929     <wsa:MessageID>
1930       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
1931     </wsa:MessageID>
1932     <wsa:To>http://example.com/serviceA/789</wsa:To>
1933     <wsa:Action>
1934       http://docs.oasis-open.org/ws-rx/wsm/200702/TerminateSequenceResponse
1935     </wsa:Action>
1936     <wsa:RelatesTo>
1937       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
1938     </wsa:RelatesTo>
1939     <wsa:From>
1940       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1941     </wsa:From>
1942   </S:Header>
1943   <S:Body>
1944     <wsm:TerminateSequenceResponse>
1945       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
1946     </wsm:TerminateSequenceResponse>
1947   </S:Body>
1948 </S:Envelope>
```

Appendix D. State Tables

1949

1950 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

1951 The state tables describe the lifetime of a Sequence in both the RM Source and the RM Destination

1952 Legend:

1953 The first column of these tables contains the motivating event and has the following format:

| |
|-------------------------------|
| Event ¹⁹⁵⁴ |
| <i>Event name</i> [source] |
| {ref} |

1955 Where:

- 1956 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
1957 described by the specification.
- 1958 • [source]: indicates the source of the event; one of:
 - 1959 ○ [msg] a Received message
 - 1960 ○ [int]: an internal event such as the firing of a timer
 - 1961 ○ [app]: the application
 - 1962 ○ [unspec]: the source is unspecified

1963 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} |

1964 Where:

- 1965 • action to take: indicates that the state machine performs the following action. Actions surrounded
1966 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
1967 "Transmit"
- 1968 • [next state]: indicates the state to which the state machine will advance upon the performance of
1969 the action. For ease of reading the next state "same" indicates that the state does not change.
- 1970 • {ref} is a reference to the document section describing the behavior in this cell

1971 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
1972 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
1973 described in this specification and does not indicate normal protocol operation. Implementations MAY
1974 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
1975 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
1976 combinations.

1977 Table 1 RM Source Sequence State Transition Table

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

| Events | Sequence States | | | | | |
|--|--|--|---|---|---|---|
| | None | Creating | Created | Closing | Closed | Terminating |
| SeqAck (final) [msg] {3.9} | Generate Unknown Sequence Fault [Same] {4.3} | Generate Unknown Sequence Fault [Same] {4.3} | Process Ack ranges [Closed] {3.9} | Process Ack ranges [Closed] {3.9} | Process Ack ranges [Same] | Process Ack ranges [Same] |
| Sequence Closed Fault [msg] {4.7} | Generate Unknown Sequence Fault [Same] {4.3} | Generate Unknown Sequence Fault [Same] {4.3} | No action [Closed] {4.7} | No action [Closed] {4.7} | No action [Same] | No action [Same] |
| Unknown Sequence Fault [msg] {4.3} | | | Terminate Sequence [None] {4.3} |
| 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] {3.4} | Terminate Sequence [None] {3.4} | Terminate Sequence [None] {3.4} | Terminate Sequence [None] {3.4} | Terminate Sequence [None] {3.4} |
| Invalid Acknowledgement [msg] {4.4} | Generate Unknown Sequence Fault [Same] {4.3} | Generate Unknown Sequence Fault [Same] {4.3} | Generate Invalid Acknowledgement Fault [Same] {4.4} |

1978 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} | |

1979

Appendix E. Acknowledgments

1980 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
1981 authors:

| | | | |
|------|------------------------------------|------|------------------------------------|
| 1982 | Ruslan Bilorusets, BEA | 1994 | Amelia Lewis, TIBCO Software |
| 1983 | Don Box, Microsoft | 1995 | Rodney Limprecht, Microsoft |
| 1984 | Luis Felipe Cabrera, Microsoft | 1996 | Steve Lucco, Microsoft |
| 1985 | Doug Davis, IBM | 1997 | Don Mullen, TIBCO Software |
| 1986 | Donald Ferguson, IBM | 1998 | Anthony Nadalin, IBM |
| 1987 | Christopher Ferris, IBM | 1999 | Mark Nottingham, BEA |
| 1988 | Tom Freund, IBM | 2000 | David Orchard, BEA |
| 1989 | Mary Ann Hondo, IBM | 2001 | Jamie Roots, IBM |
| 1990 | John Ibbotson, IBM | 2002 | Shivajee Samdarshi, TIBCO Software |
| 1991 | Lei Jin, BEA | 2003 | John Shewchuk, Microsoft |
| 1992 | Chris Kaler, Microsoft | 2004 | Tony Storey, IBM |
| 1993 | David Langworthy-Editor, Microsoft | | |

2005 The following individuals have provided invaluable input into the initial contribution:

| | | | |
|------|----------------------------|------|------------------------------|
| 2006 | Keith Ballinger, Microsoft | 2020 | David Ingham, Microsoft |
| 2007 | Stefan Batres, Microsoft | 2021 | Gopal Kakivaya, Microsoft |
| 2008 | Rebecca Bergersen, Iona | 2022 | Johannes Klein, Microsoft |
| 2009 | Allen Brown, Microsoft | 2023 | Frank Leymann, IBM |
| 2010 | Michael Conner, IBM | 2024 | Martin Nally, IBM |
| 2011 | George Copeland, Microsoft | 2025 | Peter Niblett, IBM |
| 2012 | Francisco Curbera, IBM | 2026 | Jeffrey Schlimmer, Microsoft |
| 2013 | Paul Fremantle, IBM | 2027 | James Snell, IBM |
| 2014 | Steve Graham, IBM | 2028 | Keith Stobie, Microsoft |
| 2015 | Pat Helland, Microsoft | 2029 | Satish Thatte, Microsoft |
| 2016 | Rick Hill, Microsoft | 2030 | Stephen Todd, IBM |
| 2017 | Scott Hinkelman, IBM | 2031 | Sanjiva Weerawarana, IBM |
| 2018 | Tim Holloway, IBM | 2032 | Roger Wolter, Microsoft |
| 2019 | Efim Hudis, Microsoft | | |

2033 The following individuals were members of the committee during the development of this specification:

| | | | |
|------|------------------------------|------|----------------------------|
| 2034 | Abbie Barbir, Nortel | 2053 | Robert Freund, Hitachi |
| 2035 | Charlton Barreto, Adobe | 2054 | Peter Furniss, Erebtor |
| 2036 | Stefan Batres, Microsoft | 2055 | Marc Goodner, Microsoft |
| 2037 | Hamid Ben Malek, Fujitsu | 2056 | Alastair Green, Choreology |
| 2038 | Andreas Bjarlestam, Ericsson | 2057 | Mike Grogan, Sun |
| 2039 | Toufic Boubez, Layer 7 | 2058 | Ondrej Hrebicek, Microsoft |
| 2040 | Doug Bunting, Sun | 2059 | Kazunori Iwasa, Fujitsu |
| 2041 | Lloyd Burch, Novell | 2060 | Chamikara Jayalath, WSO2 |
| 2042 | Steve Carter, Novell | 2061 | Lei Jin, BEA |
| 2043 | Martin Chapman, Oracle | 2062 | Ian Jones, BTplc |
| 2044 | Dave Chappell, Sonic | 2063 | Anish Karmarkar, Oracle |
| 2045 | Paul Cotton, Microsoft | 2064 | Paul Knight, Nortel |
| 2046 | Glen Daniels, Sonic | 2065 | Dan Leshchiner, Tibco |
| 2047 | Doug Davis, IBM | 2066 | Mark Little, JBoss |
| 2048 | Blake Dournaee, Intel | 2067 | Lily Liu, webMethods |
| 2049 | Jacques Durand, Fujitsu | 2068 | Matt Lovett, IBM |
| 2050 | Colleen Evans, Microsoft | 2069 | Ashok Malhotra, Oracle |
| 2051 | Christopher Ferris, IBM | 2070 | Jonathan Marsh, Microsoft |
| 2052 | Paul Fremantle, WSO2 | 2071 | Daniel Millwood, IBM |

| | | | |
|------|---------------------------|------|----------------------------|
| 2072 | Jeff Mischkinsky, Oracle | 2083 | Stefan Rossmanith, SAP |
| 2073 | Nilo Mitra, Ericsson | 2084 | Tom Rutt, Fujitsu |
| 2074 | Peter Niblett, IBM | 2085 | Rich Salz, IBM |
| 2075 | Duane Nickull, Adobe | 2086 | Shivajee Samdarshi, Tibco |
| 2076 | Eisaku Nishiyama, Hitachi | 2087 | Vladimir Videlov, SAP |
| 2077 | Dave Orchard, BEA | 2088 | Claus von Riegen, SAP |
| 2078 | Chouthri Palanisamy, NEC | 2089 | Pete Wenzel, Sun |
| 2079 | Sanjay Patil, SAP | 2090 | Steve Winkler, SAP |
| 2080 | Gilbert Pilz, BEA | 2091 | Ümit Yalçinalp, SAP |
| 2081 | Martin Raeppele, SAP | 2092 | Nobuyuki Yamamoto, Hitachi |
| 2082 | Eric Rajkovic, Oracle | | |
| 2093 | | | |