



# 1 Web Services Reliable Messaging 2 (WS-ReliableMessaging)

## 3 Working Draft 16, November 20, 2006

### 4 Document identifier:

5 wsrn-1.1-spec-wd-16

### 6 Location:

7 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-spec-wd-16.pdf>

### 8 Editors:

9 Doug Davis, IBM <[dug@us.ibm.com](mailto:dug@us.ibm.com)>  
10 Anish Karmarkar, Oracle <[Anish.Karmarkar@oracle.com](mailto:Anish.Karmarkar@oracle.com)>  
11 Gilbert Pilz, BEA <[gpilz@bea.com](mailto:gpilz@bea.com)>  
12 Steve Winkler, SAP <[steve.winkler@sap.com](mailto:steve.winkler@sap.com)>  
13 Ümit Yalçınalp, SAP <[umit.yalcinalp@sap.com](mailto:umit.yalcinalp@sap.com)>

### 14 Contributors:

15 See the Acknowledgments (Appendix E).

### 16 Abstract:

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

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

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

### 31 Status:

32 This document was last revised or approved by the WS-RX on the above date. The level of approval is  
33 also listed above. Check the current location noted above for possible later revisions of this document.  
34 This document is updated periodically on no particular schedule. Technical Committee members should  
35 send comments on this specification to the Technical Committee's email list. Others should send  
36 comments to the Technical Committee by using the "Send A Comment" button on the Technical  
37 Committee's web page at <http://www.oasis-open.org/committees/ws-rx>. For information on whether any  
38 patents have been disclosed that may be essential to implementing this specification, and any offers of  
39 patent licensing terms, please refer to the Intellectual Property Rights section of the Technical  
40 Committee web page (<http://www.oasis-open.org/committees/ws-rx/ipr.php>). The non-normative errata  
41 page for this specification is located at <http://www.oasis-open.org/committees/ws-rx>.

## 42 Table of Contents

43	1 Introduction.....	4
44	1.1 Notational Conventions.....	4
45	1.2 Namespace.....	5
46	1.3 Conformance.....	5
47	2 Reliable Messaging Model.....	6
48	2.1 Glossary.....	6
49	2.2 Protocol Preconditions.....	7
50	2.3 Protocol Invariants.....	8
51	2.4 Example Message Exchange.....	8
52	3 RM Protocol Elements.....	10
53	3.1 Considerations on the Use of Extensibility Points.....	10
54	3.2 Considerations on the Use of "Piggy-Backing".....	10
55	3.3 Composition with WS-Addressing.....	10
56	3.4 Sequence Creation.....	10
57	3.5 Closing A Sequence.....	15
58	3.6 Sequence Termination.....	16
59	3.7 Sequences.....	18
60	3.8 Request Acknowledgement.....	19
61	3.9 Sequence Acknowledgement.....	20
62	4 Faults.....	23
63	4.1 SequenceFault Element.....	24
64	4.2 Sequence Terminated.....	25
65	4.3 Unknown Sequence.....	25
66	4.4 Invalid Acknowledgement.....	26
67	4.5 Message Number Rollover.....	26
68	4.6 Create Sequence Refused.....	27
69	4.7 Sequence Closed.....	27
70	4.8 WSRM Required.....	28
71	5 Security Threats and Countermeasures.....	29
72	5.1 Threats and Countermeasures.....	29
73	5.1.1 Integrity Threats.....	29
74	5.1.1.1 Countermeasures.....	29
75	5.1.2 Resource Consumption Threats.....	30
76	5.1.2.1 Countermeasures.....	30
77	5.1.3 Sequence Spoofing Threats.....	30
78	5.1.3.1 Sequence Hijacking.....	30
79	5.1.3.2 Countermeasures.....	30

80	5.2 Security Solutions and Technologies.....	31
81	5.2.1 Transport Layer Security.....	31
82	5.2.1.1 Model.....	31
83	5.2.1.2 Countermeasure Implementation.....	32
84	5.2.2 SOAP Message Security.....	33
85	5.2.2.1 Model.....	33
86	5.2.2.2 Countermeasure Implementation.....	33
87	6 Securing Sequences.....	35
88	6.1 Securing Sequences Using WS-Security.....	35
89	6.2 Securing Sequences Using SSL/TLS.....	36
90	7 References.....	38
91	7.1 Normative.....	38
92	7.2 Non-Normative.....	39
93	Appendix A. Schema.....	41
94	Appendix B. WSDL.....	46
95	Appendix C. Message Examples.....	48
96	Appendix C.1 Create Sequence.....	48
97	Appendix C.2 Initial Transmission.....	48
98	Appendix C.3 First Acknowledgement.....	50
99	Appendix C.4 Retransmission.....	50
100	Appendix C.5 Termination.....	51
101	Appendix D. State Tables.....	53
102	Appendix E. Acknowledgments.....	57
103	Appendix F. Revision History.....	58
104	Appendix G. Notices.....	64

# 105 **1 Introduction**

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

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

## 115 **1.1 Notational Conventions**

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

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

- 120 • The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- 121 • Characters are appended to elements and attributes to indicate cardinality:
  - 122 ○ "?" (0 or 1)
  - 123 ○ "\*" (0 or more)
  - 124 ○ "+" (1 or more)
- 125 • The character "|" is used to indicate a choice between alternatives.
- 126 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group  
127 with respect to cardinality or choice.
- 128 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content  
129 specified in this document. Additional children elements and/or attributes MAY be added at the  
130 indicated extension points but they MUST NOT contradict the semantics of the parent and/or  
131 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 132 • XML namespace prefixes (See Section [1.2](#)) are used to indicate the namespace of the element  
133 being defined.

134 Elements and Attributes defined by this specification are referred to in the text of this document using  
135 XPath 1.0 [[XPATH 1.0](#)] expressions. Extensibility points are referred to using an extended version of this  
136 syntax:

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

## 143 1.2 Namespace

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

145 <http://docs.oasis-open.org/ws-rx/wsrn/200608>

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

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

150 Table 1

Prefix	Namespace
S	(Either SOAP 1.1 or 1.2)
S11	<a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a>
S12	<a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a>
wsrn	<a href="http://docs.oasis-open.org/ws-rx/wsrn/200608">http://docs.oasis-open.org/ws-rx/wsrn/200608</a>
wsa	<a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a>
wsaw	<a href="http://www.w3.org/2006/05/addressing/wsdl">http://www.w3.org/2006/05/addressing/wsdl</a>
wsse	<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a>
xs	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>

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

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

## 154 1.3 Conformance

155 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or  
156 REQUIRED level requirements defined herein. A SOAP Node MUST NOT use the XML namespace  
157 identifier for this specification (listed in Section 1.2) within SOAP Envelopes unless it is conformant with  
158 this specification.

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

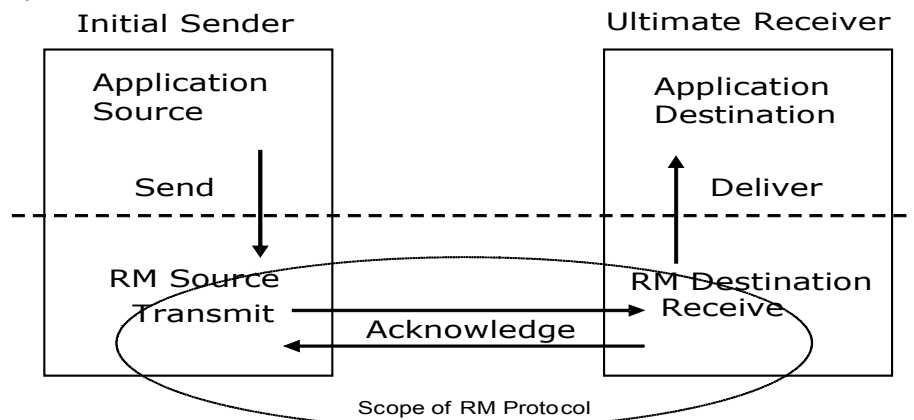
## 161 2 Reliable Messaging Model

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

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

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

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



187 Figure 1: Reliable Messaging Model

### 188 2.1 Glossary

189 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

203 **Deliver:** The act of transferring a message from the RM Destination to the Application Destination.

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

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

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

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

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

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

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

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

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

## 218 **2.2 Protocol Preconditions**

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

- 221 • For any single message exchange the RM Source **MUST** have an endpoint reference that uniquely  
222 identifies the RM Destination Endpoint.
- 223 • The RM Source **MUST** have successfully created a `Sequence` with the RM Destination.
- 224 • The RM Source **MUST** be capable of formulating messages that adhere to the RM Destination's  
225 policies.
- 226 • If a secure exchange of messages is **REQUIRED**, then the RM Source and RM Destination **MUST**  
227 have a security context.

## 228 2.3 Protocol Invariants

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

- 230 • The RM Source MUST assign each message within a Sequence a message number (defined  
231 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers  
232 MUST be assigned in the same order in which messages are sent by the Application Source.
- 233 • Within every Acknowledgement Message it issues, the RM Destination MUST include one or more  
234 AcknowledgementRange child elements that contain, in their collective ranges, the message  
235 number of every message accepted by the RM Destination. The RM Destination MUST exclude, in  
236 the AcknowledgementRange elements, the message numbers of any messages it has not  
237 accepted. If no messages have been received the RM Destination MUST return None instead of an  
238 AcknowledgementRange (s). The RM Destination MAY transmit a Nack for a specific message  
239 or messages in stead of an AcknowledgementRange (s).
- 240 • While the Sequence is not closed or terminated, the RM Source SHOULD retransmit  
241 unacknowledged messages.

## 242 2.4 Example Message Exchange

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

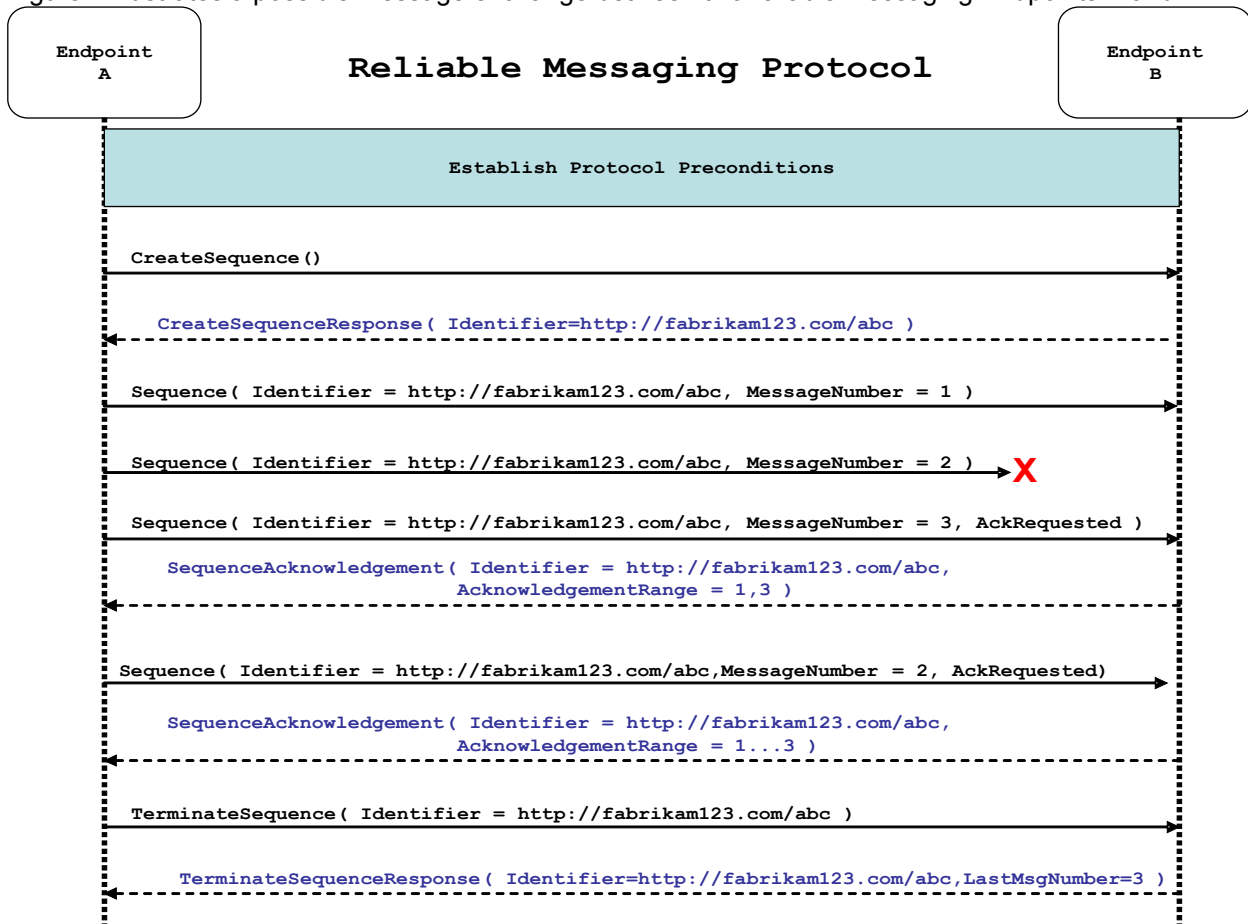


Figure 2: The WS-ReliableMessaging Protocol



- 244 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,  
245 and establishing trust.
- 246 2. The RM Source requests creation of a new Sequence.
- 247 3. The RM Destination creates a new Sequence and returns its unique identifier.
- 248 4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1.  
249 In the figure above, the RM Source sends 3 messages in the Sequence.
- 250 5. The 2<sup>nd</sup> message in the Sequence is lost in transit.
- 251 6. The 3<sup>rd</sup> message is the last in this Sequence and the RM Source includes an `AckRequested`  
252 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 253 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the  
254 RM Source's `AckRequested` header.
- 255 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new  
256 message from the perspective of the underlying transport, but it has the same Sequence Identifier  
257 and MessageNumber so the RM Destination can recognize it as a duplicate of the earlier message,  
258 in case the original and retransmitted messages are both Received. The RM Source includes an  
259 `AckRequested` header in the retransmitted message so the RM Destination will expedite an  
260 acknowledgement.
- 261 9. The RM Destination Receives the second transmission of the message with MessageNumber 2  
262 and acknowledges receipt of message numbers 1, 2, and 3.
- 263 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to the  
264 RM Destination indicating that the Sequence is completed. The `TerminateSequence` message  
265 indicates that message number 3 was the last message in the Sequence. The RM Destination then  
266 reclaims any resources associated with the Sequence.
- 267 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source will  
268 not be sending any more messages. The RM Destination sends a `TerminateSequenceResponse`  
269 message to the RM Source and reclaims any resources associated with the Sequence.

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

282 Now that the basic model has been outlined, the details of the elements used in this protocol are now  
283 provided in Section 3.

## 284 **3 RM Protocol Elements**

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

### 287 **3.1 Considerations on the Use of Extensibility Points**

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

### 292 **3.2 Considerations on the Use of "Piggy-Backing"**

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

### 302 **3.3 Composition with WS-Addressing**

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

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

```
311 http://docs.oasis-open.org/ws-rx/wsrn/200608/CreateSequence
```

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

```
314 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
```

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

```
317 http://docs.oasis-open.org/ws-rx/wsrn/200608/AckRequested
```

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

## 320 3.4 Sequence Creation

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

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

328 The following exemplar defines the `CreateSequence` syntax:

```
329 <wsrm:CreateSequence ...>
330   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
331   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
332   <wsrm:Offer ...>
333     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
334     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
335     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
336     <wsrm:IncompleteSequenceBehavior>
337       wsrn:IncompleteSequenceBehaviorType
338     </wsrm:IncompleteSequenceBehavior> ?
339     ...
340   </wsrm:Offer> ?
341   ...
342 </wsrm:CreateSequence>
```

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

344 `/wsrm:CreateSequence`

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

349 `/wsrm:CreateSequence/wsrm:AcksTo`

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

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

359 `/wsrm:CreateSequence/wsrm:Expires`

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

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

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

367 `/wsmr:CreateSequence/wsmr:Offer`

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

370 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier`

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

373 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Identifier/@{any}`

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

376 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Endpoint`

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

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

386 The Offer of an Endpoint containing the "http://www.w3.org/2005/08/addressing/anonymous" IRI as its  
387 address is problematic due to the inability of a source to connect to this address and retry  
388 unacknowledged messages (as described in Section 2.3). In the absence of an extension that addresses  
389 this issue, an RM Destination MUST NOT accept (via the  
390 `/wsmr:CreateSequenceResponse/wsmr:Accept` element described below) an Offer that contains  
391 the "http://www.w3.org/2005/08/addressing/anonymous" IRI as its address.

392 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Expires`

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

396 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Expires/@{any}`

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

399 `/wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior`

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

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

406 A value of “DiscardFollowingFirstGap” indicates that messages in the Sequence beyond the first gap  
407 MUST be discarded when there are one or more gaps in the final `SequenceAcknowledgement`.

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

410 `/wsrm:CreateSequence/wsrm:Offer/{any}`

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

413 `/wsrm:CreateSequence/wsrm:Offer/@{any}`

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

416 `/wsrm:CreateSequence/{any}`

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

419 `/wsrm:CreateSequence/@{any}`

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

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

426 The following exemplar defines the `CreateSequenceResponse` syntax:

```
427 <wsrm:CreateSequenceResponse ...>  
428   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
429   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
430   <wsrm:IncompleteSequenceBehavior>  
431     wsrm:IncompleteSequenceBehaviorType  
432   </wsrm:IncompleteSequenceBehavior> ?  
433   <wsrm:Accept ...>  
434     <wsrm:AcksTo wsa:EndpointReferenceType </wsrm:AcksTo>  
435     ...  
436   </wsrm:Accept> ?  
437   ...  
438 </wsrm:CreateSequenceResponse>
```

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

440 `/wsrm:CreateSequenceResponse`

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

444 `/wsrm:CreateSequenceResponse/wsrm:Identifier`

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

448 `/wsrm:CreateSequenceResponse/wsrm:Identifier/@{any}`

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

451 /wsmr:CreateSequenceResponse/wsmr:Expires

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

460 /wsmr:CreateSequenceResponse/wsmr:Expires/@{any}

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

463 /wsmr:CreateSequenceResponse/wsmr:IncompleteSequenceBehavior

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

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

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

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

474 /wsmr:CreateSequenceResponse/wsmr:Accept

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

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

480 /wsmr:CreateSequenceResponse/wsmr:Accept/wsmr:AcksTo

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

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

489 /wsmr:CreateSequenceResponse/wsmr:Accept/{any}

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

492 `/wsmr:CreateSequenceResponse/wsmr:Accept/@{any}`

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

495 `/wsmr:CreateSequenceResponse/{any}`

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

498 `/wsmr:CreateSequenceResponse/@{any}`

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

### 501 **3.5 Closing A Sequence**

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

507 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of  
508 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept  
509 any new messages for the specified Sequence, other than those already accepted at the time the  
510 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or  
511 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST  
512 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`  
513 element) header block on any messages associated with the Sequence destined to the RM Source,  
514 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM  
515 Source.

516 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM  
517 Source SHOULD include the `LastMsgNumber` element in any `CloseSequence` messages it sends. The  
518 RM Destination can use this information, for example, to implement the behavior indicated by  
519 `/wsmr:CreateSequenceResponse/wsmr:IncompleteSequenceBehavior`. The value of the  
520 `LastMsgNumber` element MUST be the same in all the `CloseSequence` messages for the closing  
521 Sequence.

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

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

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

535 The following exemplar defines the `CloseSequence` syntax:

```
536 <wsrm:CloseSequence ...>  
537   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
538   <wsrm:LastMsgNumber> wsrm:MessageNumberType </wsrm:LastMsgNumber> ?  
539   ...  
540 </wsrm:CloseSequence>
```

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

542 `/wsrm:CloseSequence`

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

546 `/wsrm:CloseSequence/wsrm:Identifier`

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

550 `/wsrm:CloseSequence/wsrm:LastMessageNumber`

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

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

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

557 `/wsrm:CloseSequence/{any}`

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

560 `/wsrm:CloseSequence@{any}`

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

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

565 The following exemplar defines the `CloseSequenceResponse` syntax:

```
566 <wsrm:CloseSequenceResponse ...>  
567   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
568   ...  
569 </wsrm:CloseSequenceResponse>
```

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

571 `/wsrm:CloseSequenceResponse`



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

574 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

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

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

581 `/wsrm:CloseSequenceResponse/{any}`

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

584 `/wsrm:CloseSequenceResponse@{any}`

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

### 587 **3.6 Sequence Termination**

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

595 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM  
596 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.  
597 The RM Destination can use this information, for example, to implement the behavior indicated by  
598 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the  
599 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the  
600 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same  
601 Sequence.

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

606 The following exemplar defines the `TerminateSequence` syntax:

```
607 <wsrm:TerminateSequence ...>  
608   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
609   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
610   ...  
611 </wsrm:TerminateSequence>
```

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

613 `/wsrm:TerminateSequence`

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

619 This element MAY also be sent by the RM Destination to indicate that it has unilaterally terminated the  
620 Sequence. Upon sending this message the RM Destination MUST NOT accept any additional messages  
621 (with the exception of the corresponding `TerminateSequenceResponse`) for this Sequence. Upon  
622 receipt of a `TerminateSequence` the RM Source MUST NOT send any additional messages (with the  
623 exception of the corresponding `TerminateSequenceResponse`) for this Sequence.

624 `/wsrm:TerminateSequence/wsrm:Identifier`

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

628 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

629 The RM Source SHOULD include this element in any `TerminateSequence` message it sends. The  
630 `LastMsgNumber` element specifies the highest assigned message number of all the Sequence Traffic  
631 Messages for the closing Sequence.

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

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

635 `/wsrm:TerminateSequence/{any}`

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

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

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

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

643 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
644 <wsrm:TerminateSequenceResponse ...>  
645   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
646   ...  
647 </wsrm:TerminateSequenceResponse>
```

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

649 `/wsrm:TerminateSequenceResponse`

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

653 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

657 `/wsrm:TerminateSequenceResponse/{any}`

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

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

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

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

## 657 3.7 Sequences

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

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

657 A following exemplar defines its syntax:

```
657 <wsrm:Sequence ...>  
657   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
657   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
657   ...  
657 </wsrm:Sequence>
```

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

657 `/wsrm:Sequence`

657 This protocol element associates the message in which it is contained with a previously established RM  
658 Sequence. It contains the Sequence's unique identifier and the containing message's ordinal position  
659 within that Sequence. The RM Destination MUST understand the `Sequence` header block. The RM  
660 Source MUST assign a `mustUnderstand` attribute with a value 1/true (from the namespace  
661 corresponding to the version of SOAP to which the `Sequence` SOAP header block is bound) to the  
662 `Sequence` header block element.

657 `/wsrm:Sequence/wsrm:Identifier`

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

657 /wsmr:Sequence/wsmr:Identifier/{any}

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

657 /wsmr:Sequence/wsmr:MessageNumber

657 The RM Source **MUST** include this element within any Sequence headers it creates. This element is of  
658 type `MessageNumberType`. It represents the ordinal position of the message within a Sequence.  
659 Sequence message numbers start at 1 and monotonically increase by 1 throughout the Sequence. See  
660 Section 4.5 for Message Number Rollover fault.

657 /wsmr:Sequence/{any}

657 This is an extensibility mechanism to allow different types of information, based on a schema, to be  
658 passed.

657 /wsmr:Sequence/{any}

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

657 The following example illustrates a Sequence header block.

```
657 <wsmr:Sequence>  
657   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
657   <wsmr:MessageNumber>10</wsmr:MessageNumber>  
657 </wsmr:Sequence>
```

## 657 3.8 Request Acknowledgement

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

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

657 An RM Destination that Receives a message that contains an `AckRequested` header block **MUST** send  
658 a message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference  
659 (see Section 3.4) for a known Sequence or else generate an `UnknownSequence` fault. It is  
660 **RECOMMENDED** that the RM Destination return a `AcknowledgementRange` or `None` element instead  
661 of a `Nack` element (see Section 3.9).

657 The following exemplar defines its syntax:

```
657 <wsmr:AckRequested ...>  
657   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
657   ...  
657 </wsmr:AckRequested>
```

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

657 /wsmr:AckRequested

657 This element requests an Acknowledgement for the identified Sequence.

657 /wsmr:AckRequested/wsmr:Identifier  
657 An RM Source that includes an `AckRequested` header block in a SOAP envelope MUST include this  
658 element in that header block. The RM Source MUST set the value of this element to the absolute URI,  
659 (conformant with RFC3986), that uniquely identifies the Sequence to which the request applies.

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

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

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

### 657 3.9 Sequence Acknowledgement

657 The RM Destination informs the RM Source of successful message receipt using a  
658 `SequenceAcknowledgement` header block. Acknowledgements can be explicitly requested using the  
659 `AckRequested` directive (see Section 3.8).

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

657 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the  
658 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing  
659 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any  
660 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted  
661 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received  
662 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`  
663 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`  
664 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination  
665 SHOULD respond on the protocol binding-specific back-channel provided by the Received message  
666 containing the `AckRequested` header block.

657 The following exemplar defines its syntax:

```
657 <wsmr:SequenceAcknowledgement ...>  
657   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
657   [ [ [ <wsmr:AcknowledgementRange ...  
657     Upper="wsmr:MessageNumberType"  
657     Lower="wsmr:MessageNumberType" /> +  
657     | <wsmr:None/> ]  
657     <wsmr:Final/> ? ]  
657   | <wsmr:Nack> wsmr:MessageNumberType </wsmr:Nack> + ]  
657   ...
```

657 `</wsrm:SequenceAcknowledgement>`

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

657 `/wsrm:SequenceAcknowledgement`

657 This element contains the Sequence Acknowledgement information.

657 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

657 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

657 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper`

657 The RM Destination MUST set the value of this attribute equal to the message number of the highest  
658 contiguous message in a Sequence range accepted by the RM Destination.

657 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower`

657 The RM Destination MUST set the value of this attribute equal to the message number of the lowest  
658 contiguous message in a Sequence range accepted by the RM Destination.

657 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}`

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

657 `/wsrm:SequenceAcknowledgement/wsrm:None`

657 The RM Destination MUST include this element within a `SequenceAcknowledgement` header block if  
658 the RM Destination has not accepted any messages for the specified Sequence. The RM Destination  
659 MUST NOT include this element if a sibling `AcknowledgementRange` or `Nack` element is also present  
660 as a child of the `SequenceAcknowledgement`.

657 `/wsrm:SequenceAcknowledgement/wsrm:Final`

657 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. This  
658 element indicates that the RM Destination is not receiving new messages for the specified Sequence. The  
659 RM Source can be assured that the ranges of messages acknowledged by this  
660 `SequenceAcknowledgement` header block will not change in the future. The RM Destination MUST  
661 include this element when the Sequence is closed. The RM Destination MUST NOT include this element  
662 when sending a `Nack`; it can only be used when sending `AcknowledgementRange` elements or a `None`.

657 /wsmr:SequenceAcknowledgement/wsmr:Nack

657 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If  
658 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing  
659 the `MessageNumber` of an unreceived message in a `Sequence`. The RM Destination MUST NOT include  
660 a `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of  
661 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the  
662 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`  
663 containing a `Nack` for a message that it has previously acknowledged within a  
664 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing  
665 a `Nack` for a message that has previously been acknowledged within a `AcknowledgementRange`.

657 /wsmr:SequenceAcknowledgement/{any}

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

657 /wsmr:SequenceAcknowledgement/@{any}

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

657 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
657 <wsmr:SequenceAcknowledgement>  
657   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
657   <wsmr:AcknowledgementRange Upper="10" Lower="1"/>  
657 </wsmr:SequenceAcknowledgement>
```

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

```
657 <wsmr:SequenceAcknowledgement>  
657   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
657   <wsmr:AcknowledgementRange Upper="2" Lower="1"/>  
657   <wsmr:AcknowledgementRange Upper="6" Lower="4"/>  
657   <wsmr:AcknowledgementRange Upper="10" Lower="8"/>  
657 </wsmr:SequenceAcknowledgement>
```

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

```
657 <wsmr:SequenceAcknowledgement>  
657   <wsmr:Identifier>http://example.com/abc</wsmr:Identifier>  
657   <wsmr:Nack>3</wsmr:Nack>  
657 </wsmr:SequenceAcknowledgement>
```

## 657 4 Faults

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

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

```
657 http://docs.oasis-open.org/ws-rx/wsrn/200608/fault
```

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

657 The definitions of faults use the following properties:

657 [Code] The fault code.

657 [Subcode] The fault subcode.

657 [Reason] The English language reason element.

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

657 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or  
658 "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

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

```
657 <S:Envelope>  
657   <S:Header>  
657     <wsa:Action>  
657       http://docs.oasis-open.org/ws-rx/wsrn/200608/fault  
657     </wsa:Action>  
657     <!-- Headers elided for brevity. -->  
657   </S:Header>  
657   <S:Body>  
657     <S:Fault>  
657       <S:Code>  
657         <S:Value> [Code] </S:Value>  
657         <S:Subcode>  
657           <S:Value> [Subcode] </S:Value>  
657         </S:Subcode>  
657       </S:Code>  
657       <S:Reason>  
657         <S:Text xml:lang="en"> [Reason] </S:Text>  
657       </S:Reason>  
657     <S:Detail>
```



```

657     [Detail]
657     ...
657     </S:Detail>
657     </S:Fault>
657     </S:Body>
657 </S:Envelope>

```

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

```

657 <S11:Envelope>
657   <S11:Header>
657     <wsrm:SequenceFault>
657       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
657       <wsrm:Detail> [Detail] </wsrm:Detail>
657       ...
657     </wsrm:SequenceFault>
657     <!-- Headers elided for brevity. -->
657   </S11:Header>
657   <S11:Body>
657     <S11:Fault>
657       <faultcode> [Code] </faultcode>
657       <faultstring> [Reason] </faultstring>
657     </S11:Fault>
657   </S11:Body>
657 </S11:Envelope>

```

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

```

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

```

## 657 4.1 SequenceFault Element

657 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during  
658 the reliable messaging specific processing of a message belonging to a Sequence. WS-  
659 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP  
660 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in  
661 conjunction with the SOAP 1.2 binding.

657 The following exemplar defines its syntax:

```

657 <wsrm:SequenceFault ...>
657   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
657   <wsrm:Detail> ... </wsrm:Detail> ?
657   ...
657 </wsrm:SequenceFault>

```

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

657 `/wsrm:SequenceFault`

657 This is the element containing Sequence information for WS-ReliableMessaging

657 /wsm:SequenceFault/wsm:FaultCode  
 657 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a  
 658 qualified name from the set of fault [Subcodes] defined below.

657 /wsm:SequenceFault/wsm:Detail  
 657 This element, if present, carries application specific error information related to the fault being described.

657 /wsm:SequenceFault/wsm:Detail/{any}  
 657 The application specific error information related to the fault being described.

657 /wsm:SequenceFault/wsm:Detail/@{any}  
 657 The application specific error information related to the fault being described.

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

657 /wsm:SequenceFault/@{any}  
 657 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the  
 658 element.

## 657 4.2 Sequence Terminated

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

657 Properties:

657 [Code] Sender or Receiver

657 [Subcode] wsm:SequenceTerminated

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

657 [Detail]

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

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

## 657 4.3 Unknown Sequence

657 Properties:

657 [Code] Sender

657 [Subcode] wsm:UnknownSequence

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

657 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source or RM Destination.	In response to a message containing an unknown or terminated Sequence identifier.	None.	MUST terminate the Sequence if not otherwise terminated.

#### 657 4.4 Invalid Acknowledgement

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

659 [Code] Sender

660 [Subcode] wsrn:InvalidAcknowledgement

661 [Reason] The SequenceAcknowledgement violates the cumulative Acknowledgement invariant.

662 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source.	In response to a SequenceAcknowledgement that violate the invariants stated in 2.3 or any of the requirements in 3.9 about valid combinations of AckRange, Nack and None in a single SequenceAcknowledgement element or with respect to already Received such elements.	Unspecified.	Unspecified.

#### 664 4.5 Message Number Rollover

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

665 Properties:

666 [Code] Sender

667 [Subcode] wsrn:MessageNumberRollover

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

669 [Detail]

```
669 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
670 <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.

## 669 4.6 Create Sequence Refused

669 Properties:

669 [Code] Sender or Receiver

669 [Subcode] wsrm:CreateSequenceRefused

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

669 [Detail]

```
669 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.

## 669 4.7 Sequence Closed

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

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

669 Properties:

669 [Code] Sender

669 [Subcode] wsrm:SequenceClosed

669 [Reason] The Sequence is closed and can not accept new messages.

669 [Detail]

```
<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.

## 669 4.8 WSRM Required

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

669 Properties:

669 [Code] Sender

669 [Subcode] wsrm:WSRMRequired

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

669 [Detail]

```
xs:any
```

## 669 **5 Security Threats and Countermeasures**

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

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

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

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

### 669 **5.1 Threats and Countermeasures**

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

#### 669 **5.1.1 Integrity Threats**

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

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

##### 669 **5.1.1.1 Countermeasures**

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

## 669 **5.1.2 Resource Consumption Threats**

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

### 669 **5.1.2.1 Countermeasures**

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

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

## 669 **5.1.3 Sequence Spoofing Threats**

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

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

### 669 **5.1.3.1 Sequence Hijacking**

669 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject  
670 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those  
671 messages.

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

### 669 **5.1.3.2 Countermeasures**

669 There are a number of countermeasures against sequence spoofing threats. The technique advocated by  
670 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

669 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that  
670 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence  
671 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that  
672 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.  
673 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a  
674 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

669 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and  
670 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its  
671 sequence peer it MUST be able to identify and authenticate the entity that sent the  
672 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a  
673 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that  
674 message and, if necessary, correlate this identity with the sequence peer identity established at sequence  
675 creation time.

## 669 **5.2 Security Solutions and Technologies**

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

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

### 669 **5.2.1 Transport Layer Security**

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

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

#### 669 **5.2.1.1 Model**

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

- 669 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 669 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM  
670 Destination.



- 669 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an  
670 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a  
671 synchronous `CreateSequenceResponse` using the session established in (1).
- 669 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit  
670 any and all messages or faults that refer to that Sequence.
- 669 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established  
670 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous  
671 exchanges, the RM Destination uses the SSL/TLS session established in (1).

### 669 5.2.1.2 Countermeasure Implementation

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

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

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

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

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

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

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

## 669 **5.2.2 SOAP Message Security**

669 The mechanisms described in WS-Security may be used in various ways to implement the  
670 countermeasures described in the previous sections. This specification advocates using the protocol  
671 described by WS-SecureConversation [[SecureConversation](#)] (optionally in conjunction with WS-Trust  
672 [[Trust](#)]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component  
673 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

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

### 669 **5.2.2.1 Model**

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

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

### 669 **5.2.2.2 Countermeasure Implementation**

669 Without relying upon any authentication information, the per-message signatures provide the necessary  
670 integrity qualities to counter the threats described in Section 5.1.1.

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

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

669 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer  
670 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

669 the association between a Sequence and its protecting security context cannot always be established  
670 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and  
671 `CreateSequenceResponse` messages may be signed by more than one security context.

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

## 669 6 Securing Sequences

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

### 669 6.1 Securing Sequences Using WS-Security

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

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

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

```
669 <wsrm:CreateSequence ...>  
669   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
669   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
669   <wsrm:Offer ...>  
669     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
669     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
669     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
669     <wsrm:IncompleteSequenceBehavior>  
669       wsrml:IncompleteSequenceBehaviorType  
669     </wsrm:IncompleteSequenceBehavior> ?  
669     ...  
669   </wsrm:Offer> ?  
669   ...  
669   <wsse:SecurityTokenReference>  
669     ...  
669   </wsse:SecurityTokenReference> ?  
669   ...  
669 </wsrm:CreateSequence>
```

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

669 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

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

669 When a RM Source transmits a `CreateSequence` that has been extended to include a  
670 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and  
671 will conform to the requirements listed above. In order to achieve this, the RM Source SHOULD include

669 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This  
670 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source  
671 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands  
672 and conforms with the requirements listed above. Note that an RM Destination understanding this header  
673 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined  
674 in WS-Security still applies.

669 The following exemplar defines the `UsesSequenceSTR` syntax:

```
669 <wsm:UsesSequenceSTR ... />
```

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

669 `/wsm:UsesSequenceSTR`

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

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

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

```
669 <soap:Envelope ...>  
669   <soap:Header>  
669     ...  
669     <wsm:UsesSequenceSTR soap:mustUnderstand='true' />  
669     ...  
669   </soap:Header>  
669   <soap:Body>  
669     <wsm:CreateSequence>  
669       <wsm:AcksTo>  
669         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
669       </wsm:AcksTo>  
669       <wsse:SecurityTokenReference>  
669         ...  
669       </wsse:SecurityTokenReference>  
669     </wsm:CreateSequence>  
669   </soap:Body>  
669 </soap:Envelope>
```

## 669 6.2 Securing Sequences Using SSL/TLS

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

669 The following exemplar defines the `UsesSequenceSSL` syntax:

```
669 <wsm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

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

669 `/wsm:UsesSequenceSSL`

669 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to  
670 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was

669 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a  
670 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand  
671 and correctly implement the functionality described in Section 5.2.1 or else generate a  
672 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

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

## 669 **7 References**

### 669 **7.1 Normative**

#### 669 **[KEYWORDS]**

669 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,  
670 March 1997

669 <http://www.ietf.org/rfc/rfc2119.txt>

#### 669 **[WS-RM Policy]**

669 OASIS WS-RX Technical Committee Draft, "[Web Services ReliableMessaging Policy Assertion\( WS-RM  
670 Policy\)](#)" October 2006

669 <http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-wd-11.pdf>

#### 669 **[SOAP 1.1]**

669 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

669 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

#### 669 **[SOAP 1.2]**

669 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

669 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

#### 669 **[URI]**

669 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,  
670 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

669 <http://ietf.org/rfc/rfc3986>

#### 669 **[UUID]**

669 P. Leach, M. Mealling, R. Salz, "[A Universally Unique IDentifier \(UUID\) URN Namespace](#)," RFC 4122,  
670 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

669 <http://www.ietf.org/rfc/rfc4122.txt>

#### 669 **[XML]**

669 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#)", September 2006.

669 <http://www.w3.org/TR/REC-xml/>

#### 669 **[XML-ns]**

669 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

669 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

#### 669 **[XML-Schema Part1]**

669 W3C Recommendation, "[XML Schema Part 1: Structures](#)," October 2004.

669 <http://www.w3.org/TR/xmlschema-1/>

669 **[XML-Schema Part2]**

669 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.

669 <http://www.w3.org/TR/xmlschema-2/>

669 **[XPath 1.0]**

669 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.

669 <http://www.w3.org/TR/xpath>

669 **[WSDL 1.1]**

669 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.

669 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

669 **[WS-Addressing]**

669 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.

669 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>

669 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.

669 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

669 **7.2 Non-Normative**

669 **[BSP 1.0]**

669 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006

669 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>

669 **[RDDL 2.0]**

669 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004

669 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>

669 **[RFC 2617]**

669 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP  
670 Authentication: Basic and Digest Access Authentication," June 1999.

669 <http://www.ietf.org/rfc/rfc2617.txt>

669 **[RFC 4346]**

669 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.

669 <http://www.ietf.org/rfc/rfc4346.txt>

669 **[WS-Policy]**

669 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.

669 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>

669 **[WS-PolicyAttachment]**

669 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.

669 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-  
670 20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)



669 **[WS-Security]**

669 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:  
670 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

669 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

669 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:  
670 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

669 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

669 **[RTTM]**

669 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May  
670 1992.

669 <http://www.rfc-editor.org/rfc/rfc1323.txt>

669 **[SecurityPolicy]**

669 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

669 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

669 **[SecureConversation]**

669 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February  
670 2005.

669 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

669 **[Trust]**

669 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

669 <http://schemas.xmlsoap.org/ws/2005/02/trust>

## 669 Appendix A. Schema

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

669 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-200608.xsd>

669 The following copy is provided for reference.

```
670 <?xml version="1.0" encoding="UTF-8"?>
671 <!--
672 OASIS takes no position regarding the validity or scope of any intellectual
673 property or other rights that might be claimed to pertain to the
674 implementation or use of the technology described in this document or the
675 extent to which any license under such rights might or might not be available;
676 neither does it represent that it has made any effort to identify any such
677 rights. Information on OASIS's procedures with respect to rights in OASIS
678 specifications can be found at the OASIS website. Copies of claims of rights
679 made available for publication and any assurances of licenses to be made
680 available, or the result of an attempt made to obtain a general license or
681 permission for the use of such proprietary rights by implementors or users of
682 this specification, can be obtained from the OASIS Executive Director.
683 OASIS invites any interested party to bring to its attention any copyrights,
684 patents or patent applications, or other proprietary rights which may cover
685 technology that may be required to implement this specification. Please
686 address the information to the OASIS Executive Director.
687 Copyright © OASIS Open 2002-2006. All Rights Reserved.
688 This document and translations of it may be copied and furnished to others,
689 and derivative works that comment on or otherwise explain it or assist in its
690 implementation may be prepared, copied, published and distributed, in whole or
691 in part, without restriction of any kind, provided that the above copyright
692 notice and this paragraph are included on all such copies and derivative
693 works. However, this document itself does not be modified in any way, such as
694 by removing the copyright notice or references to OASIS, except as needed for
695 the purpose of developing OASIS specifications, in which case the procedures
696 for copyrights defined in the OASIS Intellectual Property Rights document must
697 be followed, or as required to translate it into languages other than English.
698 The limited permissions granted above are perpetual and will not be revoked by
699 OASIS or its successors or assigns.
700 This document and the information contained herein is provided on an "AS IS"
701 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
702 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
703 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
704 FOR A PARTICULAR PURPOSE.
705 -->
706 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
707 xmlns:wsa="http://www.w3.org/2005/08/addressing"
708 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
709 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
710 elementFormDefault="qualified" attributeFormDefault="unqualified">
711   <xs:import namespace="http://www.w3.org/2005/08/addressing"
712   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
713   <!-- Protocol Elements -->
714   <xs:complexType name="SequenceType">
715     <xs:sequence>
716       <xs:element ref="wsrm:Identifier"/>
717       <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
718       <xs:any namespace="##other" processContents="lax" minOccurs="0"
719 maxOccurs="unbounded"/>
720     </xs:sequence>
```

```

721     <xs:anyAttribute namespace="##other" processContents="lax"/>
722 </xs:complexType>
723 <xs:element name="Sequence" type="wsrm:SequenceType"/>
724 <xs:element name="SequenceAcknowledgement">
725   <xs:complexType>
726     <xs:sequence>
727       <xs:element ref="wsrm:Identifier"/>
728       <xs:choice>
729         <xs:sequence>
730           <xs:choice>
731             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
732               <xs:complexType>
733                 <xs:sequence/>
734                 <xs:attribute name="Upper" type="xs:unsignedLong"
735 use="required"/>
736                 <xs:attribute name="Lower" type="xs:unsignedLong"
737 use="required"/>
738               <xs:anyAttribute namespace="##other" processContents="lax"/>
739             </xs:complexType>
740           </xs:element>
741           <xs:element name="None">
742             <xs:complexType>
743               <xs:sequence/>
744             </xs:complexType>
745           </xs:element>
746         </xs:choice>
747         <xs:element name="Final" minOccurs="0">
748           <xs:complexType>
749             <xs:sequence/>
750           </xs:complexType>
751         </xs:element>
752       </xs:sequence>
753       <xs:element name="Nack" type="xs:unsignedLong"
754 maxOccurs="unbounded"/>
755     </xs:choice>
756     <xs:any namespace="##other" processContents="lax" minOccurs="0"
757 maxOccurs="unbounded"/>
758   </xs:sequence>
759   <xs:anyAttribute namespace="##other" processContents="lax"/>
760 </xs:complexType>
761 </xs:element>
762 <xs:complexType name="AckRequestedType">
763   <xs:sequence>
764     <xs:element ref="wsrm:Identifier"/>
765     <xs:any namespace="##other" processContents="lax" minOccurs="0"
766 maxOccurs="unbounded"/>
767   </xs:sequence>
768   <xs:anyAttribute namespace="##other" processContents="lax"/>
769 </xs:complexType>
770 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
771 <xs:element name="Identifier">
772   <xs:complexType>
773     <xs:annotation>
774       <xs:documentation>
775         This type is for elements whose [children] is an anyURI and can have
776 arbitrary attributes.
777       </xs:documentation>
778     </xs:annotation>
779     <xs:simpleContent>
780       <xs:extension base="xs:anyURI">
781         <xs:anyAttribute namespace="##other" processContents="lax"/>
782       </xs:extension>
783     </xs:simpleContent>

```

```

721     </xs:complexType>
722 </xs:element>
723 <xs:element name="Address">
724   <xs:complexType>
725     <xs:simpleContent>
726       <xs:extension base="xs:anyURI">
727         <xs:anyAttribute namespace="##other" processContents="lax"/>
728       </xs:extension>
729     </xs:simpleContent>
730   </xs:complexType>
731 </xs:element>
732 <xs:simpleType name="MessageNumberType">
733   <xs:restriction base="xs:unsignedLong">
734     <xs:minInclusive value="1"/>
735     <xs:maxInclusive value="9223372036854775807"/>
736   </xs:restriction>
737 </xs:simpleType>
738 <!-- Fault Container and Codes -->
739 <xs:simpleType name="FaultCodes">
740   <xs:restriction base="xs:QName">
741     <xs:enumeration value="wsrm:SequenceTerminated"/>
742     <xs:enumeration value="wsrm:UnknownSequence"/>
743     <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
744     <xs:enumeration value="wsrm:MessageNumberRollover"/>
745     <xs:enumeration value="wsrm:CreateSequenceRefused"/>
746     <xs:enumeration value="wsrm:SequenceClosed"/>
747     <xs:enumeration value="wsrm:WSRMRequired"/>
748     <xs:enumeration value="wsrm:UnsupportedSelection"/>
749   </xs:restriction>
750 </xs:simpleType>
751 <xs:complexType name="SequenceFaultType">
752   <xs:sequence>
753     <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
754     <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
755     <xs:any namespace="##other" processContents="lax" minOccurs="0"
756 maxOccurs="unbounded"/>
757   </xs:sequence>
758   <xs:anyAttribute namespace="##other" processContents="lax"/>
759 </xs:complexType>
760 <xs:complexType name="DetailType">
761   <xs:sequence>
762     <xs:any namespace="##other" processContents="lax" minOccurs="0"
763 maxOccurs="unbounded"/>
764   </xs:sequence>
765   <xs:anyAttribute namespace="##other" processContents="lax"/>
766 </xs:complexType>
767 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
768 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
769 <xs:element name="CreateSequenceResponse"
770 type="wsrm:CreateSequenceResponseType"/>
771 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
772 <xs:element name="CloseSequenceResponse"
773 type="wsrm:CloseSequenceResponseType"/>
774 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
775 <xs:element name="TerminateSequenceResponse"
776 type="wsrm:TerminateSequenceResponseType"/>
777 <xs:complexType name="CreateSequenceType">
778   <xs:sequence>
779     <xs:element ref="wsrm:AcksTo"/>
780     <xs:element ref="wsrm:Expires" minOccurs="0"/>
781     <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
782     <xs:any namespace="##other" processContents="lax" minOccurs="0"
783 maxOccurs="unbounded"/>

```

```

721     <xs:annotation>
722         <xs:documentation>
723             It is the authors intent that this extensibility be used to
724 transfer a Security Token Reference as defined in WS-Security.
725         </xs:documentation>
726     </xs:annotation>
727 </xs:any>
728 </xs:sequence>
729 <xs:anyAttribute namespace="##other" processContents="lax" />
730 </xs:complexType>
731 <xs:complexType name="CreateSequenceResponseType">
732     <xs:sequence>
733         <xs:element ref="wsrm:Identifier" />
734         <xs:element ref="wsrm:Expires" minOccurs="0" />
735         <xs:element name="IncompleteSequenceBehavior"
736 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0" />
737         <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0" />
738         <xs:any namespace="##other" processContents="lax" minOccurs="0"
739 maxOccurs="unbounded" />
740     </xs:sequence>
741     <xs:anyAttribute namespace="##other" processContents="lax" />
742 </xs:complexType>
743 <xs:complexType name="CloseSequenceType">
744     <xs:sequence>
745         <xs:element ref="wsrm:Identifier" />
721         <xs:element ref="wsrm:MessageNumberType" />
722         <xs:any namespace="##other" processContents="lax" minOccurs="0"
723 maxOccurs="unbounded" />
724     </xs:sequence>
725     <xs:anyAttribute namespace="##other" processContents="lax" />
726 </xs:complexType>
727 <xs:complexType name="CloseSequenceResponseType">
728     <xs:sequence>
729         <xs:element ref="wsrm:Identifier" />
730         <xs:any namespace="##other" processContents="lax" minOccurs="0"
731 maxOccurs="unbounded" />
732     </xs:sequence>
733     <xs:anyAttribute namespace="##other" processContents="lax" />
734 </xs:complexType>
735 <xs:complexType name="TerminateSequenceType">
736     <xs:sequence>
737         <xs:element ref="wsrm:Identifier" />
721         <xs:element ref="wsrm:MessageNumberType" minOccurs="0" />
722         <xs:any namespace="##other" processContents="lax" minOccurs="0"
723 maxOccurs="unbounded" />
724     </xs:sequence>
725     <xs:anyAttribute namespace="##other" processContents="lax" />
726 </xs:complexType>
727 <xs:complexType name="TerminateSequenceResponseType">
728     <xs:sequence>
729         <xs:element ref="wsrm:Identifier" />
730         <xs:any namespace="##other" processContents="lax" minOccurs="0"
731 maxOccurs="unbounded" />
732     </xs:sequence>
733     <xs:anyAttribute namespace="##other" processContents="lax" />
734 </xs:complexType>
735 <xs:element name="AcksTo" type="wsa:EndpointReferenceType" />
736 <xs:complexType name="OfferType">
737     <xs:sequence>
738         <xs:element ref="wsrm:Identifier" />
739         <xs:element name="Endpoint" type="wsa:EndpointReferenceType" />
740         <xs:element ref="wsrm:Expires" minOccurs="0" />
741         <xs:element name="IncompleteSequenceBehavior"

```

```

721 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
722   <xs:any namespace="##other" processContents="lax" minOccurs="0"
723 maxOccurs="unbounded"/>
724   </xs:sequence>
725   <xs:anyAttribute namespace="##other" processContents="lax"/>
726 </xs:complexType>
727 <xs:complexType name="AcceptType">
728   <xs:sequence>
729     <xs:element ref="wsrm:AcksTo"/>
730     <xs:any namespace="##other" processContents="lax" minOccurs="0"
731 maxOccurs="unbounded"/>
732   </xs:sequence>
733   <xs:anyAttribute namespace="##other" processContents="lax"/>
734 </xs:complexType>
735 <xs:element name="Expires">
736   <xs:complexType>
737     <xs:simpleContent>
738       <xs:extension base="xs:duration">
739         <xs:anyAttribute namespace="##other" processContents="lax"/>
740       </xs:extension>
741     </xs:simpleContent>
742   </xs:complexType>
743 </xs:element>
744 <xs:simpleType name="IncompleteSequenceBehaviorType">
745   <xs:restriction base="xs:string">
746     <xs:enumeration value="DiscardEntireSequence"/>
747     <xs:enumeration value="DiscardFollowingFirstGap"/>
748     <xs:enumeration value="NoDiscard"/>
749   </xs:restriction>
750 </xs:simpleType>
751 <xs:element name="UsesSequenceSTR">
752   <xs:complexType>
753     <xs:sequence/>
754     <xs:anyAttribute namespace="##other" processContents="lax"/>
755   </xs:complexType>
756 </xs:element>
757 <xs:element name="UsesSequenceSSL">
758   <xs:complexType>
759     <xs:sequence/>
760     <xs:anyAttribute namespace="##other" processContents="lax"/>
761   </xs:complexType>
762 </xs:element>
763 <xs:element name="UnsupportedElement">
764   <xs:simpleType>
765     <xs:restriction base="xs:QName"/>
766   </xs:simpleType>
767 </xs:element>
768 </xs:schema>

```

## 759 Appendix B. WSDL

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

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

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

759 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsd/wsrn-1.1-wsd-200608.wsd>

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

```
761 <?xml version="1.0" encoding="utf-8"?>
762 <!--
763 OASIS takes no position regarding the validity or scope of any intellectual
764 property or other rights that might be claimed to pertain to the
765 implementation or use of the technology described in this document or the
766 extent to which any license under such rights might or might not be available;
767 neither does it represent that it has made any effort to identify any such
768 rights. Information on OASIS's procedures with respect to rights in OASIS
769 specifications can be found at the OASIS website. Copies of claims of rights
770 made available for publication and any assurances of licenses to be made
771 available, or the result of an attempt made to obtain a general license or
772 permission for the use of such proprietary rights by implementors or users of
773 this specification, can be obtained from the OASIS Executive Director.
774 OASIS invites any interested party to bring to its attention any copyrights,
775 patents or patent applications, or other proprietary rights which may cover
776 technology that may be required to implement this specification. Please
777 address the information to the OASIS Executive Director.
778 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
779 This document and translations of it may be copied and furnished to others,
780 and derivative works that comment on or otherwise explain it or assist in its
781 implementation may be prepared, copied, published and distributed, in whole or
782 in part, without restriction of any kind, provided that the above copyright
783 notice and this paragraph are included on all such copies and derivative
784 works. However, this document itself does not be modified in any way, such as
785 by removing the copyright notice or references to OASIS, except as needed for
786 the purpose of developing OASIS specifications, in which case the procedures
787 for copyrights defined in the OASIS Intellectual Property Rights document must
788 be followed, or as required to translate it into languages other than English.
789 The limited permissions granted above are perpetual and will not be revoked by
790 OASIS or its successors or assigns.
791 This document and the information contained herein is provided on an "AS IS"
792 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
793 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
794 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
795 FOR A PARTICULAR PURPOSE.
796 -->
797 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
798 xmlns:xs="http://www.w3.org/2001/XMLSchema"
799 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
800 open.org/ws-rx/wsrn/200608" xmlns:tns="http://docs.oasis-open.org/ws-
801 rx/wsrn/200608/wsd" targetNamespace="http://docs.oasis-open.org/ws-
802 rx/wsrn/200608/wsd">
803 <wsdl:types>
```

```

804     <xs:schema>
805         <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsr/200608"
806 schemaLocation="http://docs.oasis-open.org/ws-rx/wsr/200608/wsr-1.1-schema-
807 200608.xsd"/>
808     </xs:schema>
809 </wsdl:types>

810     <wsdl:message name="CreateSequence">
811         <wsdl:part name="create" element="rm:CreateSequence"/>
812     </wsdl:message>
813     <wsdl:message name="CreateSequenceResponse">
814         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
815     </wsdl:message>
816     <wsdl:message name="CloseSequence">
817         <wsdl:part name="close" element="rm:CloseSequence"/>
818     </wsdl:message>
819     <wsdl:message name="CloseSequenceResponse">
820         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
821     </wsdl:message>
822     <wsdl:message name="TerminateSequence">
823         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
824     </wsdl:message>
825     <wsdl:message name="TerminateSequenceResponse">
826         <wsdl:part name="terminateResponse"
827 element="rm:TerminateSequenceResponse"/>
828     </wsdl:message>

829     <wsdl:portType name="SequenceAbstractPortType">
830         <wsdl:operation name="CreateSequence">
831             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
832 open.org/ws-rx/wsr/200608/CreateSequence"/>
833             <wsdl:output message="tns:CreateSequenceResponse"
834 wsaw:Action="http://docs.oasis-open.org/ws-
835 rx/wsr/200608/CreateSequenceResponse"/>
836         </wsdl:operation>
837         <wsdl:operation name="CloseSequence">
838             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
839 open.org/ws-rx/wsr/200608/CloseSequence"/>
840             <wsdl:output message="tns:CloseSequenceResponse"
841 wsaw:Action="http://docs.oasis-open.org/ws-
842 rx/wsr/200608/CloseSequenceResponse"/>
843         </wsdl:operation>
844         <wsdl:operation name="TerminateSequence">
845             <wsdl:input message="tns:TerminateSequence"
846 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsr/200608/TerminateSequence"/>
847             <wsdl:output message="tns:TerminateSequenceResponse"
848 wsaw:Action="http://docs.oasis-open.org/ws-
849 rx/wsr/200608/TerminateSequenceResponse"/>
850         </wsdl:operation>
851     </wsdl:portType>

852 </wsdl:definitions>

```



## 804 Appendix C. Message Examples

### 804 Appendix C.1 Create Sequence

#### 804 Create Sequence

```
804 <?xml version="1.0" encoding="UTF-8"?>
804 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
804 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
804 xmlns:wsa="http://www.w3.org/2005/08/addressing">
804   <S:Header>
804     <wsa:MessageID>
804       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
804     </wsa:MessageID>
804     <wsa:To>http://example.com/serviceB/123</wsa:To>
804     <wsa:Action>http://docs.oasis-open.org/ws-
805 rx/wsmr/200608/CreateSequence</wsa:Action>
804     <wsa:ReplyTo>
804       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
804     </wsa:ReplyTo>
804   </S:Header>
804   <S:Body>
804     <wsmr:CreateSequence>
804       <wsmr:AcksTo>
804         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
804       </wsmr:AcksTo>
804     </wsmr:CreateSequence>
804   </S:Body>
804 </S:Envelope>
```

#### 804 Create Sequence Response

```
804 <?xml version="1.0" encoding="UTF-8"?>
804 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
805 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
806 xmlns:wsa="http://www.w3.org/2005/08/addressing">
804   <S:Header>
804     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
804     <wsa:RelatesTo>
804       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
804     </wsa:RelatesTo>
804     <wsa:Action>
804       http://docs.oasis-open.org/ws-rx/wsmr/200608/CreateSequenceResponse
804     </wsa:Action>
804   </S:Header>
804   <S:Body>
804     <wsmr:CreateSequenceResponse>
804       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
804     </wsmr:CreateSequenceResponse>
804   </S:Body>
804 </S:Envelope>
```

### 804 Appendix C.2 Initial Transmission

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

804 **Message 1**

```
804 <?xml version="1.0" encoding="UTF-8"?>
804 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
804 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
804 xmlns:wsa="http://www.w3.org/2005/08/addressing">
804   <S:Header>
804     <wsa:MessageID>
804       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
804     </wsa:MessageID>
804     <wsa:To>http://example.com/serviceB/123</wsa:To>
804     <wsa:From>
804       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
804     </wsa:From>
804     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
804     <wsmr:Sequence>
804       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
804       <wsmr:MessageNumber>1</wsmr:MessageNumber>
804     </wsmr:Sequence>
804   </S:Header>
804   <S:Body>
804     <!-- Some Application Data -->
804   </S:Body>
804 </S:Envelope>
```

804 **Message 2**

```
804 <?xml version="1.0" encoding="UTF-8"?>
804 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
804 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
804 xmlns:wsa="http://www.w3.org/2005/08/addressing">
804   <S:Header>
804     <wsa:MessageID>
804       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
804     </wsa:MessageID>
804     <wsa:To>http://example.com/serviceB/123</wsa:To>
804     <wsa:From>
804       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
804     </wsa:From>
804     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
804     <wsmr:Sequence>
804       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
804       <wsmr:MessageNumber>2</wsmr:MessageNumber>
804     </wsmr:Sequence>
804   </S:Header>
804   <S:Body>
804     <!-- Some Application Data -->
804   </S:Body>
804 </S:Envelope>
```

804 **Message 3**

```
804 <?xml version="1.0" encoding="UTF-8"?>
804 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
804 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
804 xmlns:wsa="http://www.w3.org/2005/08/addressing">
804   <S:Header>
804     <wsa:MessageID>
804       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
804     </wsa:MessageID>
805     <wsa:To>http://example.com/serviceB/123</wsa:To>
806     <wsa:From>
807     <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
```

```

808 </wsa:From>
808 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
808 <wsrm:Sequence>
808 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
808 <wsrm:MessageNumber>3</wsrm:MessageNumber>
808 </wsrm:Sequence>
808 <wsrm:AckRequested>
808 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
808 </wsrm:AckRequested>
808 </S:Header>
808 <S:Body>
808 <!-- Some Application Data -->
808 </S:Body>
808 </S:Envelope>

```

### 808 Appendix C.3 First Acknowledgement

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

```

808 <?xml version="1.0" encoding="UTF-8"?>
808 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
808 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
808 xmlns:wsa="http://www.w3.org/2005/08/addressing">
808 <S:Header>
808 <wsa:MessageID>
808 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
808 </wsa:MessageID>
808 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
808 <wsa:From>
808 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
808 </wsa:From>
808 <wsa:Action>
808 http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
808 </wsa:Action>
808 <wsrm:SequenceAcknowledgement>
808 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
808 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
808 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
808 </wsrm:SequenceAcknowledgement>
808 </S:Header>
808 <S:Body/>
808 </S:Envelope>

```

### 808 Appendix C.4 Retransmission

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

```

808 <?xml version="1.0" encoding="UTF-8"?>
808 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
808 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
808 xmlns:wsa="http://www.w3.org/2005/08/addressing">
808 <S:Header>
808 <wsa:MessageID>
808 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
808 </wsa:MessageID>
808 <wsa:To>http://example.com/serviceB/123</wsa:To>
808 <wsa:From>
808 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
808 </wsa:From>

```

```

808 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
808 <wsrm:Sequence>
808 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
808 <wsrm:MessageNumber>2</wsrm:MessageNumber>
808 </wsrm:Sequence>
808 <wsrm:AckRequested>
808 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
808 </wsrm:AckRequested>
808 </S:Header>
809 <S:Body>
810 <!-- Some Application Data -->
811 </S:Body>
811 </S:Envelope>

```

## 811 Appendix C.5 Termination

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

```

811 <?xml version="1.0" encoding="UTF-8"?>
811 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
811 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
811 xmlns:wsa="http://www.w3.org/2005/08/addressing">
811 <S:Header>
811 <wsa:MessageID>
811 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
811 </wsa:MessageID>
811 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
811 <wsa:From>
811 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
811 </wsa:From>
811 <wsa:Action>
811 http://docs.oasis-open.org/ws-rx/wsrn/200608/SequenceAcknowledgement
811 </wsa:Action>
811 <wsrm:SequenceAcknowledgement>
811 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
811 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
811 </wsrm:SequenceAcknowledgement>
811 </S:Header>
811 <S:Body/>
811 </S:Envelope>

```

### 811 Terminate Sequence

```

811 <?xml version="1.0" encoding="UTF-8"?>
811 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
811 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
811 xmlns:wsa="http://www.w3.org/2005/08/addressing">
811 <S:Header>
811 <wsa:MessageID>
812 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
813 </wsa:MessageID>
813 <wsa:To>http://example.com/serviceB/123</wsa:To>
813 <wsa:Action>
813 http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence
813 </wsa:Action>
813 <wsa:From>
813 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
813 </wsa:From>
813 </S:Header>
813 <S:Body>
813 <wsrm:TerminateSequence>

```

```
813     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
813     <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
813     </wsrm:TerminateSequence>
813 </S:Body>
813 </S:Envelope>
```

### 813 Terminate Sequence Response

```
813 <?xml version="1.0" encoding="UTF-8"?>
813 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
813 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
813 xmlns:wsa="http://www.w3.org/2005/08/addressing">
813   <S:Header>
813     <wsa:MessageID>
813       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
813     </wsa:MessageID>
813     <wsa:To>http://example.com/serviceA/789</wsa:To>
813     <wsa:Action>
813       http://docs.oasis-open.org/ws-rx/wsr/200608/TerminateSequenceResponse
813     </wsa:Action>
813     <wsa:RelatesTo>
813       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
813     </wsa:RelatesTo>
813     <wsa:From>
813       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
813     </wsa:From>
813   </S:Header>
813   <S:Body>
813     <wsrm:TerminateSequenceResponse>
813       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
813     </wsrm:TerminateSequenceResponse>
813   </S:Body>
813 </S:Envelope>
```

## 813 Appendix D. State Tables

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

813 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

813 Legend:

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

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

813 Where:

- 813 ● Event Name: indicates the name of the event. Event Names surrounded by “<>” are optional as  
814 described by the specification.
- 813 ● [source]: indicates the source of the event; one of:
  - 813 ● [msg] a Received message
  - 813 ● [int]: an internal event such as the firing of a timer
  - 813 ● [app]: the application
  - 813 ● [unspec]: the source is unspecified

813 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}

813 Where:

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

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

813 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
<b>Create Sequence</b> [unspec] {3.4}	Xmit Create Sequence [Creating] {3.4}	N/A	N/A	N/A	N/A	N/A
<b>Create Sequence Response</b> [msg] {3.4}		Process Create Sequence Response [Created] {3.4}				
<b>Create Sequence Refused Fault</b> [msg] {3.4}		No action [None] {4.6}				
<b>Send message</b> [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [Same] {2}	N/A	N/A
<b>Retransmit of un-ack'd message</b> [int]	N/A	N/A	Xmit message [Same] {2.4}	Xmit message [Same] {2.4}	N/A	N/A
<b>SeqAck (non-final)</b> [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}
<b>Nack</b> [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]
<b>Message Number Rollover Fault</b> [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Rollover]	No action [Same]	No action [Same]	No action [Same]
<b>CloseSequence</b> [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}
<b>&lt;Close Sequence&gt;</b> [int] {3.5}	N/A		Xmit Close Sequence [Closing] {3.5}	N/A	N/A	N/A
<b>Close Sequence Response</b> [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
<b>SeqAck (final)</b> [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]
<b>Sequence Closed Fault</b> [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]
<b>Unknown Sequence Fault</b> [msg] {4.3}			Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
<b>Sequence Terminated Fault</b> [msg] {4.2}	N/A		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
<b>TerminateSequence</b> [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}
<b>Terminate Sequence</b> [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	N/A
<b>Terminate Sequence Response</b> [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}				Terminate Sequence [None] {3.6}
<b>Expires exceeded</b> [int]	N/A	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}
<b>Invalid Acknowledgement</b> [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

813 Table 2 RM Destination Sequence State Transition Table

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



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

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

## 813 **Appendix E. Acknowledgments**

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

813 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),  
814 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),  
815 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),  
816 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don  
817 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),  
818 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony  
819 Storey(IBM).

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

813 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen  
814 Brown(Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM),  
815 Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott  
816 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal  
817 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter  
818 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish  
819 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

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

813 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben  
814 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd  
815 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul  
816 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques  
817 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),  
818 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair  
819 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),  
820 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul  
821 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt  
822 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff  
823 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku  
824 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert  
825 Pilz(BEA), Martin Raeppele(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom  
826 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von  
827 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçinalp(SAP), Nobuyuki  
828 Yamamoto(Hitachi).

## Appendix F. Revision History

Rev	Date	By Whom	What
wd-01	2005-07-07	Christopher Ferris	Initial version created based on submission by the authors.
ws-02	2005-07-21	Doug Davis	I011 (PT0S) added
wd-02	2005-08-16	Anish Karmarkar	Trivial editorial changes
ws-03	2005-09-15	Doug Davis	I019 and i028 (CloseSeq) added
wd-05	2005-09-26	Gilbert Pilz	i005 (Source resend of nacks messages when ack already received) added.
wd-05	2005-09-27	Doug Davis	i027 (InOrder delivery assurance spanning multiple sequences) added
wd-05	2005-09-27	Doug Davis	i020 (Semantics of "At most once" Delivery Assurance) added
wd-05	2005-09-27	Doug Davis	i034 (Fault while processing a piggy-backed RM header) added
wd-05	2005-09-27	Doug Davis	i033 (Processing model of NACKs) added
wd-05	2005-09-27	Doug Davis	i031 (AckRequested schema inconsistency) added
wd-05	2005-09-27	Doug Davis	i025 (SeqAck/None) added
wd-05	2005-09-27	Doug Davis	i029 (Remove dependency on WS-Security) added
wd-05	2005-09-27	Doug Davis	i039 (What does 'have a mU attribute' mean) added
wd-05	2005-09-27	Doug Davis	i040 (Change 'optiona'/'required' to 'OPTIONAL'/'REQUIRED') added
wd-05	2005-09-30	Anish Karmarkar	i017 (Change NS to <a href="http://docs.oasis-open.org/wsrn/200510/">http://docs.oasis-open.org/wsrn/200510/</a> )
wd-05	2005-09-30	Anish Karmarkar	i045 (Include SecureConversation as a reference and move it to non-normative citation)
wd-05	2005-09-30	Anish Karmarkar	i046 (change the type of wsrn:FaultCode element)
wd-06	2005-11-02	Gilbert Pilz	Start wd-06 by changing title page from cd-01.
wd-06	2005-11-03	Gilbert Pilz	i047 (Reorder spec sections)
wd-07	2005-11-17	Gilbert Pilz	Start wd-07
wd-07	2005-11-28	Doug Davis	i071 – except for period in Appendix headings
wd-07	2005-11-28	Doug Davis	i10
wd-07	2005-11-28	Doug Davis	i030
wd-07	2005-11-28	Doug Davis	i037
wd-07	2005-11-28	Doug Davis	i038
wd-07	2005-11-28	Doug Davis	i041
wd-07	2005-11-28	Doug Davis	i043
wd-07	2005-11-28	Doug Davis	i044

Rev	Date	By Whom	What
wd-07	2005-11-28	Doug Davis	i048
wd-07	2005-11-28	Doug Davis	i051
wd-07	2005-11-28	Doug Davis	i053
wd-07	2005-11-28	Doug Davis	i059
wd-07	2005-11-28	Doug Davis	i062
wd-07	2005-11-28	Doug Davis	i063
wd-07	2005-11-28	Doug Davis	i065
wd-07	2005-11-28	Doug Davis	i067
wd-07	2005-11-28	Doug Davis	i068
wd-07	2005-11-28	Doug Davis	i069
wd-07	2005-11-28	Doug Davis	Fix bulleted list (#2) in section 2.3
wd-07	2005-11-29	Gilbert Pilz	i074 (Use of [tcShortName] in artifact locations namespaces, etc)
wd-07	2005-11-29	Gilbert Pilz	i071 – Fixed styles and formatting for TOC. Fixed styles of the appendix headings.
wd-07	2005-11-30	Doug Davis	Removed dup definition of "Receive"
wd-07	2005-11-30	Gilbert Pilz	Fixed lost formatting from heading for Namespace section. Fixed style of text body elements to match OASIS example documents. Fixed tables to match OASIS example documents.
wd-07	2005-12-01	Gilbert Pilz	Updated fix for i074 to eliminate trailing '/'. Added corresponding text around action IRI composition.
wd-07	2005-12-01	Gilbert Pilz	Use non-fixed fields for date values on both title page and body footers.
wd-07	2005-12-01	Doug Davis	Alphabetize the glossary
wd-07	2005-12-02	Doug Davis	i064
wd-07	2005-12-02	Doug Davis	i066
wd-08	2005-12-15	Doug Davis	Add back in RM Source to glossary
wd-08	2005-12-15	Steve Winkler	Doug added Steve's editorial nits
wd-08	2005-12-21	Doug Davis	i050
wd-08	2005-12-21	Doug Davis	i081
wd-08	2005-12-21	Doug Davis	i080 – but i050 negates the need for any changes
wd-08	2005-12-21	Doug Davis	i079
wd-08	2005-12-21	Doug Davis	i076 – didn't add text about "replies" since the RMD to RMS sequence could be used for any message not just replies
wd-08	2005-12-21	Umit Yalcinalp	Action Su03: removed wsse from Table 1
wd-08	2005-12-21	Umit Yalcinalp	i057 per Sunnyvale F2F 2005, Cleaned up some formatting errors in contributors
wd-08	2005-12-27	Doug Davis	i060
wd-08	2005-12-27	Gilbert Pilz	Moved schema and WSDL files to their own artifacts. Converted source document to

Rev	Date	By Whom	What
			OpenDocument Text format. Changed line numbers to be a single style.
wd-08	2005-12-28	Anish Karmarkar	Included a section link to c:\temp\wsrm-1.1-schema-200510.xsd and to c:\temp\wsrm-1.1-wsdl-200510.wsdl
wd-08	2006-01-04	Gilbert Pilz	Fixed formatting for included sections.
wd-08	2006-01-05	Gilbert Pilz	Created links for unused references. Fixed exemplars for CloseSequence and CloseSequenceResponse.
wd-09	2006-01-11	Doug Davis	Minor tweaks to text/typos.
wd-10	2006-01-23	Doug Davis	Accept all changes from wd-09  Make some minor editorial tweaks from Marc's comments.
wd-10	2006-02-14	Doug Davis	Issue 082 resolution
wd-10	2006-02-14	Doug Davis	Issue 083 resolution
wd-10	2006-02-14	Doug Davis	Issue 085 resolution
wd-10	2006-02-14	Doug Davis	Issues 086, 087 resolutions  Defined MessageNumberType
wd-10	2006-02-15	Doug Davis	Issue 078 resolution
wd-10	2006-02-15	Doug Davis	Issue 094 resolution
wd-10	2006-02-15	Doug Davis	Issue 095 resolution
wd-10	2006-02-15	Gilbert Pilz	Issue 088 – added namespace URI link to namespace URI; added text explaining that this URI could be dereferenced to produce the RDDDL doc; added non-normative reference to RDDDL 2.0
wd-10	2006-02-17	Anish Karmarkar	Namespace changed to 200602 for both WSDL and XSD docs.
wd-10	2006-02-17	Anish Karmarkar	Issue i087 as it applies to WSRM spec.
wd-10	2006-02-17	Anish Karmarkar	Added titles and minor text for state table (issue i058).
wd-11	2006-02-22	Doug Davis	Accept all changes for new WD  Minor typos fixed
wd-11	2006-02-23	Doug Davis	s'/close'/close/g – per Marc Goodner  Added first ref to [URI] – per Marc G again
wd-11	2006-02-27	Doug Davis	Issue i061 applied
wd-11	2006-02-28	Doug Davis	Fixed typo around the use of "above" and "below"
wd-11	2006-03-01	Doug Davis	Minor typos found by Marc Goodner
wd-11	2006-03-02	Doug Davis	Minor typos found by Matt Lovett
wd-11	2006-03-08	Doug Davis	Issue 091 applied
wd-11	2006-03-08	Doug Davis	Issue 092 applied
wd-11	2006-03-08	Doug Davis	Issue 100 applied

Rev	Date	By Whom	What
wd-12	2006-03-20	Doug Davis	Added space in "SOAP1.x" – PaulCotton
wd-12	2006-04-11	Doug Davis	Issue 007 applied
wd-12	2006-04-11	Doug Davis	Issue 090 applied
wd-12	2006-04-11	Doug Davis	Issue 098 applied
wd-12	2006-04-11	Doug Davis	Issue 099 applied
wd-12	2006-04-11	Doug Davis	Issue 101 applied
wd-12	2006-04-11	Doug Davis	Issue 103 applied
wd-12	2006-04-11	Doug Davis	Issue 104 applied
wd-12	2006-04-11	Doug Davis	Issue 105 applied
wd-12	2006-04-11	Doug Davis	Issue 107 applied
wd-12	2006-04-11	Doug Davis	Issue 109 applied
wd-12	2006-04-11	Doug Davis	Issue 110 applied
wd-12	2006-04-12	Doug Davis	Used "generated" instead of "issue" or "send" when talking about faults.
wd-12	2006-04-24	Gilbert Pilz	Update references to WS-Addressing to the Proposed Recommendations; update WS-RM namespace to "200604".
wd-13	2006-05-08	Gilbert Pilz	i093 part 1; more work needed
wd-13	2006-05-10	Doug Davis	Issue 096 applied
wd-13	2006-05-26	Gilbert Pilz	i093 part 2; reflects decisions from 2006-05-25 meeting
wd-13	2006-05-28	Gilbert Pilz	Issue 106 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 118 applied
wd-13	2006-05-29	Gilbert Pilz	Issue 120 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 114 applied
wd-13	2006-05-30	Gilbert Pilz	Issue 116 applied
wd-14	2006-06-05	Gilbert Pilz	Accept all changes; bump WD number
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Marc Goodner
wd-14	2006-06-07	Doug Davis	Change a couple of period/sp/sp to period/sp
wd-14	2006-06-07	Doug Davis	Added a space in "URI]of" – per Marc Goodner
wd-14	2006-06-07	Doug Davis	Issue 131 applied
wd-14	2006-06-07	Doug Davis	Issue 132 applied
wd-14	2006-06-07	Doug Davis	Issue 119 applied
wd-14	2006-06-07	Doug Davis	Applied lots of minor edits from Doug Davis
wd-14	2006-06-07	Doug Davis	s/"none"/"full-uri"/ - per Marc Goodner
wd-14	2006-06-12	Doug Davis	Complete i106
wd-14	2006-06-12	Doug Davis	Issues 089 applied
wd-14	2006-06-12	Doug Davis	Fix for several RFC2119 keywords – per Anish
wd-15	2006-06-12	Doug Davis	Accept all changed, dump WD number
wd-15	2006-06-12	Doug Davis	Move WSDL after Schema
wd-15	2006-06-12	Doug Davis	Nits – remove tabs, extra [yyy]'s ...
wd-15	2006-06-14	Doug Davis	Remove extra "OPTIONAL"s – Matt Lovett

Rev	Date	By Whom	What
wd-15	2006-06-14	Doug Davis	Remove blank rows/columns from state table. Fix italics in state table
wd-15	2006-06-15	Doug Davis	Typo – section D was empty
wd-15	2006-06-16	Doug Davis	Issue 125 applied
wd-15	2006-06-16	Doug Davis	Issue 126 applied
wd-15	2006-06-16	Doug Davis	Issue 127 applied
wd-15	2006-06-16	Doug Davis	Issue 133 applied
wd-15	2006-06-16	Doug Davis	Issue 136 applied
wd-15	2006-06-16	Doug Davis	Issue 138 applied
wd-15	2006-06-16	Doug Davis	Issue 135 applied
wd-15	2006-06-20	Doug Davis	Added all TC members to the ack list
wd-15	2006-06-22	Doug Davis	Issue 129 applied
wd-15	2006-06-22	Doug Davis	Issue 130 applied
wd-15	2006-06-22	Doug Davis	Issue 137 applied
wd-15	2006-06-26	Doug Davis	Issue 111 applied
wd-15	2006-06-26	Doug Davis	Missed a part of issue 129
wd-15	2006-06-30	Doug Davis	Fixed a typo in schema
wd-15	2006-06-30	Doug Davis	Issue 141 applied
wd-15	2006-06-30	Doug Davis	Issue 142 applied
wd-15	2006-06-30	Doug Davis	Issue 148 applied
wd-15	2006-06-30	Doug Davis	Issue 149 applied
wd-15	2006-06-30	Doug Davis	Issue 150 applied
wd-15	2006-07-06	Doug Davis	Issue 121 applied
wd-15	2006-07-21	Doug Davis	Issue 139 applied
wd-15	2006-07-21	Doug Davis	Issue 144 applied
wd-15	2006-07-21	Doug Davis	Issue 147 applied
wd-15	2006-07-21	Doug Davis	Issues 122-124 applied
wd-15	2006-07-27	Doug Davis	Updated list of oasis TC members (i134)
wd-15	2006-07-27	Doug Davis	Issue 140 applied
wd-15	2006-07-27	Doug Davis	Issue 145 applied
wd-15	2006-07-27	Doug Davis	Issue 143 applied
wd-15	2006-07-28	Doug Davis	Lots of minor typos found by Matt L.
wd-15	2006-07-28	Doug Davis	Issue 113 applied
wd-15	2006-08-04	Doug Davis	Update old namespaces – found by PaulC
wd-15	2006-08-04	Doug Davis	Issue 150 applied
wd-15	2006-08-04	Doug Davis	Minor typos – found by PeterN
wd-15	2006-08-04	Doug Davis	Verify all [refs]
wd-15	2006-08-04	Doug Davis	Change namespace to 2006/08
wd-15	2006-08-04	Doug Davis	Issue 148 applied
wd-15	2006-08-07	Doug Davis	Add some new glossary terms – per GilP
cd-04	2006-08-10	Gilbert Pilz	Formatting changes for better HTML rendering.



Rev	Date	By Whom	What
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.
wd-16	2006-10-25	Doug Davis	Accept all changes, update to wd16
wd-16	2006-10-26	Doug Davis	PR002 applied
wd-16	2006-10-26	Doug Davis	PR003 applied
wd-16	2006-10-26	Doug Davis	PR004 applied
wd-16	2006-10-27	Doug Davis	PR005 applied
wd-16	2006-10-27	Doug Davis	PR006 applied
wd-16	2006-10-27	Doug Davis	PR024 applied
wd-16	2006-11-13	Doug Davis	PR010 applied
wd-16	2006-11-13	Doug Davis	PR011 applied (technically as part of PR004)
wd-16	2006-11-13	Doug Davis	PR016 applied
wd-16	2006-11-13	Doug Davis	PR032 applied
wd-16	2006-11-20	Doug Davis	PR025 applied
wd-16	2006-11-20	Doug Davis	PR023 applied
wd-16	2006-12-03	Doug Davis	PR036 applied
wd-16	2006-12-03	Doug Davis	PR017 applied
wd-16	2006-12-11	Doug Davis	PR012 applied (and PR013)
wd-16	2006-12-14	Doug Davis	PR033 applied – changed a 'return' to 'generate' when talking about a fault
wd-16	2007-01-04	Doug Davis	PR018 applied
wd-16	2007-01-05	Doug Davis	Moved MakeConnection to new spec
wd-16	2007-01-17	Doug Davis	PR026 applied
wd-16	2007-01-18	Doug Davis	PR021 applied
wd-16	2007-01-18	Doug Davis	PR022 applied
wd-16	2007-01-18	Doug Davis	Fixed a few typos (Doug,Gil)
wd-16	2007-01-18	Gilbert Pilz	PR007 applied

## 813 **Appendix G. Notices**

813 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
814 might be claimed to pertain to the implementation or use of the technology described in this document or  
815 the extent to which any license under such rights might or might not be available; neither does it represent  
816 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
817 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
818 available for publication and any assurances of licenses to be made available, or the result of an attempt  
819 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
820 users of this specification, can be obtained from the OASIS Executive Director.

821 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
822 other proprietary rights which may cover technology that may be required to implement this specification.  
823 Please address the information to the OASIS Executive Director.

821 Copyright (C) OASIS Open (2006). All Rights Reserved.

821 This document and translations of it may be copied and furnished to others, and derivative works that  
822 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
823 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
824 this paragraph are included on all such copies and derivative works. However, this document itself may  
825 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
826 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
827 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
828 into languages other than English.

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

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