



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

3 Working Draft 15, July 28, 2006

4 **Document identifier:**

5 wsrn-1.1-spec-wd-15

6 **Location:**

7 **Editors:**

8 Doug Davis, IBM <dug@us.ibm.com>

9 Anish Karmarkar, Oracle <Anish.Karmarkar@oracle.com>

10 Gilbert Pilz, BEA <gpilz@bea.com>

11 Steve Winkler, SAP <steve.winkler@sap.com>

12 Ümit Yalçınalp, SAP <umit.yalcinalp@sap.com>

13 **Contributors:**

14 TBD

15 **Abstract:**

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

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

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

30 **Status:**

31 This document is a work in progress and will be updated to reflect issues as they are resolved by the  
32 Web Services Reliable Exchange (WS-RX) Technical Committee.

## 33 **Table of Contents**

34	1 Introduction.....	4
35	1.1 Notational Conventions.....	4
36	1.2 Namespace.....	5
37	1.3 Compliance.....	5
38	2 Reliable Messaging Model.....	6
39	2.1 Glossary.....	6
40	2.2 Protocol Preconditions.....	7
41	2.3 Protocol Invariants.....	7
42	2.4 Example Message Exchange.....	7
43	3 RM Protocol Elements.....	10
44	3.1 Sequence Creation.....	10
45	3.2 Closing A Sequence.....	14
46	3.3 Sequence Termination.....	16
47	3.4 Sequences.....	17
48	3.5 Request Acknowledgement.....	18
49	3.6 Sequence Acknowledgement.....	19
50	3.7 MakeConnection.....	22
51	3.8 MessagePending.....	23
52	4 Faults.....	25
53	4.1 SequenceFault Element.....	26
54	4.2 Sequence Terminated.....	27
55	4.3 Unknown Sequence.....	27
56	4.4 Invalid Acknowledgement.....	28
57	4.5 Message Number Rollover.....	28
58	4.6 Create Sequence Refused.....	29
59	4.7 Sequence Closed.....	29
60	4.8 WSRM Required.....	30
61	4.9 Unsupported Selection .....	30
62	5 Security Threats and Countermeasures.....	32
63	5.1 Threats and Countermeasures.....	32
64	5.1.1 Integrity Threats.....	32
65	5.1.1.1 Countermeasures.....	32
66	5.1.2 Resource Consumption Threats.....	33
67	5.1.2.1 Countermeasures.....	33
68	5.1.3 Sequence Spoofing Threats.....	33
69	5.1.3.1 Sequence Hijacking.....	33
70	5.1.3.2 Countermeasures.....	33

71	5.2 Security Solutions and Technologies.....	34
72	5.2.1 Transport Layer Security.....	34
73	5.2.1.1 Model.....	34
74	5.2.1.2 Countermeasure Implementation.....	35
75	5.2.2 SOAP Message Security.....	36
76	5.2.2.1 Model.....	36
77	5.2.2.2 Countermeasure Implementation.....	36
78	6 Securing Sequences.....	38
79	6.1 Securing Sequences Using WS-Security.....	38
80	6.2 Securing Sequences Using SSL/TLS.....	39
81	7 References.....	40
82	7.1 Normative.....	40
83	7.2 Non-Normative.....	40
84	A. Schema.....	42
85	B. WSDL.....	48
86	C. Message Examples.....	51
87	C.1 Create Sequence.....	51
88	C.2 Initial Transmission.....	51
89	C.3 First Acknowledgement.....	53
90	C.4 Retransmission.....	53
91	C.5 Termination.....	54
92	D. State Tables.....	56
93	E. Acknowledgments.....	61
94	F. Revision History.....	62
95	G. Notices.....	67

# 96 1 Introduction

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

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

## 106 1.1 Notational Conventions

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

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

- 111 • The syntax appears as an XML instance, but values in italics indicate data types instead of values.
- 112 • Characters are appended to elements and attributes to indicate cardinality:
  - 113 ○ "?" (0 or 1)
  - 114 ○ "\*" (0 or more)
  - 115 ○ "+" (1 or more)
- 116 • The character "|" is used to indicate a choice between alternatives.
- 117 • The characters "[" and "]" are used to indicate that contained items are to be treated as a group  
118 with respect to cardinality or choice.
- 119 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attribute content  
120 specified in this document. Additional children elements and/or attributes MAY be added at the  
121 indicated extension points but they MUST NOT contradict the semantics of the parent and/or  
122 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 123 • XML namespace prefixes (See Section 1.2) are used to indicate the namespace of the element  
124 being defined.

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

- 128 • An element extensibility point is referred to using {any} in place of the element name. This  
129 indicates that any element name can be used, from any namespace other than the wsm:  
130 namespace.
- 131 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This  
132 indicates that any attribute name can be used, from any namespace other than the wsm:  
133 namespace.

## 134 1.2 Namespace

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

136 <http://docs.oasis-open.org/ws-rx/wsrn/200604>

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

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

141 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/200604">http://docs.oasis-open.org/ws-rx/wsrn/200604</a>
wsa	<a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</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>

142 The normative schema for WS-ReliableMessaging can be found at:

143 <http://docs.oasis-open.org/ws-rx/wsrn/200604/wsrn-1.1-schema-200604.xsd>

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

## 145 1.3 Compliance

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

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

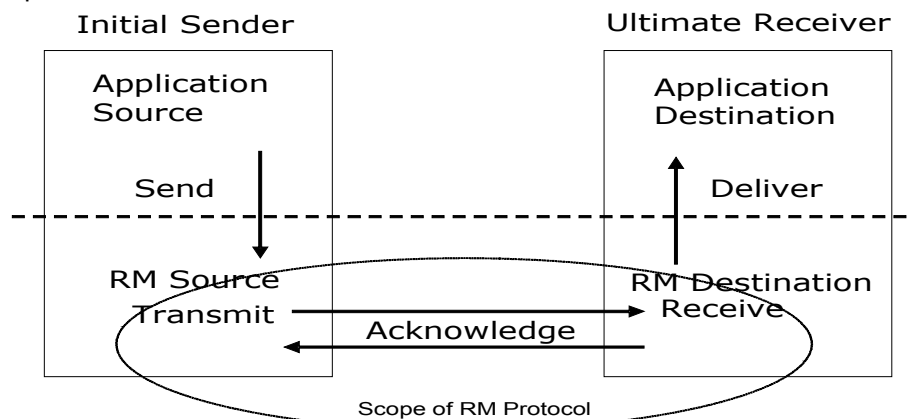
## 152 2 Reliable Messaging Model

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

155 The WS-ReliableMessaging specification defines an interoperable protocol that requires a Reliable  
156 Messaging (RM) Source and Reliable Messaging Destination to ensure that each message transmitted by  
157 the RM Source is accepted by an RM Destination, or barring acceptance, that an RM Source can, except  
158 in the most extreme circumstances, accurately determine the disposition of each message transmitted as  
159 perceived by the RM Destination, so as to resolve any in-doubt status regarding receipt of the messages  
160 transmitted. Note that this specification places no restriction on the scope of the RM Source or RM  
161 Destination entities. For example, either can span multiple WSDL Ports or endpoints.

162 The protocol enables the implementation of a broad range of reliability features which include ordered  
163 delivery, duplicate elimination, and guaranteed receipt. The protocol can also be implemented with a  
164 range of robustness characteristics ranging from in-memory persistence that is scoped to a single process  
165 lifetime, to replicated durable storage that is recoverable in all but the most extreme circumstances. It is  
166 expected that the endpoints will implement as many or as few of these reliability characteristics as  
167 necessary for the correct operation of the application using the protocol. Regardless of which of the  
168 reliability features is enabled, the wire protocol does not change.

169 Figure 1 below illustrates the entities and events in a simple reliable exchange of messages. First, the  
170 Application Source Sends a message for reliable transfer. The Reliable Messaging Source accepts the  
171 message and transmits it one or more times. After accepting the message, the RM Destination  
172 Acknowledges it. Finally, the RM Destination delivers the message to the Application Destination. The  
173 exact roles the entities play and the complete meaning of the events will be defined throughout this  
174 specification.



175 Figure 1: Reliable Messaging Model

### 176 2.1 Glossary

177 The following definitions are used throughout this specification:

178 **Accept:** The act of qualifying a message by the RM Destination such that it becomes eligible for delivery  
179 and acknowledgement.

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

182 **Application Destination:** The endpoint to which a message is Delivered.

183 **Application Source:** The endpoint that sends a message.  
184 **Deliver:** The act of transferring a message from the RM Destination to the Application Destination.  
185 **Endpoint:** As defined in the WS-Addressing specification [[WS-Addressing](#)]; a Web service endpoint is a  
186 (referenceable) entity, processor, or resource to which Web service messages can be addressed.  
187 Endpoint references convey the information needed to address a Web service endpoint.  
188 **Receive:** The act of reading a message from a network connection and accepting it.  
189 **RM Destination:** For any one reliably sent message the endpoint that receives the message.  
190 **RM Source:** The endpoint that transmits the message.  
191 **Send:** The act of submitting a message to the RM Source for reliable transfer.  
192 **Transmit:** The act of writing a message to a network connection.

## 193 **2.2 Protocol Preconditions**

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

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

## 203 **2.3 Protocol Invariants**

204 During the lifetime of a Sequence, two invariants are **REQUIRED** for correctness:

- 205 • The RM Source **MUST** assign each message within a Sequence a message number (defined  
206 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers  
207 **MUST** be assigned in the same order in which messages are sent by the Application Source.
- 208 • Within every acknowledgement it issues, the RM Destination **MUST** include one or more  
209 acknowledgement ranges that contain the message number of every message accepted by the RM  
210 Destination. The RM Destination **MUST** exclude the message numbers of any messages it has not  
211 accepted.

## 212 **2.4 Example Message Exchange**

213 Figure 2 illustrates a possible message exchange between two reliable messaging endpoints A and B.

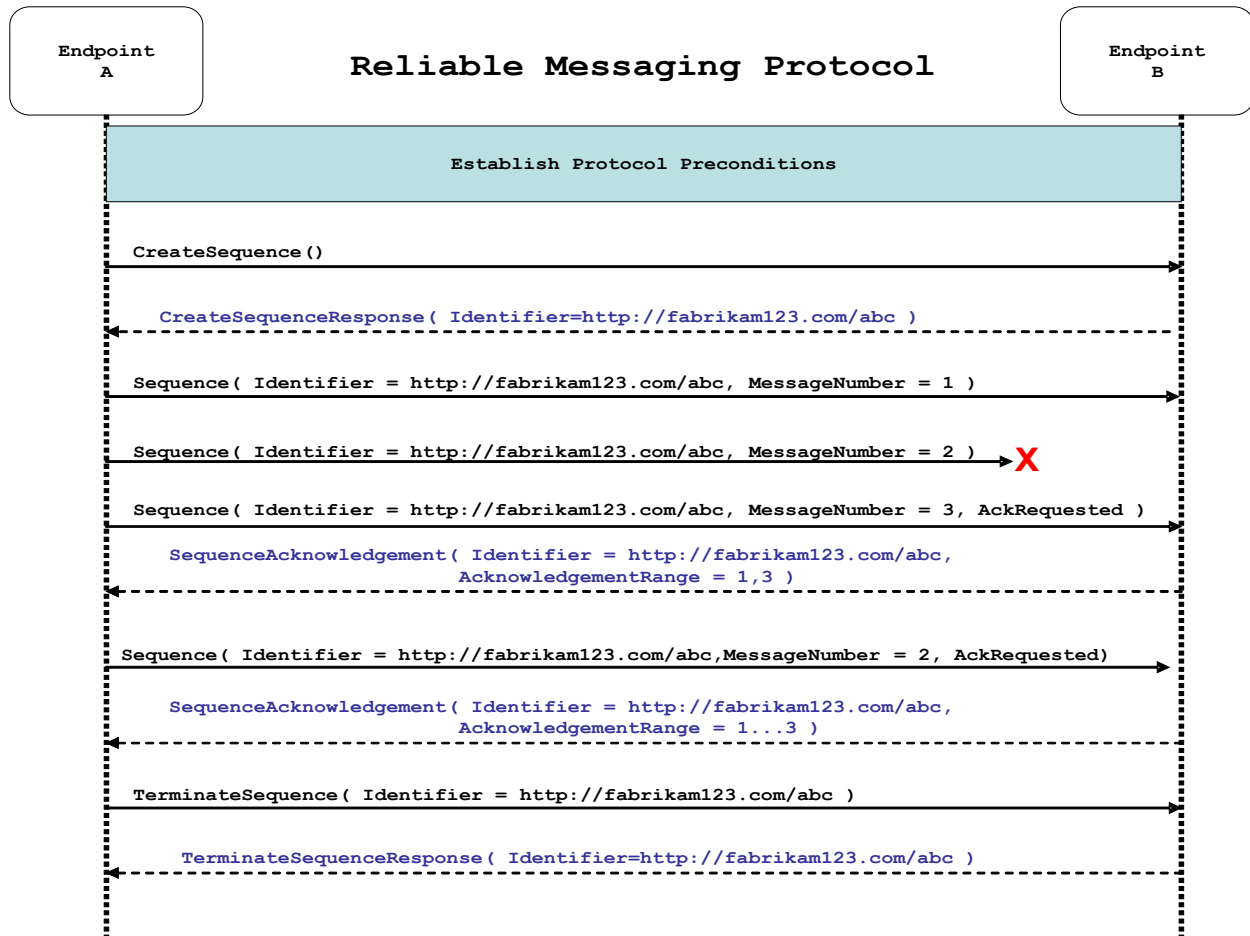


Figure 2: The WS-ReliableMessaging Protocol

- 214 1. The protocol preconditions are established. These include policy exchange, endpoint resolution,  
215 and establishing trust.
- 216 2. The RM Source requests creation of a new Sequence.
- 217 3. The RM Destination creates a new Sequence and returns its unique identifier.
- 218 4. The RM Source begins transmitting messages in the Sequence beginning with MessageNumber 1.  
219 In the figure above, the RM Source sends 3 messages in the Sequence.
- 220 5. The 2<sup>nd</sup> message in the Sequence is lost in transit.
- 221 6. The 3<sup>rd</sup> message is the last in this Sequence and the RM Source includes an `AckRequested`  
222 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 223 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the  
224 RM Source's `AckRequested` header.
- 225 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new  
226 message from the perspective of the underlying transport, but it has the same Sequence Identifier  
227 and MessageNumber so the RM Destination can recognize it as a duplicate of the earlier message,  
228 in case the original and retransmitted messages are both received. The RM Source includes an  
229 `AckRequested` header in the retransmitted message so the RM Destination will expedite an  
230 acknowledgement.



231 9. The RM Destination receives the second transmission of the message with MessageNumber 2 and  
232 acknowledges receipt of message numbers 1, 2, and 3.

233 10. The RM Source receives this acknowledgement and sends a TerminateSequence message to the  
234 RM Destination indicating that the Sequence is completed and reclaims any resources associated  
235 with the Sequence.

236 11. The RM Destination receives the TerminateSequence message indicating that the RM Source will  
237 not be sending any more messages. The RM Destination sends a TerminateSequenceResponse  
238 message to the RM Source and reclaims any resources associated with the Sequence.

239 The RM Source will expect to receive acknowledgements from the RM Destination during the course of a  
240 message exchange at occasions described in Section 3 below. Should an acknowledgement not be  
241 received in a timely fashion, the RM Source MUST re-transmit the message since either the message or  
242 the associated acknowledgement might have been lost. Since the nature and dynamic characteristics of  
243 the underlying transport and potential intermediaries are unknown in the general case, the timing of re-  
244 transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been  
245 demonstrated to cause transport or intermediary flooding which are counterproductive to the intention of  
246 providing a reliable exchange of messages. Consequently, implementers are encouraged to utilize  
247 adaptive mechanisms that dynamically adjust re-transmission time and the back-off intervals that are  
248 appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP  
249 transports, a mechanism similar to that described as RTTM in RFC 1323 [RTTM] SHOULD be  
250 considered.

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

## 253 3 RM Protocol Elements

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

258 Some RM header blocks may be added to messages that happen to be targeted to the same endpoint to  
259 which those headers are to be sent (a concept often referred to as "piggy-backing"), thus saving the  
260 overhead of an additional message exchange. Reference parameters MUST be considered when  
261 determining whether two EPRs are targeted to the same endpoint.

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

- 264 1. When an endpoint generates a message that carries an RM protocol element, that is defined in  
265 section 3 below, in the body of a SOAP envelope that endpoint MUST include in that envelope a  
266 `wsa:Action` SOAP header block whose value is an IRI that is a concatenation of the WS-RM  
267 namespace URI, followed by a '/', followed by the value of the local name of the child element of  
268 the SOAP body. For example, for a Sequence creation request message as described in section  
269 3.1 below, the value of the `wsa:Action` IRI would be:

```
270 http://docs.oasis-open.org/ws-rx/wsrn/200602/CreateSequence
```

- 271 2. When an endpoint generates a `SequenceAcknowledgement` message that has no element  
272 content in the SOAP body, then the value of the `wsa:Action` IRI MUST be:

```
273 http://docs.oasis-open.org/ws-rx/wsrn/200602/SequenceAcknowledgement
```

- 274 3. When an endpoint generates a `AckRequested` message that has no element content in the  
275 SOAP body, then the value of the `wsa:Action` IRI MUST be:

```
276 http://docs.oasis-open.org/ws-rx/wsrn/200602/AckRequested
```

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

### 279 3.1 Sequence Creation

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

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

287 The following exemplar defines the `CreateSequence` syntax:

```
288 <wsrm:CreateSequence ...>  
289   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>  
290   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
291   <wsrm:Offer ...>  
292     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
293     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>  
294     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
295     <wsrm:IncompleteSequenceBehavior>  
296       wsrm:IncompleteSequenceBehaviorType  
297     </wsrm:IncompleteSequenceBehavior> ?
```

```
298     ...
299     </wsrm:Offer> ?
300     ...
301 </wsrm:CreateSequence>
```

302 /wsrm:CreateSequence

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

307 /wsrm:CreateSequence/wsrm:AcksTo

308 The RM Source MUST include this element in any `CreateSequence` message it sends. This element is of  
309 type `wsa:EndpointReferenceType` (as specified by WS-Addressing). It specifies the endpoint  
310 reference to which messages containing `SequenceAcknowledgement` header blocks and faults related  
311 to the created Sequence are to be sent, unless otherwise noted in this specification (for example, see  
312 Section 3.2).

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

317 /wsrm:CreateSequence/wsrm:Expires

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

322 /wsrm:CreateSequence/wsrm:Expires/@{any}

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

325 /wsrm:CreateSequence/wsrm:Offer

326 This element, if present, enables an RM Source to offer a corresponding Sequence for the reliable  
327 exchange of messages transmitted from RM Destination to RM Source.

328 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier

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

331 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier/@{any}

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

334 /wsrm:CreateSequence/wsrm:Offer/wsrm:Endpoint

335 An RM Source MUST include this element, of type `wsa:EndpointReferenceType` (as specified by  
336 WS-Addressing) This element specifies the endpoint reference to which WS-RM protocol messages  
337 related to the offered Sequence are to be sent.

338 Implementations MUST NOT use an endpoint reference in the Endpoint element that would prevent the  
339 sending of WS-RM protocol messages. For example, using the WS-Addressing  
340 "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM Destination to ever  
341 send WS-RM protocol messages (e.g. `TerminateSequence`) to the RM Source for the Offered  
342 Sequence. Implementations MAY use the WS-RM anonymous URI template and doing so implies that  
343 messages will be retrieved using a mechanism such as the `MakeConnection` message (see section  
344 3.7).

345 `/wsmr:CreateSequence/wsmr:Offer/wsmr:Expires`

346 This element, if present, of type `xs:duration` specifies the duration for the offered Sequence. A value of  
347 'PT0S' indicates that the offered Sequence will never expire. Absence of the element indicates an implied  
348 value of 'PT0S'.

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

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

352 `/wsmr:CreateSequence/wsmr:Offer/wsmr:IncompleteSequenceBehavior`

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

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

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

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

363 `/wsmr:CreateSequence/wsmr:Offer/{any}`

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

366 `/wsmr:CreateSequence/wsmr:Offer/@{any}`

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

369 `/wsmr:CreateSequence/{any}`

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

372 `/wsmr:CreateSequence/@{any}`

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

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

379 The following exemplar defines the `CreateSequenceResponse` syntax:

```
380 <wsmr:CreateSequenceResponse ...>
381   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>
382   <wsmr:Expires ...> xs:duration </wsmr:Expires> ?
383   <wsmr:IncompleteSequenceBehavior>
384     wsmr:IncompleteSequenceBehaviorType
385   </wsmr:IncompleteSequenceBehavior> ?
386   <wsmr:Accept ...>
387     <wsmr:AcksTo> wsa:EndpointReferenceType </wsmr:AcksTo>
388     ...
389   </wsmr:Accept> ?
390   ...
391 </wsmr:CreateSequenceResponse>
```

392 `/wsmr:CreateSequenceResponse`

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

396 `/wsmr:CreateSequenceResponse/wsmr:Identifier`

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

400 `/wsmr:CreateSequenceResponse/wsmr:Identifier/@{any}`

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

403 `/wsmr:CreateSequenceResponse/wsmr:Expires`

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

412 `/wsmr:CreateSequenceResponse/wsmr:Expires/@{any}`

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

415 `/wsmr:CreateSequenceResponse/wsmr:IncompleteSequenceBehavior`

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

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

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

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

426 `/wsrm:CreateSequenceResponse/wsrm:Accept`

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

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

432 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`

433 The RM Destination MUST include this element, of type `wsa:EndpointReferenceType` (as specified  
434 by WS-Addressing). It specifies the endpoint reference to which messages containing  
435 `SequenceAcknowledgement` header blocks and faults related to the created Sequence are to be sent,  
436 unless otherwise noted in this specification (for example, see Section 3.2).

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

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

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

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

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

447 `/wsrm:CreateSequenceResponse/{any}`

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

450 `/wsrm:CreateSequenceResponse/@{any}`

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

## 453 **3.2 Closing A Sequence**

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

459 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of  
460 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept  
461 any new messages for the specified Sequence, other than those already accepted at the time the  
462 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or  
463 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST

464 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`  
465 element) header block on any messages associated with the Sequence destined to the RM Source,  
466 including the `CloseSequenceResponse` message or on any Sequence fault transmitted to the RM Source.

467 While the RM Destination MUST NOT accept any new messages for the specified Sequence it MUST still  
468 process RM protocol messages. For example, it MUST respond to `AckRequested`, `TerminateSequence`  
469 as well as `CloseSequence` messages. Note, subsequent `CloseSequence` messages have no effect on the  
470 state of the Sequence.

471 In the case where the RM Destination wishes to discontinue use of a Sequence it is RECOMMENDED  
472 that it close the Sequence. Please see `Final` and the `SequenceClosed` fault. Whenever possible the  
473 `SequenceClosed` fault SHOULD be used in place of the `SequenceTerminated` fault to allow the RM  
474 Source to still receive Acknowledgements.

475 The following exemplar defines the `CloseSequence` syntax:

```
476 <wsrm:CloseSequence ...>  
477   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
478   ...  
479 </wsrm:CloseSequence>
```

480 `/wsrm:CloseSequence`

481 This element is sent by an RM Source to indicate that the RM Destination MUST NOT accept any new  
482 messages for this Sequence. A `SequenceClosed` fault MUST be generated by the RM Destination when it  
483 receives a message for a Sequence that is already closed.

484 `/wsrm:CloseSequence/wsrm:Identifier`

485 The RM Source MUST include this element in any `CloseSequence` messages it sends. The RM Source  
486 MUST set the value of this element to the absolute URI (conformant with RFC3986) of the Sequence that  
487 is being closed.

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

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

491 `/wsrm:CloseSequence/{any}`

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

494 `/wsrm:CloseSequence@{any}`

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

497 A `CloseSequenceResponse` is sent in the body of a response message by an RM Destination in  
498 response to receipt of a `CloseSequence` request message. It indicates that the RM Destination has  
499 closed the Sequence.

500 The following exemplar defines the `CloseSequenceResponse` syntax:

```
501 <wsrm:CloseSequenceResponse ...>  
502   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
503   ...  
504 </wsrm:CloseSequenceResponse>
```

505 `/wsrm:CloseSequenceResponse`

506 This element is sent in the body of a response message by an RM Destination in response to receipt of a  
507 `CloseSequence` request message. It indicates that the RM Destination has closed the Sequence.

508 /wsmr:CloseSequenceResponse/wsmr:Identifier

509 The RM Destination MUST include this element in any CloseSequenceResponse message it sends. The  
510 RM Destination MUST set the value of this element to the absolute URI (conformant with RFC3986) of the  
511 Sequence that is being closed.

512 /wsmr:CloseSequenceResponse/wsmr:Identifier/@{any}

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

515 /wsmr:CloseSequenceResponse/{any}

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

518 /wsmr:CloseSequenceResponse@{any}

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

### 521 **3.3 Sequence Termination**

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

529 The following exemplar defines the `TerminateSequence` syntax:

```
530 <wsmr:TerminateSequence ...>  
531   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
532   ...  
533 </wsmr:TerminateSequence>
```

534 /wsmr:TerminateSequence

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

540 /wsmr:TerminateSequence/wsmr:Identifier

541 The RM Source MUST include this element in any `TerminateSequence` message it sends. The RM  
542 Source MUST set the value of this element to the absolute URI (conformant with RFC3986) of the  
543 Sequence that is being terminated.

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

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

547 /wsmr:TerminateSequence/{any}



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

550 `/wsmr:TerminateSequence/@{any}`

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

553 A `TerminateSequenceResponse` is sent in the body of a response message by an RM Destination in  
554 response to receipt of a `TerminateSequence` request message. It indicates that the RM Destination has  
555 terminated the Sequence.

556 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
557 <wsmr:TerminateSequenceResponse ...>  
558   <wsmr:Identifier ...> xs:anyURI </wsmr:Identifier>  
559   ...  
560 </wsmr:TerminateSequenceResponse>
```

561 `/wsmr:TerminateSequenceResponse`

562 This element is sent in the body of a response message by an RM Destination in response to receipt of a  
563 `TerminateSequence` request message. It indicates that the RM Destination has terminated the  
564 Sequence. The RM Destination MUST NOT send this element as a header block.

565 `/wsmr:TerminateSequenceResponse/wsmr:Identifier`

566 The RM Destination MUST include this element in any `TerminateSequenceResponse` message it  
567 sends. The RM Destination MUST set the value of this element to the absolute URI (conformant with  
568 RFC3986) of the Sequence that is being terminated.

569 `/wsmr:TerminateSequenceResponse/wsmr:Identifier/@{any}`

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

572 `/wsmr:TerminateSequenceResponse/{any}`

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

575 `/wsmr:TerminateSequenceResponse/@{any}`

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

578 On receipt of a `TerminateSequence` message an RM Destination MUST respond with a corresponding  
579 `TerminateSequenceResponse` message or generate a fault `UnknownSequenceFault` if the  
580 Sequence is not known.

## 581 **3.4 Sequences**

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

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

589 A following exemplar defines its syntax:

```
590 <wsrm:Sequence ...>
591   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
592   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>
593   ...
594 </wsrm:Sequence>
```

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

596 /wsrm:Sequence

597 This protocol element associates the message in which it is contained with a previously established RM  
598 Sequence. It contains the Sequence's unique identifier and the containing message's ordinal position  
599 within that Sequence. The RM Destination MUST understand the *Sequence* header block. The RM  
600 Source MUST assign a *mustUnderstand* attribute with a value 1/true (from the namespace  
601 corresponding to the version of SOAP to which the *Sequence* SOAP header block is bound) to the  
602 *Sequence* header block element.

603 /wsrm:Sequence/wsrm:Identifier

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

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

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

610 /wsrm:Sequence/wsrm:MessageNumber

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

615 /wsrm:Sequence/{any}

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

618 /wsrm:Sequence/@{any}

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

621 The following example illustrates a Sequence header block.

```
622 <wsrm:Sequence>
623   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
624   <wsrm:MessageNumber>10</wsrm:MessageNumber>
625 </wsrm:Sequence>
```

### 626 3.5 Request Acknowledgement

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

629 The RM Source MAY request an acknowledgement message from the RM Destination at any time by  
630 including an *AckRequested* header block in any message targeted to the RM Destination. An RM

631 Destination that receives a message that contains an `AckRequested` header block MUST send a  
632 message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference  
633 (see Section 3.1) for a known Sequence or else generate an `UnknownSequence` fault. If a non-  
634 `mustUnderstand` fault occurs when processing an RM header that was piggy-backed on another  
635 message, a fault MUST be generated, but the processing of the original message MUST NOT be  
636 affected. It is RECOMMENDED that the RM Destination return a `AcknowledgementRange` or `None`  
637 element instead of a `Nack` element (see Section 3.6).

638 The following exemplar defines its syntax:

```
639 <wsm:AckRequested ...>  
640   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
641   ...  
642 </wsm:AckRequested>
```

643 `/wsm:AckRequested`

644 This element requests an acknowledgement for the identified Sequence.

645 `/wsm:AckRequested/wsm:Identifier`

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

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

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

652 `/wsm:AckRequested/{any}`

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

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

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

## 658 3.6 Sequence Acknowledgement

659 The RM Destination informs the RM Source of successful message receipt using a  
660 `SequenceAcknowledgement` header block. The RM Destination MAY transmit the  
661 `SequenceAcknowledgement` header block independently or it MAY include the  
662 `SequenceAcknowledgement` header block on any message targeted to the `AcksTo` EPR.  
663 Acknowledgements can be explicitly requested using the `AckRequested` directive (see Section 3.5). If a  
664 non-`mustUnderstand` fault occurs when processing an RM header that was piggy-backed on another  
665 message, a fault MUST be generated, but the processing of the original message MUST NOT be  
666 affected.

667 A RM Destination MAY include a `SequenceAcknowledgement` header block on any SOAP envelope  
668 targetted to the endpoint referenced by the `AcksTo` EPR.

669 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the  
670 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing  
671 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST transmit any  
672 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be transmitted

673 on the protocol binding-specific channel. Such a channel is provided by the context of a received message  
674 containing a SOAP envelope that contains a `Sequence` header block and/or a `AckRequested` header  
675 block for that same `Sequence` identifier.

676 The following exemplar defines its syntax:

```
677 <wsm:SequenceAcknowledgement ...>  
678   <wsm:Identifier ...> xs:anyURI </wsm:Identifier>  
679   [ [ [ <wsm:AcknowledgementRange ...  
680         Upper="wsm:MessageNumberType"  
681         Lower="wsm:MessageNumberType"/> +  
682         | <wsm:None/> ]  
683         <wsm:Final/> ? ]  
684         | <wsm:Nack> wsm:MessageNumberType </wsm:Nack> + ]  
685   ...  
686   ...  
687 </wsm:SequenceAcknowledgement>
```

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

689 `/wsm:SequenceAcknowledgement`

690 This element contains the `Sequence` acknowledgement information.

691 `/wsm:SequenceAcknowledgement/wsm:Identifier`

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

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

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

700 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange`

701 The RM Destination MAY include one or more instances of this element within a  
702 `SequenceAcknowledgement` header block. It contains a range of `Sequence` `MessageNumbers`  
703 successfully accepted by the RM Destination. The ranges SHOULD NOT overlap. The RM Destination  
704 MUST NOT include this element if a sibling `Nack` or `None` element is also present as a child of  
705 `SequenceAcknowledgement`.

706 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@Upper`

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

709 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@Lower`

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

712 `/wsm:SequenceAcknowledgement/wsm:AcknowledgementRange/@{any}`

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

715 `/wsm:SequenceAcknowledgement/wsm:None`

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

720 `/wsrm:SequenceAcknowledgement/wsrm:Final`

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

727 `/wsrm:SequenceAcknowledgement/wsrm:Nack`

728 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If  
729 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing  
730 the `MessageNumber` of an unreceived message in a Sequence. The RM Destination MUST NOT include  
731 a `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of  
732 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the  
733 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`  
734 containing a `Nack` for a message that it has previously acknowledged within a  
735 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing  
736 a `Nack` for a message that has previously been acknowledged within a `AcknowledgementRange`.

737 `/wsrm:SequenceAcknowledgement/{any}`

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

740 `/wsrm:SequenceAcknowledgement/@{any}`

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

743 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
745 <wsrm:SequenceAcknowledgement>  
746   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
747   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
748 </wsrm:SequenceAcknowledgement>
```

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

```
751 <wsrm:SequenceAcknowledgement>  
752   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
753   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
754   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
755   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
756 </wsrm:SequenceAcknowledgement>
```

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

```
758 <wsrm:SequenceAcknowledgement>  
759   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
```

```
760 <wsrm:Nack>3</wsrm:Nack>
761 </wsrm:SequenceAcknowledgement>
```

### 762 3.7 MakeConnection

763 When an endpoint is not directly addressable (e.g. behind a firewall or not able to allow incoming  
764 connections), an anonymous URI in the EPR address property can indicate such an endpoint. The WS-  
765 Addressing anonymous URI is one such anonymous URI. This specification defines a URI template (the  
766 WS-RM anonymous URI) which may be used to uniquely identify anonymous endpoints.

```
767 http://docs.oasis-open.org/ws-rx/wsr/200604/anonymous?id={uuid}
```

768 This URI template in an EPR indicates a protocol-specific back-channel will be established through a  
769 mechanism such as `MakeConnection`, defined below. When using this URI template, “{uuid}” MUST be  
770 replaced by a UUID value as defined by RFC4122[UUID]. This UUID value uniquely distinguishes the  
771 endpoint. A sending endpoint SHOULD transmit messages at endpoints identified with the URI template  
772 using a protocol-specific back-channel, including but not limited to those established with a  
773 `MakeConnection` message. Note, this URI is semantically similar to the WS-Addressing anonymous  
774 URI if a protocol-specific back-channel is available.

775 The `MakeConnection` is a one-way operation that establishes a contextualized back-channel for the  
776 transmission of messages according to matching criteria (defined below). In the non-faulting case, if no  
777 matching message is available then no SOAP envelopes will be returned on the back-channel. A common  
778 usage will be a client RM Destination sending `MakeConnection` to a server RM Source for the purpose  
779 of receiving asynchronous response messages.

780 The following exemplar defines the `MakeConnection` syntax:

```
781 <wsrm:MakeConnection ...>
782   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?
783   <wsrm:Address ...> xs:anyURI </wsrm:Address> ?
784   ...
785 </wsrm:MakeConnection>
```

786 `/wsrm:MakeConnection`

787 This element allows the sender to create a transport-specific back-channel that can be used to return a  
788 message that matches the selection criteria. Endpoints MUST NOT send this element as a header block.

789 `/wsrm:MakeConnection/wsrm:Identifier`

790 This element specifies the WS-RM Sequence Identifier that establishes the context for the transport-  
791 specific back-channel. The Sequence Identifier should be compared with the Sequence Identifiers  
792 associated with the messages held by the sending endpoint, and if there is a matching message it will be  
793 returned. If this element is omitted from the message then the `Address` MUST be included in the  
794 message.

795 `/wsrm:MakeConnection/wsrm:Identifier/@{any}`

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

798 `/wsrm:MakeConnection/wsrm:Address`

799 This element specifies the URI (`wsa:Address`) of the initiating endpoint. Endpoints MUST NOT return  
800 messages on the transport-specific back-channel unless they have been addressed to this URI. This  
801 `Address` property and a message’s WS-Addressing destination property are considered identical when  
802 they are exactly the same character-for-character. Note that URIs which are not identical in this sense  
803 may in fact be functionally equivalent. Examples include URI references which differ only in case, or

804 which are in external entities which have different effective base URIs. If this element is omitted from the  
805 message then the `Identifier` MUST be included in the message.

806 `/wsrm:MakeConnection/wsrm:Address/@{any}`

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

809 `/wsrm:MakeConnection/{any}`

810 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,  
811 to be passed. This allows fine-tuning of the messages to be returned, additional selection criteria included  
812 here are logically ANDed with the `Address` and/or `Identifier`. If an extension is not supported by the  
813 endpoint then it should return a `UnsupportedSelection` fault.

814 `/wsrm:MakeConnection/@{any}`

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

817 If both `Identifier` and `Address` are present, then the endpoint processing the `MakeConnection`  
818 message MUST insure that any SOAP Envelope flowing on the backchannel MUST be associated with  
819 the given `Sequence` and MUST be addressed to the given URI.

820 The management of messages that are awaiting the establishment of a back-channel to their receiving  
821 endpoint is an implementation detail that is outside the scope of this specification. Note, however, that  
822 these messages form a class of asynchronous messages that is not dissimilar from “ordinary”  
823 asynchronous messages that are waiting for the establishment of a connection to their destination  
824 endpoints.

825 This specification places no constraint on the types of messages that can be returned on the transport-  
826 specific back-channel. As in an asynchronous environment, it is up to the recipient of the  
827 `MakeConnection` message to decide which messages are appropriate for transmission to any particular  
828 endpoint. However, the endpoint processing the `MakeConnection` message MUST insure that the  
829 messages match the selection criteria as specified by the child elements of the `MakeConnection`  
830 element.

### 831 **3.8 MessagePending**

832 When `MakeConnection` is used, and a message is returned on the transport-specific back-channel, the  
833 `MessagePending` header SHOULD be included on the returned message as an indicator whether there  
834 are additional messages waiting to be retrieved using the same selection criteria that was specified in the  
835 `MakeConnection` element.

836 The following exemplar defines the `MessagePending` syntax:

```
837 <wsrm:MessagePending pending="xs:boolean" ...>  
838   ...  
839 </wsrm:MessagePending>
```

840 `/wsrm:MessagePending`

841 This element indicates whether additional messages are waiting to be retrieved.

842 `/wsrm:MessagePending@pending`

843 This attribute, when set to 'true', indicates that there is at least one message waiting to be retrieved. When  
844 this attribute is set to 'false' it indicates there are currently no messages waiting to be retrieved.

845 /wsm:MessagePending/{any}

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

848 /wsm:MessagePending/@{any}

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

851 The absence of the `MessagePending` header has no implication as to whether there are additional  
852 messages waiting to be retrieved.



## 853 4 Faults

854 Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. Create  
855 Sequence Refused is a possible fault reply for this operation. Unknown Sequence is a fault generated by  
856 endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences  
857 are detected. WSRM Required is a fault generated an RM Destination that requires the use of WS-RM on  
858 a received message that did not use the protocol. All other faults in this section relate to known  
859 Sequences. RM Destinations that generate Sequence faults SHOULD send those faults to the same  
860 [destination] as `SequenceAcknowledgement` messages.

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

863 `http://docs.oasis-open.org/ws-rx/wsr/200604/fault`

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

866 The definitions of faults use the following properties:

867 [Code] The fault code.

868 [Subcode] The fault subcode.

869 [Reason] The English language reason element.

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

873 Entities that generate WS-ReliableMessaging faults MUST set the [Code] property to either "Sender" or  
874 "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

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

```
876 <S:Envelope>
877   <S:Header>
878     <wsa:Action>
879       http://docs.oasis-open.org/ws-rx/wsr/200604/fault
880     </wsa:Action>
881     <!-- Headers elided for clarity. -->
882   </S:Header>
883   <S:Body>
884     <S:Fault>
885       <S:Code>
886         <S:Value> [Code] </S:Value>
887         <S:Subcode>
888           <S:Value> [Subcode] </S:Value>
889         </S:Subcode>
890       </S:Code>
891       <S:Reason>
892         <S:Text xml:lang="en"> [Reason] </S:Text>
893       </S:Reason>
894       <S:Detail>
895         [Detail]
896       ...
```

```
897     </S:Detail>
898   </S:Fault>
899 </S:Body>
900 </S:Envelope>
```

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

```
903 <S11:Envelope>
904   <S11:Header>
905     <wsrm:SequenceFault>
906       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
907       <wsrm:Detail> [Detail] </wsrm:Detail>
908       ...
909     </wsrm:SequenceFault>
910     <!-- Headers elided for clarity. -->
911   </S11:Header>
912   <S11:Body>
913     <S11:Fault>
914       <faultcode> [Code] </faultcode>
915       <faultstring> [Reason] </faultstring>
916     </S11:Fault>
917   </S11:Body>
918 </S11:Envelope>
```

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

```
921 <S11:Envelope>
922   <S11:Body>
923     <S11:Fault>
924       <faultcode> [Subcode] </faultcode>
925       <faultstring> [Reason] </faultstring>
926     </S11:Fault>
927   </S11:Body>
928 </S11:Envelope>
```

## 929 4.1 SequenceFault Element

930 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during  
931 the reliable messaging specific processing of a message belonging to a Sequence. WS-  
932 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP  
933 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in  
934 conjunction with the SOAP 1.2 binding.

935 The following exemplar defines its syntax:

```
936 <wsrm:SequenceFault ...>
937   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
938   <wsrm:Detail> ... </wsrm:Detail> ?
939   ...
940 </wsrm:SequenceFault>
```

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

942 `/wsrm:SequenceFault`

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

944 `/wsrm:SequenceFault/wsrm:FaultCode`

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

947 /wsm:SequenceFault/wsm:Detail  
 948 This element, if present, carries application specific error information related to the fault being described.  
 949 /wsm:SequenceFault/wsm:Detail/{any}  
 950 The application specific error information related to the fault being described.  
 951 /wsm:SequenceFault/wsm:Detail/@{any}  
 952 The application specific error information related to the fault being described.  
 953 /wsm:SequenceFault/{any}  
 954 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,  
 955 to be passed.  
 956 /wsm:SequenceFault/@{any}  
 957 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the  
 958 element.

959 **4.2 Sequence Terminated**

960 The endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding  
 961 endpoint of this decision.  
 962 Properties:  
 963 [Code] Sender or Receiver  
 964 [Subcode] wsm:SequenceTerminated  
 965 [Reason] The Sequence has been terminated due to an unrecoverable error.  
 966 [Detail]

967 `<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.

968 **4.3 Unknown Sequence**

969 Properties:  
 970 [Code] Sender  
 971 [Subcode] wsm:UnknownSequence  
 972 [Reason] The value of wsm:Identifier is not a known Sequence identifier.  
 973 [Detail]

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

#### 975 4.4 Invalid Acknowledgement

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

978 [Code] Sender

979 [Subcode] `wsrn:InvalidAcknowledgement`

980 [Reason] The `SequenceAcknowledgement` violates the cumulative acknowledgement invariant.

981 [Detail]

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

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source.	In response to a <code>SequenceAcknowledgement</code> that violate the invariants stated in 2.3 or any of the requirements in 3.6 about valid combinations of <code>AckRange</code> , <code>Nack</code> and <code>None</code> in a single <code>SequenceAcknowledgement</code> element or with respect to already received such elements.	Unspecified.	Unspecified.

#### 983 4.5 Message Number Rollover

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

985 Properties:

986 [Code] Sender

987 [Subcode] `wsrn:MessageNumberRollover`

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

989 [Detail]

990 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`  
991 `<wsrm:MaxMessageNumber> wsrm:MessageNumberType </wsrm:MaxMessageNumber>`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	Message number in / <code>wsrc:Sequence/wsrc:MessageNumber</code> of a received message exceeds the internal limitations of an RM Destination or reaches the maximum value of 9,223,372,036,854,775,807.	RM Destination SHOULD continue to accept undelivered messages until the Sequence is closed or terminated.	RM Source SHOULD continue to retransmit undelivered messages until the Sequence is closed or terminated. The RM Source MUST NOT send any new messages.

992 **4.6 Create Sequence Refused**

993 Properties:

994 [Code] Sender

995 [Subcode] `wsrc:CreateSequenceRefused`

996 [Reason] The create Sequence request has been refused by the RM Destination.

997 [Detail]

998 `xs:any`

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

999 **4.7 Sequence Closed**

1000 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1001 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that  
 1002 is closed or when an RM Destination is asked to close a Sequence that is already closed.

1003 Properties:

1004 [Code] Sender

1005 [Subcode] `wsrc:SequenceClosed`

1006 [Reason] The Sequence is closed and can not accept new messages.

1007 [Detail]

1008 `<wsrc:Identifier...> xs:anyURI </wsrc: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.

## 1009 4.8 WSRM Required

1010 If an RM Destination requires the use of WS-RM, this fault is generated when it receives an incoming  
1011 message that did not use this protocol.

1012 Properties:

1013 [Code] Sender

1014 [Subcode] wsrn:WSRMRequired

1015 [Reason] The RM Destination requires the use of WSRM.

1016 [Detail]

1017 `xs:any`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Destination.	On receipt of a message that does not use this protocol and for which this protocol is required.	Unspecified.	Unspecified.

## 1018 4.9 Unsupported Selection

1019 The QName of the unsupported element(s) are included in the detail.

1020 Properties:

1021 [Code] Receiver

1022 [Subcode] wsrn:UnsupportedSelection

1023 [Reason] The extension element used in the message selection is not supported by the RM Source

1024 [Detail]

1025 `<wsrn:UnsupportedElement> xs:QName </wsrn:UnsupportedElement>+`

Generated by	Condition	Action Upon Generation	Action Upon Receipt
RM Source or RM Destination.	In response to a <code>MakeConnection</code> message containing a selection criteria in the extensibility section of the message that is not support.ed	Unspecified.	Unspecified.

## 1026 **5 Security Threats and Countermeasures**

1027 This specification considers two sets of security requirements, those of the applications that use the WS-  
1028 RM protocol and those of the protocol itself.

1029 This specification makes no assumptions about the security requirements of the applications that use WS-  
1030 RM. However, once those requirements have been satisfied within a given operational context, the  
1031 addition of WS-RM to this operational context should not undermine the fulfillment of those requirements;  
1032 the use of WS-RM should not create additional attack vectors within an otherwise secure system.

1033 There are many other security concerns that one may need to consider when implementing or using this  
1034 protocol. The material below should not be considered as a "check list". Implementers and users of this  
1035 protocol are urged to perform a security analysis to determine their particular threat profile and the  
1036 appropriate responses to those threats.

1037 Implementers are also advised that there is a core tension between security and reliable messaging that  
1038 can be problematic if not addressed by implementations; one aspect of security is to prevent message  
1039 replay but one of the invariants of this protocol is to resend messages until they are acknowledged.  
1040 Consequently, if the security sub-system processes a message but a failure occurs before the reliable  
1041 messaging sub-system receives that message, then it is possible (and likely) that the security sub-system  
1042 will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-  
1043 system will likely continue to expect and even solicit the missing message(s). Care should be taken to  
1044 avoid and prevent this condition.

### 1045 **5.1 Threats and Countermeasures**

1046 The primary security requirement of this protocol is to protect the specified semantics and protocol  
1047 invariants against various threats. The following sections describe several threats to the integrity and  
1048 operation of this protocol and provide some general outlines of countermeasures to those threats.  
1049 Implementers and users of this protocol should keep in mind that all threats are not necessarily applicable  
1050 to all operational contexts.

#### 1051 **5.1.1 Integrity Threats**

1052 In general, any mechanism which allows an attacker to alter the information in a Sequence Traffic  
1053 Message, Sequence Lifecycle Message, or Sequence-related fault, or which allows an attacker to alter the  
1054 correlation of a RM Protocol Header Block to its intended message represents a threat to the WS-RM  
1055 protocol.

1056 For example, if an attacker is able to swap `Sequence` headers on messages in transit between the RM  
1057 Source and RM Destination then they have undermined the implementation's ability to guarantee the first  
1058 invariant described in Section 2.3. The result is that there is no way of guaranteeing that messages will be  
1059 delivered to the Application Destination in the same order that they were sent by the Application Source.

##### 1060 **5.1.1.1 Countermeasures**

1061 Integrity threats are generally countered via the use of digital signatures some level of the communication  
1062 protocol stack. Note that, in order to counter header swapping attacks, the signature SHOULD include  
1063 both the SOAP body and any relevant SOAP headers (e.g. `Sequence` header). Because some headers  
1064 (`AckRequested`, `SequenceAcknowledgement`) are independent of the body of the SOAP message in which  
1065 they occur, implementations MUST allow for signatures that cover only these headers.



## 1066 **5.1.2 Resource Consumption Threats**

1067 The creation of a Sequence with an RM Destination consumes various resources on the systems used to  
1068 implement that RM Destination. These resources can include network connections, database tables,  
1069 message queues, etc. This behavior can be exploited to conduct denial of service attacks against an RM  
1070 Destination. For example, a simple attack is to repeatedly send `CreateSequence` messages to an RM  
1071 Destination. Another attack is to create a Sequence for a service that is known to require in-order  
1072 message delivery and use this Sequence to send a stream of very large messages to that service, making  
1073 sure to omit message number "1" from that stream.

### 1074 **5.1.2.1 Countermeasures**

1075 There are a number of countermeasures against the described resource consumption threats. The  
1076 technique advocated by this specification is for the RM Destination to restrict the ability to create a  
1077 Sequence to a specific set of entities/principals. This reduces the number of potential attackers and, in  
1078 some cases, allows the identity of any attackers to be determined.

1079 The ability to restrict Sequence creation depends, in turn, upon the RM Destination's ability identify and  
1080 authenticate the RM Source that issued the `CreateSequence` message.

## 1081 **5.1.3 Sequence Spoofing Threats**

1082 Sequence spoofing is a class of threats in which the attacker uses knowledge of the `Identifier` for a  
1083 particular Sequence to forge Sequence Lifecycle or Traffic Messages. For example the attacker creates a  
1084 fake `TerminateSequence` message that references the target Sequence and sends this message to the  
1085 appropriate RM Destination. Some sequence spoofing attacks also require up-to-date knowledge of the  
1086 current `MessageNumber` for their target Sequence.

1087 In general any Sequence Lifecycle Message, RM Protocol Header Block, or sequence-correlated SOAP  
1088 fault (e.g. `InvalidAcknowledgement`) can be used by someone with knowledge of the Sequence identifier  
1089 to attack the Sequence. These attacks are "two-way" in that an attacker may choose to target the RM  
1090 Source by, for example, inserting a fake `SequenceAcknowledgement` header into a message that it sends  
1091 to the `AcksTo` EPR of an RM Source.

### 1092 **5.1.3.1 Sequence Hijacking**

1093 Sequence hijacking is a specific case of a sequence spoofing attack. The attacker attempts to inject  
1094 Sequence Traffic Messages into an existing Sequence by inserting fake `Sequence` headers into those  
1095 messages.

1096 Note that "sequence hijacking" should not be equated with "security session hijacking". Although a  
1097 Sequence may be bound to some form of a security session in order to counter the threats described in  
1098 this section, applications MUST NOT rely on WS-RM-related information to make determinations about  
1099 the identity of the entity that created a message; applications SHOULD rely only upon information that is  
1100 established by the security infrastructure to make such determinations. Failure to observe this rule  
1101 creates, among other problems, a situation in which the absence of WS-RM may deprive an application of  
1102 the ability to authenticate its peers even though the necessary security processing has taken place.

### 1103 **5.1.3.2 Countermeasures**

1104 There are a number of countermeasures against sequence spoofing threats. The technique advocated by  
1105 this specification is to consider the Sequence to be a shared resource that is jointly owned by the RM

1106 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that  
1107 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence  
1108 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it receives that  
1109 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.  
1110 For its part the RM Source SHOULD ensure that every message or fault that it receives that refers to a  
1111 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1112 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and  
1113 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its  
1114 sequence peer it MUST be able to identify and authenticate the entity that sent the  
1115 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a  
1116 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that  
1117 message and, if necessary, correlate this identity with the sequence peer identity established at sequence  
1118 creation time.

## 1119 **5.2 Security Solutions and Technologies**

1120 The security threats described in the previous sections are neither new nor unique. The solutions that  
1121 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This  
1122 section maps the facilities provided by common web services security solutions against countermeasures  
1123 described in the previous sections.

1124 Before continuing this discussion, however, some examination of the underlying requirements of the  
1125 previously described countermeasures is necessary. Specifically it should be noted that the technique  
1126 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates  
1127 the issuer of a `CreateSequence` message. Secondly, the RM Destination to performs an authorization  
1128 check against this authenticated identity and determines if the RM Source is permitted to create  
1129 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime  
1130 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any  
1131 discussion of such facilities is considered to be beyond the scope of this specification.

### 1132 **5.2.1 Transport Layer Security**

1133 This section describes how the the facilities provided by SSL/TLS [RFC 4346] can be used to implement  
1134 the countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints  
1135 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1136 The description provided here is general in nature and is not intended to serve as a complete definition on  
1137 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the  
1138 choice of features as well as the manner in which they will be used. The mechanisms described in the  
1139 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the  
1140 requirements and constraints of the use of SSL/TLS.

#### 1141 **5.2.1.1 Model**

1142 The basic model for using SSL/TLS is as follows:

- 1143 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1144 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM  
1145 Destination.

- 1146 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an  
1147 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a  
1148 synchronous `CreateSequenceResponse` using the session established in (1).
- 1149 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to transmit  
1150 any and all messages or faults that refer to that Sequence.
- 1151 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established  
1152 in (3) to transmit any and all messages or faults that refer to that Sequence or, for synchronous  
1153 exchanges, the RM Destination uses the SSL/TLS session established in (1).

### 1154 5.2.1.2 Countermeasure Implementation

1155 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the  
1156 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the  
1157 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If  
1158 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and  
1159 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1160 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification  
1161 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods  
1162 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS  
1163 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1164 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP  
1165 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the  
1166 establishment of the the SSL/TLS session, the sending party authenticates itself to the receiving  
1167 party using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might  
1168 authenticate itself to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using  
1169 BasicAuth. Similarly the RM Destination might authenticate itself to the RM Source (e.g. when  
1170 sending an acknowledgement) using BasicAuth.
- 1171 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the  
1172 connection authenticates itself to the party accepting the connection using an X.509 certificate  
1173 that is exchanged during the SSL/TLS handshake.

1174 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself  
1175 using one the above mechanisms. The authenticated identity can then be used to determine if the RM  
1176 Source is authorized to create a Sequence with the RM Destination.

1177 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1178 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the  
1179 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than  
1180 on authentication information. For example, an RM Destination can determine that a Sequence Traffic  
1181 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS  
1182 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a  
1183 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a  
1184 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used  
1185 to protect that Sequence.

1186 This specification does not preclude the use of other methods of using SSL/TLS to implement the  
1187 countermeasures (such as associating specific authentication information with a Sequence) although such  
1188 methods are not covered by this document.

1189 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS  
1190 session) are outside the scope of this specification.

## 1191 **5.2.2 SOAP Message Security**

1192 The mechanisms described in WS-Security may be used in various ways to implement the  
1193 countermeasures described in the previous sections. This specification advocates using the protocol  
1194 described by WS-SecureConversation [WS-SecureConversation] (optionally in conjunction with WS-Trust  
1195 [Trust]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component  
1196 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1197 The description provided here is general in nature and is not intended to serve as a complete definition on  
1198 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations  
1199 need to agree on the choice of features as well as the manner in which they will be used. The  
1200 mechanisms described in the Web Services Security Policy Language MAY be used by services to  
1201 describe the requirements and constraints of the use of WS-SecureConversation.

### 1202 **5.2.2.1 Model**

1203 The basic model for using WS-SecureConversation is as follows:

- 1204 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This  
1205 may involve the participation of third parties such as a security token service. The tokens  
1206 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1207 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security  
1208 context that will be used to protect the Sequence. This is done so that, in cases where the  
1209 `CreateSequence` message is signed by more than one security context, the RM Source can  
1210 indicate which security context should be used to protect the newly created Sequence.
- 1211 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)  
1212 associated with the security context to sign (as defined by WS-Security) at least the body and any  
1213 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

### 1214 **5.2.2.2 Countermeasure Implementation**

1215 Without relying upon any authentication information, the per-message signatures provide the necessary  
1216 integrity qualities to counter the threats described in Section 5.1.1.

1217 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of  
1218 authentication claims must be provided by the RM Source to the RM Destination during the establishment  
1219 of the Security Context. These claims can then be used to determine if the RM Source is authorized to  
1220 create a Sequence with the RM Destination.

1221 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring  
1222 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the  
1223 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security  
1224 context rather than on any authentication claims that may have been established during security context  
1225 initiation. Note that other methods of using WS-SecurityConversation to implement the countermeasures  
1226 (such as associating specific authentication claims to a Sequence) are possible but not covered by this  
1227 document.

1228 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer  
1229 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1230 the association between a Sequence and its protecting security context cannot always be established  
1231 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and  
1232 `CreateSequenceResponse` messages may be signed by more than one security context.

1233 Issues specific to the life-cycle management of WS-SecurityConversation security contexts (such as  
1234 amending or renewing contexts) are outside the scope of this specification.

## 1235 6 Securing Sequences

1236 As noted in Section 5, the RM Source and RM Destination should be able to protect their shared  
1237 Sequences against the threat of Sequence Spoofing attacks. There are a number of OPTIONAL means of  
1238 achieving this objective depending upon the underlying security infrastructure.

### 1239 6.1 Securing Sequences Using WS-Security

1240 One mechanism for protecting a Sequence is to include a security token using a  
1241 `wsse:SecurityTokenReference` element from WS-Security (see section 9 in WS-  
1242 SecureConversation) in the `CreateSequence` element. This establishes an association between the  
1243 created (and, if present, offered) Sequence(s) and the referenced security token, such that the RM Source  
1244 and Destination MUST use the security token as the basis for authorization of all subsequent interactions  
1245 related to the Sequence(s). The `wsse:SecurityTokenReference` explicitly identifies the token as  
1246 there may be more than one token on a `CreateSequence` message or inferred from the communication  
1247 context (e.g. transport protection).

1248 It is RECOMMENDED that a message independent referencing mechanism be used to identify the token,  
1249 if the token being referenced supports such mechanism.

1250 The following exemplar defines the `CreateSequence` syntax when extended to include a  
1251 `wsse:SecurityTokenReference`:

```
1252 <wsrm:CreateSequence ...>  
1253   <wsrm:AcksTo ...> wsa:EndpointReferenceType </wsrm:AcksTo>  
1254   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1255   <wsrm:Offer ...>  
1256     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
1257     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?  
1258     ...  
1259   </wsrm:Offer> ?  
1260   ...  
1261   <wsse:SecurityTokenReference>  
1262     ...  
1263   </wsse:SecurityTokenReference> ?  
1264   ...  
1265 </wsrm:CreateSequence>
```

1266 `/wsrm:CreateSequence/wsse:SecurityTokenReference`

1267 This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in  
1268 section 3.1) to communicate an explicit reference to the security token, using a  
1269 `wsse:SecurityTokenReference` as documented in WS-Security [WSSecurity], that the RM Source  
1270 and Destination MUST use to authorize messages for the created (and, if present, the offered) Sequence  
1271 (s). All subsequent messages related to the created (and, if present, the offered) Sequence(s) MUST  
1272 demonstrate proof-of-rights to the referenced key (e.g., using the key or deriving from the key).

1273 When a RM Source transmits a `CreateSequence` that has been extended to include a  
1274 `wsse:SecurityTokenReference` it SHOULD ensure that the RM Destination both understands and  
1275 will conform with the requirements listed above. In order to achieve this, the RM Source SHOULD include  
1276 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This  
1277 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source  
1278 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands  
1279 and conforms with the requirements listed above. Note that an RM Destination understanding this header  
1280 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined  
1281 in WS-Security still applies.

1282 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1283 <wsrm:UsesSequenceSTR ... />
```

1284 `/wsrm:UsesSequenceSTR`

1285 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the  
1286 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value  
1287 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension  
1288 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested  
1289 Sequence creation.

1290 The following is an example of a `CreateSequence` message using the

1291 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1292 <soap:Envelope ...>
1293   <soap:Header>
1294     ...
1295     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />
1296     ...
1297   </soap:Header>
1298   <soap:Body>
1299     <wsrm:CreateSequence>
1300       <wsrm:AcksTo>
1301         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1302       </wsrm:AcksTo>
1303       <wsse:SecurityTokenReference>
1304         ...
1305       </wsse:SecurityTokenReference>
1306     </wsrm:CreateSequence>
1307   </soap:Body>
1308 </soap:Envelope>
```

## 1309 6.2 Securing Sequences Using SSL/TLS

1310 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).  
1311 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying  
1312 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a  
1313 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a  
1314 SOAP header block within the `CreateSequence` message.

1315 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1316 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1317 `/wsrm:UsesSequenceSSL`

1318 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to  
1319 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was  
1320 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a  
1321 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand  
1322 and correctly implement the functionality described in Section 5.2.1 or else generate a  
1323 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1324 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related  
1325 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from  
1326 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the  
1327 `CreateSequenceResponse` message.

## 1328 **7 References**

### 1329 **7.1 Normative**

#### 1330 **[KEYWORDS]**

1331 S. Bradner, "[Key words for use in RFCs to Indicate Requirement Levels](#)," RFC 2119, Harvard University,  
1332 March 1997

#### 1333 **[SOAP 1.1]**

1334 W3C Note, "[SOAP: Simple Object Access Protocol 1.1](#)," 08 May 2000.

#### 1335 **[SOAP 1.2]**

1336 W3C Recommendation, "[SOAP Version 1.2 Part 1: Messaging Framework](#)" June 2003.

#### 1337 **[URI]**

1338 T. Berners-Lee, R. Fielding, L. Masinter, "[Uniform Resource Identifiers \(URI\): Generic Syntax](#)," RFC 3986,  
1339 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

#### 1340 **[UUID]**

1341 P. Leach, M. Mealling, R. Salz, "[A Universally Unique Identifier \(UUID\) URN Namespace](#)," RFC 4122,  
1342 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

#### 1343 **[XML]**

1344 W3C Recommendation, "[Extensible Markup Language \(XML\) 1.0 \(Second Edition\)](#)", October 2000.

#### 1345 **[XML-ns]**

1346 W3C Recommendation, "[Namespaces in XML](#)," 14 January 1999.

#### 1347 **[XML-Schema Part1]**

1348 W3C Recommendation, "[XML Schema Part 1: Structures](#)," 2 May 2001.

#### 1349 **[XML-Schema Part2]**

1350 W3C Recommendation, "[XML Schema Part 2: Datatypes](#)," 2 May 2001.

#### 1351 **[XPath 1.0]**

1352 W3C Recommendation, "[XML Path Language \(XPath\) Version 1.0](#)," 16 November 1999.

#### 1353 **[WSDL 1.1]**

1354 W3C Note, "[Web Services Description Language \(WSDL 1.1\)](#)," 15 March 2001.

#### 1355 **[WS-Addressing]**

1356 W3C Recommendation, "[Web Services Addressing 1.0 - Core](#)", May 2006.

1357 W3C Recommendation, "[Web Services Addressing 1.0 – SOAP Binding](#)", May 2006.

### 1358 **7.2 Non-Normative**

#### 1359 **[BSP 1.0]**

1360 WS-I Working Group Draft. "[Basic Security Profile Version 1.0](#)," March 2006

#### 1361 **[RDDL 2.0]**



- 1362 Johnathan Borden, Tim Bray, eds. "[Resource Directory Description Language \(RDDL\) 2.0](#)," January 2004
- 1363 **[RFC 2617]**
- 1364 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "[HTTP](#)  
1365 [Authentication: Basic and Digest Access Authentication](#)," June 1999.
- 1366 **[RFC 4346]**
- 1367 T. Dierks, E. Rescorla, "[The Transport Layer Security \(TLS\) Protocol Version 1.1](#)," April 2006.
- 1368 **[WS-Policy]**
- 1369 W3C Member Submission, "[Web Services Policy Framework \(WS-Policy\)](#)," April 2006.
- 1370 **[WS-PolicyAttachment]**
- 1371 W3C Member Submission, "[Web Services Policy Attachment \(WS-PolicyAttachment\)](#)," April 2006.
- 1372 **[WS-Security]**
- 1373 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security:](#)  
1374 [SOAP Message Security 1.0 \(WS-Security 2004\)](#)", OASIS Standard 200401, March 2004.
- 1375 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security:](#)  
1376 [SOAP Message Security 1.1 \(WS-Security 2004\)](#)", OASIS Standard 200602, February 2006.
- 1377 **[RTTM]**
- 1378 V. Jacobson, R. Braden, D. Borman, "[TCP Extensions for High Performance](#)", RFC 1323, May  
1379 1992.
- 1380 **[SecurityPolicy]**
- 1381 G. Della-Libra, et. al. "[Web Services Security Policy Language \(WS-SecurityPolicy\)](#)", July 2005
- 1382 **[SecureConversation]**
- 1383 S. Anderson, et al, "[Web Services Secure Conversation Language \(WS-SecureConversation\)](#)," February  
1384 2005.
- 1385 **[Trust]**
- 1386 S. Anderson, et al, "[Web Services Trust Language \(WS-Trust\)](#)," February 2005.

## 1387 **A. Schema**

1388 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-  
1389 Schema Part2] is located at:

1390 <http://docs.oasis-open.org/ws-rx/wsrn/200604/wsrn-1.1-schema-200604.xsd>

1391 The following copy is provided for reference.

```

1392 <?xml version="1.0" encoding="UTF-8"?>
1393 <!--
1394 OASIS takes no position regarding the validity or scope of any intellectual
1395 property or other rights that might be claimed to pertain to the
1396 implementation or use of the technology described in this document or the
1397 extent to which any license under such rights might or might not be available;
1398 neither does it represent that it has made any effort to identify any such
1399 rights. Information on OASIS's procedures with respect to rights in OASIS
1400 specifications can be found at the OASIS website. Copies of claims of rights
1401 made available for publication and any assurances of licenses to be made
1402 available, or the result of an attempt made to obtain a general license or
1403 permission for the use of such proprietary rights by implementors or users of
1404 this specification, can be obtained from the OASIS Executive Director.
1405 OASIS invites any interested party to bring to its attention any copyrights,
1406 patents or patent applications, or other proprietary rights which may cover
1407 technology that may be required to implement this specification. Please
1408 address the information to the OASIS Executive Director.
1409 Copyright © OASIS Open 2002-2006. All Rights Reserved.
1410 This document and translations of it may be copied and furnished to others,
1411 and derivative works that comment on or otherwise explain it or assist in its
1412 implementation may be prepared, copied, published and distributed, in whole or
1413 in part, without restriction of any kind, provided that the above copyright
1414 notice and this paragraph are included on all such copies and derivative
1415 works. However, this document itself does not be modified in any way, such as
1416 by removing the copyright notice or references to OASIS, except as needed for
1417 the purpose of developing OASIS specifications, in which case the procedures
1418 for copyrights defined in the OASIS Intellectual Property Rights document must
1419 be followed, or as required to translate it into languages other than English.
1420 The limited permissions granted above are perpetual and will not be revoked by
1421 OASIS or its successors or assigns.
1422 This document and the information contained herein is provided on an "AS IS"
1423 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1424 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1425 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1426 FOR A PARTICULAR PURPOSE.
1427 -->
1428 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1429 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1430 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200604"
1431 targetNamespace="http://docs.oasis-open.org/ws-rx/wsm/200604"
1432 elementFormDefault="qualified" attributeFormDefault="unqualified">
1433   <xs:import namespace="http://www.w3.org/2005/08/addressing"
1434   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1435   <!-- Protocol Elements -->
1436   <xs:complexType name="SequenceType">
1437     <xs:sequence>
1438       <xs:element ref="wsm:Identifier"/>
1439       <xs:element name="MessageNumber" type="wsm:MessageNumberType"/>
1440       <xs:any namespace="##other" processContents="lax" minOccurs="0"
1441 maxOccurs="unbounded"/>
1442     </xs:sequence>
1443     <xs:anyAttribute namespace="##other" processContents="lax"/>
1444   </xs:complexType>
1445   <xs:element name="Sequence" type="wsm:SequenceType"/>
1446   <xs:element name="SequenceAcknowledgement">
1447     <xs:complexType>
1448       <xs:sequence>
1449         <xs:element ref="wsm:Identifier"/>
1450         <xs:choice>
1451           <xs:sequence>
1452             <xs:choice>
1453               <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1454                 <xs:complexType>

```

```

1455         <xs:sequence/>
1456         <xs:attribute name="Upper" type="xs:unsignedLong"
1457 use="required"/>
1458         <xs:attribute name="Lower" type="xs:unsignedLong"
1459 use="required"/>
1460         <xs:anyAttribute namespace="##other" processContents="lax"/>
1461     </xs:complexType>
1462 </xs:element>
1463 <xs:element name="None">
1464     <xs:complexType>
1465         <xs:sequence/>
1466     </xs:complexType>
1467 </xs:element>
1468 </xs:choice>
1469 <xs:element name="Final" minOccurs="0">
1470     <xs:complexType>
1471         <xs:sequence/>
1472     </xs:complexType>
1473 </xs:element>
1474 </xs:sequence>
1475 <xs:element name="Nack" type="xs:unsignedLong"
1476 minOccurs="unbounded"/>
1477 </xs:choice>
1478 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1479 minOccurs="unbounded"/>
1480 </xs:sequence>
1481 <xs:anyAttribute namespace="##other" processContents="lax"/>
1482 </xs:complexType>
1483 </xs:element>
1484 <xs:complexType name="AckRequestedType">
1485     <xs:sequence>
1486         <xs:element ref="wsrm:Identifier"/>
1487         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1488 minOccurs="unbounded"/>
1489     </xs:sequence>
1490     <xs:anyAttribute namespace="##other" processContents="lax"/>
1491 </xs:complexType>
1492 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1493 <xs:complexType name="MessagePendingType">
1494     <xs:sequence>
1495         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1496 minOccurs="unbounded"/>
1497     </xs:sequence>
1498     <xs:attribute name="pending" type="xs:boolean"/>
1499     <xs:anyAttribute namespace="##other" processContents="lax"/>
1500 </xs:complexType>
1501 <xs:element name="MessagePending" type="wsrm:MessagePendingType"/>
1502 <xs:element name="Identifier">
1503     <xs:complexType>
1504         <xs:annotation>
1505             <xs:documentation>
1506                 This type is for elements whose [children] is an anyURI and can have
1507 arbitrary attributes.
1508             </xs:documentation>
1509         </xs:annotation>
1510         <xs:simpleContent>
1511             <xs:extension base="xs:anyURI">
1512                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1513             </xs:extension>
1514         </xs:simpleContent>
1515     </xs:complexType>
1516 </xs:element>
1517 <xs:element name="Address">

```

```

1518     <xs:complexType>
1519         <xs:simpleContent>
1520             <xs:extension base="xs:anyURI">
1521                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1522             </xs:extension>
1523         </xs:simpleContent>
1524     </xs:complexType>
1525 </xs:element>
1526 <xs:complexType name="MakeConnectionType">
1527     <xs:sequence>
1528         <xs:element ref="wsrm:Identifier" minOccurs="0" maxOccurs="1"/>
1529         <xs:element ref="wsrm:Address" minOccurs="0" maxOccurs="1"/>
1530         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1531 maxOccurs="unbounded"/>
1532     </xs:sequence>
1533     <xs:anyAttribute namespace="##other" processContents="lax"/>
1534 </xs:complexType>
1535 <xs:element name="MakeConnection" type="wsrm:MakeConnectionType"/>
1536 <xs:simpleType name="MessageNumberType">
1537     <xs:restriction base="xs:unsignedLong">
1538         <xs:minInclusive value="1"/>
1539         <xs:maxInclusive value="9223372036854775807"/>
1540     </xs:restriction>
1541 </xs:simpleType>
1542 <!-- Fault Container and Codes -->
1543 <xs:simpleType name="FaultCodes">
1544     <xs:restriction base="xs:QName">
1545         <xs:enumeration value="wsrm:SequenceTerminated"/>
1546         <xs:enumeration value="wsrm:UnknownSequence"/>
1547         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1548         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1549         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1550         <xs:enumeration value="wsrm:SequenceClosed"/>
1551         <xs:enumeration value="wsrm:WSRMRequired"/>
1552         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1553     </xs:restriction>
1554 </xs:simpleType>
1555 <xs:complexType name="SequenceFaultType">
1556     <xs:sequence>
1557         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1558         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1559         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1560 maxOccurs="unbounded"/>
1561     </xs:sequence>
1562     <xs:anyAttribute namespace="##other" processContents="lax"/>
1563 </xs:complexType>
1564 <xs:complexType name="DetailType">
1565     <xs:sequence>
1566         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1567 maxOccurs="unbounded"/>
1568     </xs:sequence>
1569     <xs:anyAttribute namespace="##other" processContents="lax"/>
1570 </xs:complexType>
1571 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1572 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1573 <xs:element name="CreateSequenceResponse"
1574 type="wsrm:CreateSequenceResponseType"/>
1575 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1576 <xs:element name="CloseSequenceResponse"
1577 type="wsrm:CloseSequenceResponseType"/>
1578 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1579 <xs:element name="TerminateSequenceResponse"
1580 type="wsrm:TerminateSequenceResponseType"/>

```

```

1581 <xs:complexType name="CreateSequenceType">
1582 <xs:sequence>
1583 <xs:element ref="wsrm:AcksTo"/>
1584 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1585 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1586 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1587 maxOccurs="unbounded">
1588 <xs:annotation>
1589 <xs:documentation>
1590 It is the authors intent that this extensibility be used to
1591 transfer a Security Token Reference as defined in WS-Security.
1592 </xs:documentation>
1593 </xs:annotation>
1594 </xs:any>
1595 </xs:sequence>
1596 <xs:anyAttribute namespace="##other" processContents="lax"/>
1597 </xs:complexType>
1598 <xs:complexType name="CreateSequenceResponseType">
1599 <xs:sequence>
1600 <xs:element ref="wsrm:Identifier"/>
1601 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1602 <xs:element name="IncompleteSequenceBehavior"
1603 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1604 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1605 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1606 maxOccurs="unbounded"/>
1607 </xs:sequence>
1608 <xs:anyAttribute namespace="##other" processContents="lax"/>
1609 </xs:complexType>
1610 <xs:complexType name="CloseSequenceType">
1611 <xs:sequence>
1612 <xs:element ref="wsrm:Identifier"/>
1613 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1614 maxOccurs="unbounded"/>
1615 </xs:sequence>
1616 <xs:anyAttribute namespace="##other" processContents="lax"/>
1617 </xs:complexType>
1618 <xs:complexType name="CloseSequenceResponseType">
1619 <xs:sequence>
1620 <xs:element ref="wsrm:Identifier"/>
1621 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1622 maxOccurs="unbounded"/>
1623 </xs:sequence>
1624 <xs:anyAttribute namespace="##other" processContents="lax"/>
1625 </xs:complexType>
1626 <xs:complexType name="TerminateSequenceType">
1627 <xs:sequence>
1628 <xs:element ref="wsrm:Identifier"/>
1629 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1630 maxOccurs="unbounded"/>
1631 </xs:sequence>
1632 <xs:anyAttribute namespace="##other" processContents="lax"/>
1633 </xs:complexType>
1634 <xs:complexType name="TerminateSequenceResponseType">
1635 <xs:sequence>
1636 <xs:element ref="wsrm:Identifier"/>
1637 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1638 maxOccurs="unbounded"/>
1639 </xs:sequence>
1640 <xs:anyAttribute namespace="##other" processContents="lax"/>
1641 </xs:complexType>
1642 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1643 <xs:complexType name="OfferType">

```

```

1644     <xs:sequence>
1645         <xs:element ref="wsrm:Identifier"/>
1646         <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1647         <xs:element ref="wsrm:Expires" minOccurs="0"/>
1648         <xs:element name="IncompleteSequenceBehavior"
1649 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1650         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1651 maxOccurs="unbounded"/>
1652     </xs:sequence>
1653     <xs:anyAttribute namespace="##other" processContents="lax"/>
1654 </xs:complexType>
1655 <xs:complexType name="AcceptType">
1656     <xs:sequence>
1657         <xs:element ref="wsrm:AcksTo"/>
1658         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1659 maxOccurs="unbounded"/>
1660     </xs:sequence>
1661     <xs:anyAttribute namespace="##other" processContents="lax"/>
1662 </xs:complexType>
1663 <xs:element name="Expires">
1664     <xs:complexType>
1665         <xs:simpleContent>
1666             <xs:extension base="xs:duration">
1667                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1668             </xs:extension>
1669         </xs:simpleContent>
1670     </xs:complexType>
1671 </xs:element>
1672 <xs:simpleType name="IncompleteSequenceBehaviorType">
1673     <xs:restriction base="xs:string">
1674         <xs:enumeration value="DiscardEntireSequence"/>
1675         <xs:enumeration value="DiscardFollowingFirstGap"/>
1676         <xs:enumeration value="NoDiscard"/>
1677     </xs:restriction>
1678 </xs:simpleType>
1679 <xs:element name="UsesSequenceSTR">
1680     <xs:sequence/>
1681     <xs:anyAttribute namespace="##other" processContents="lax"/>
1682 </xs:element>
1683 <xs:element name="UsesSequenceSSL">
1684     <xs:sequence/>
1685     <xs:anyAttribute namespace="##other" processContents="lax"/>
1686 </xs:element>
1687 <xs:element name="UnsupportedElement">
1688     <xs:simpleType>
1689         <xs:restriction base="xs:QName"/>
1690     </xs:simpleType>
1691 </xs:element>
1692 </xs:schema>

```

1693 **B. WSDL**

1694 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1695 <http://docs.oasis-open.org/ws-rx/wsrn/200604/wsd/wsrn-1.1-wsd-200604.wsd>

1696 The following non-normative copy is provided for reference.



```

1697 <?xml version="1.0" encoding="utf-8"?>
1698 <!--
1699 OASIS takes no position regarding the validity or scope of any intellectual
1700 property or other rights that might be claimed to pertain to the
1701 implementation or use of the technology described in this document or the
1702 extent to which any license under such rights might or might not be available;
1703 neither does it represent that it has made any effort to identify any such
1704 rights. Information on OASIS's procedures with respect to rights in OASIS
1705 specifications can be found at the OASIS website. Copies of claims of rights
1706 made available for publication and any assurances of licenses to be made
1707 available, or the result of an attempt made to obtain a general license or
1708 permission for the use of such proprietary rights by implementors or users of
1709 this specification, can be obtained from the OASIS Executive Director.
1710 OASIS invites any interested party to bring to its attention any copyrights,
1711 patents or patent applications, or other proprietary rights which may cover
1712 technology that may be required to implement this specification. Please
1713 address the information to the OASIS Executive Director.
1714 Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
1715 This document and translations of it may be copied and furnished to others,
1716 and derivative works that comment on or otherwise explain it or assist in its
1717 implementation may be prepared, copied, published and distributed, in whole or
1718 in part, without restriction of any kind, provided that the above copyright
1719 notice and this paragraph are included on all such copies and derivative
1720 works. However, this document itself does not be modified in any way, such as
1721 by removing the copyright notice or references to OASIS, except as needed for
1722 the purpose of developing OASIS specifications, in which case the procedures
1723 for copyrights defined in the OASIS Intellectual Property Rights document must
1724 be followed, or as required to translate it into languages other than English.
1725 The limited permissions granted above are perpetual and will not be revoked by
1726 OASIS or its successors or assigns.
1727 This document and the information contained herein is provided on an "AS IS"
1728 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
1729 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT
1730 INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
1731 FOR A PARTICULAR PURPOSE.
1732 -->
1733 <wsdl:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
1734 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1735 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:rm="http://docs.oasis-
1736 open.org/ws-rx/wsrM/200604" xmlns:tns="http://docs.oasis-open.org/ws-
1737 rx/wsrM/200604/wsdl" targetNamespace="http://docs.oasis-open.org/ws-
1738 rx/wsrM/200604/wsdl">
1739     <wsdl:types>
1740         <xs:schema>
1741             <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrM/200604"
1742             schemaLocation="http://docs.oasis-open.org/ws-rx/wsrM/200604/wsrM-1.1-schema-
1743             200604.xsd"/>
1744         </xs:schema>
1745     </wsdl:types>
1746     <wsdl:message name="CreateSequence">
1747         <wsdl:part name="create" element="rm:CreateSequence"/>
1748     </wsdl:message>
1749     <wsdl:message name="CreateSequenceResponse">
1750         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1751     </wsdl:message>
1752     <wsdl:message name="CloseSequence">
1753         <wsdl:part name="close" element="rm:CloseSequence"/>
1754     </wsdl:message>
1755     <wsdl:message name="CloseSequenceResponse">
1756         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1757     </wsdl:message>

```

```

1758     <wsdl:message name="TerminateSequence">
1759         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1760     </wsdl:message>
1761     <wsdl:message name="TerminateSequenceResponse">
1762         <wsdl:part name="terminateResponse"
1763 element="rm:TerminateSequenceResponse"/>
1764     </wsdl:message>
1765     <wsdl:message name="MakeConnection">
1766         <wsdl:part name="makeConnection" element="rm:MakeConnection"/>
1767     </wsdl:message>

1768     <wsdl:portType name="SequenceAbstractPortType">
1769         <wsdl:operation name="CreateSequence">
1770             <wsdl:input message="tns:CreateSequence" wsa:Action="http://docs.oasis-
1771 open.org/ws-rx/wsrn/200604/CreateSequence"/>
1772             <wsdl:output message="tns:CreateSequenceResponse"
1773 wsa:Action="http://docs.oasis-open.org/ws-
1774 rx/wsrn/200604/CreateSequenceResponse"/>
1775         </wsdl:operation>
1776         <wsdl:operation name="CloseSequence">
1777             <wsdl:input message="tns:CloseSequence" wsa:Action="http://docs.oasis-
1778 open.org/ws-rx/wsrn/200604/CloseSequence"/>
1779             <wsdl:output message="tns:CloseSequenceResponse"
1780 wsa:Action="http://docs.oasis-open.org/ws-
1781 rx/wsrn/200604/CloseSequenceResponse"/>
1782         </wsdl:operation>
1783         <wsdl:operation name="TerminateSequence">
1784             <wsdl:input message="tns:TerminateSequence"
1785 wsa:Action="http://docs.oasis-open.org/ws-rx/wsrn/200604/TerminateSequence"/>
1786             <wsdl:output message="tns:TerminateSequenceResponse"
1787 wsa:Action="http://docs.oasis-open.org/ws-
1788 rx/wsrn/200604/TerminateSequenceResponse"/>
1789         </wsdl:operation>
1790         <wsdl:operation name="MakeConnection">
1791             <wsdl:input message="tns:MakeConnection" wsa:Action="http://docs.oasis-
1792 open.org/ws-rx/wsrn/200604/MakeConnection"/>
1793         </wsdl:operation>
1794     </wsdl:portType>
1795 </wsdl:definitions>

```

## 1796 C. Message Examples

### 1797 C.1 Create Sequence

#### 1798 Create Sequence

```
1799 <?xml version="1.0" encoding="UTF-8"?>
1800 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1801 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200604"
1802 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1803   <S:Header>
1804     <wsa:MessageID>
1805       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1806     </wsa:MessageID>
1807     <wsa:To>http://example.com/serviceB/123</wsa:To>
1808     <wsa:Action>http://docs.oasis-open.org/ws-
1809 rx/wsmr/200604/CreateSequence</wsa:Action>
1810     <wsa:ReplyTo>
1811       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1812     </wsa:ReplyTo>
1813   </S:Header>
1814   <S:Body>
1815     <wsmr:CreateSequence>
1816       <wsmr:AcksTo>
1817         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1818       </wsmr:AcksTo>
1819     </wsmr:CreateSequence>
1820   </S:Body>
1821 </S:Envelope>
```

#### 1822 Create Sequence Response

```
1823 <?xml version="1.0" encoding="UTF-8"?>
1824 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1825 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200604"
1826 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1827   <S:Header>
1828     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1829     <wsa:RelatesTo>
1830       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1831     </wsa:RelatesTo>
1832     <wsa:Action>
1833       http://docs.oasis-open.org/ws-rx/wsmr/200604/CreateSequenceResponse
1834     </wsa:Action>
1835   </S:Header>
1836   <S:Body>
1837     <wsmr:CreateSequenceResponse>
1838       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1839     </wsmr:CreateSequenceResponse>
1840   </S:Body>
1841 </S:Envelope>
```

### 1842 C.2 Initial Transmission

1843 The following example WS-ReliableMessaging headers illustrate the message exchange in the above  
1844 figure. The three messages have the following headers; the third message is identified as the last  
1845 message in the Sequence:

#### 1846 Message 1

```

1847 <?xml version="1.0" encoding="UTF-8"?>
1848 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1849 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200604"
1850 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1851   <S:Header>
1852     <wsa:MessageID>
1853       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9e
1854     </wsa:MessageID>
1855     <wsa:To>http://example.com/serviceB/123</wsa:To>
1856     <wsa:From>
1857       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1858     </wsa:From>
1859     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1860     <wsmr:Sequence>
1861       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1862       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1863     </wsmr:Sequence>
1864   </S:Header>
1865   <S:Body>
1866     <!-- Some Application Data -->
1867   </S:Body>
1868 </S:Envelope>

```

## 1869 Message 2

```

1870 <?xml version="1.0" encoding="UTF-8"?>
1871 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1872 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200604"
1873 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1874   <S:Header>
1875     <wsa:MessageID>
1876       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1877     </wsa:MessageID>
1878     <wsa:To>http://example.com/serviceB/123</wsa:To>
1879     <wsa:From>
1880       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1881     </wsa:From>
1882     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1883     <wsmr:Sequence>
1884       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1885       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1886     </wsmr:Sequence>
1887   </S:Header>
1888   <S:Body>
1889     <!-- Some Application Data -->
1890   </S:Body>
1891 </S:Envelope>

```

## 1892 Message 3

```

1893 <?xml version="1.0" encoding="UTF-8"?>
1894 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1895 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200604"
1896 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1897   <S:Header>
1898     <wsa:MessageID>
1899       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1900     </wsa:MessageID>
1901     <wsa:To>http://example.com/serviceB/123</wsa:To>
1902     <wsa:From>
1903       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1904     </wsa:From>
1905     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1906     <wsmr:Sequence>
1907       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>

```

```

1908     <wsrm:MessageNumber>3</wsrm:MessageNumber>
1909     </wsrm:Sequence>
1910     <wsrm:AckRequested>
1911         <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1912     </wsrm:AckRequested>
1913 </S:Header>
1914 <S:Body>
1915     <!-- Some Application Data -->
1916 </S:Body>
1917 </S:Envelope>

```

### 1918 C.3 First Acknowledgement

1919 Message number 2 has not been accepted by the RM Destination due to some transmission error so it  
1920 responds with an acknowledgement for messages 1 and 3:

```

1921 <?xml version="1.0" encoding="UTF-8"?>
1922 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1923 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200604"
1924 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1925   <S:Header>
1926     <wsa:MessageID>
1927       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1928     </wsa:MessageID>
1929     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1930     <wsa:From>
1931       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1932     </wsa:From>
1933     <wsa:Action>
1934       http://docs.oasis-open.org/ws-rx/wsrm/200604/SequenceAcknowledgement
1935     </wsa:Action>
1936     <wsrm:SequenceAcknowledgement>
1937       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1938       <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1939       <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1940     </wsrm:SequenceAcknowledgement>
1941   </S:Header>
1942   <S:Body/>
1943 </S:Envelope>

```

### 1944 C.4 Retransmission

1945 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and  
1946 requests an acknowledgement:

```

1947 <?xml version="1.0" encoding="UTF-8"?>
1948 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1949 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200604"
1950 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1951   <S:Header>
1952     <wsa:MessageID>
1953       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1954     </wsa:MessageID>
1955     <wsa:To>http://example.com/serviceB/123</wsa:To>
1956     <wsa:From>
1957       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1958     </wsa:From>
1959     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1960     <wsrm:Sequence>
1961       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1962       <wsrm:MessageNumber>2</wsrm:MessageNumber>
1963     </wsrm:Sequence>

```

```

1964 <wsrm:AckRequested>
1965 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1966 </wsrm:AckRequested>
1967 </S:Header>
1968 <S:Body>
1969 <!-- Some Application Data -->
1970 </S:Body>
1971 </S:Envelope>

```

## 1972 C.5 Termination

1973 The RM Destination now responds with an acknowledgement for the complete Sequence which can then  
1974 be terminated:

```

1975 <?xml version="1.0" encoding="UTF-8"?>
1976 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1977 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200604"
1978 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1979 <S:Header>
1980 <wsa:MessageID>
1981 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
1982 </wsa:MessageID>
1983 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1984 <wsa:From>
1985 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1986 </wsa:From>
1987 <wsa:Action>
1988 http://docs.oasis-open.org/ws-rx/wsrm/200604/SequenceAcknowledgement
1989 </wsa:Action>
1990 <wsrm:SequenceAcknowledgement>
1991 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1992 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
1993 </wsrm:SequenceAcknowledgement>
1994 </S:Header>
1995 <S:Body/>
1996 </S:Envelope>

```

## 1997 Terminate Sequence

```

1998 <?xml version="1.0" encoding="UTF-8"?>
1999 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2000 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200604"
2001 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2002 <S:Header>
2003 <wsa:MessageID>
2004 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2005 </wsa:MessageID>
2006 <wsa:To>http://example.com/serviceB/123</wsa:To>
2007 <wsa:Action>
2008 http://docs.oasis-open.org/ws-rx/wsrm/200604/TerminateSequence
2009 </wsa:Action>
2010 <wsa:From>
2011 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2012 </wsa:From>
2013 </S:Header>
2014 <S:Body>
2015 <wsrm:TerminateSequence>
2016 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2017 </wsrm:TerminateSequence>
2018 </S:Body>
2019 </S:Envelope>

```

## 2020 Terminate Sequence Response

```

2021 <?xml version="1.0" encoding="UTF-8"?>

```

```
2022 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2023 xmlns:wsm="http://docs.oasis-open.org/ws-rx/wsm/200604"
2024 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2025   <S:Header>
2026     <wsa:MessageID>
2027       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2028     </wsa:MessageID>
2029     <wsa:To>http://example.com/serviceA/789</wsa:To>
2030     <wsa:Action>
2031       http://docs.oasis-open.org/ws-rx/wsm/200604/TerminateSequenceResponse
2032     </wsa:Action>
2033     <wsa:RelatesTo>
2034       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2035     </wsa:RelatesTo>
2036     <wsa:From>
2037       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2038     </wsa:From>
2039   </S:Header>
2040   <S:Body>
2041     <wsm:TerminateSequenceResponse>
2042       <wsm:Identifier>http://Business456.com/RM/ABC</wsm:Identifier>
2043     </wsm:TerminateSequenceResponse>
2044   </S:Body>
2045 </S:Envelope>
```

## 2046 D. State Tables

2047 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2048 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2049 Legend:

2050 The first column of these tables contains the motivating event and has the following format:

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

2051 Where:

- 2052 ● Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as  
2053 described by the specification.
- 2054 ● [source]: indicates the source of the event; one of:
  - 2055 ● [msg] a received message
  - 2056 ● [int]: an internal event such as the firing of a timer
  - 2057 ● [app]: the application
  - 2058 ● [unspec]: the source is unspecified

2059 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}

2060 Where:

- 2061 ● action to take: indicates that the state machine performs the following action. Actions surrounded  
2062 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word  
2063 "Transmit"
- 2064 ● [next state]: indicates the state to which the state machine will advance upon the performance of  
2065 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2066 ● {ref} is a reference to the document section describing the behavior in this cell

2067 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these  
2068 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not  
2069 described in this specification and does not indicate normal protocol operation. Implementations MAY  
2070 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations  
2071 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state  
2072 combinations.



2073 Table 1 RM Source Sequence State Transition Table

Events	Sequence States						
	None	Creating	Created	Rollover	Closing	Closed	Terminating
<b>Create Sequence</b> [unspec] {3.1}	Xmit Create Sequence [Creating] {3.1}	N/A	N/A	N/A	N/A	N/A	N/A
<b>Create Sequence Response</b> [msg] {3.1}		Process Create Sequence Response [Created] {3.1}					
<b>Create Sequence Refused Fault</b> [msg] {3.1}		No action [None] {4.6}					
<b>Send message</b> [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [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}	Xmit message [Same] {2.4}	N/A	N/A
<b>SeqAck (non-final)</b> [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}
<b>Nack</b> [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	<Xmit message(s)> [Same] {3.6}	<Xmit message(s)> [Same] {3.6}	<Xmit message(s)> [Same] {3.6}	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]	No action [Same]
<b>&lt;Close Sequence&gt;</b> [int] {3.2}	N/A		Xmit Close Sequence [Closing] {3.2}	Xmit Close Sequence [Closing] {3.2}	N/A	N/A	N/A
<b>Close Sequence Response</b> [msg] {3.2}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}			No action [Closed] {3.2}	No action [Same] {3.2}	No action [Same] {3.2}

Events	Sequence States						
	None	Creating	Created	Rollover	Closing	Closed	Terminating
<b>SeqAck (final)</b> [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Closed] {3.6}	Process Ack ranges [Closed] {3.6}	Process Ack ranges [Closed] {3.6}	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 [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}	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}	Terminate Sequence [None] {4.2}
<b>Terminate Sequence</b> [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	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] {3.3}
<b>Expires exceeded</b> [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
<b>Invalid Acknowledgment</b> [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}	Generate Invalid Acknowledgment Fault [Same] {4.4}

2074 Table 2 RM Destination Sequence State Transition Table

Events	Sequence States		
	None	Created	Closed
<b>CreateSequence (successful)</b> [msg/int] {3.1}	Xmit Create Sequence Response [Created] {3.1}	N/A	N/A
<b>CreateSequence (unsuccessful)</b> [msg/int] {3.1}	Generate Create Sequence Refused Fault [None] {3.1}	N/A	N/A

Events	Sequence States		
	None	Created	Closed
<b>Message (with message number within range)</b> [msg] {3.2}	Generate Unknown Sequence Fault [Same] {4.3}	Accept Message; <Xmit SeqAck> [Same]	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.2}
<b>Message (with message number outside of range)</b> [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.4}{4.5}	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.2}
<b>&lt;AckRequested&gt;</b> [msg] {3.5}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.5}	Xmit SeqAck+Final [Same] {3.6}
<b>CloseSequence</b> [msg] {3.2}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.2}	Generate Sequence Closed Fault [Same] {4.7}
<b>&lt;CloseSequence autonomously&gt;</b> [int]	N/A	No Action [Closed]	N/A
<b>TerminateSequence</b> [msg] {3.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.3}	Xmit Terminate Sequence Response [None] {3.3}
<b>UnknownSequence Fault</b> [msg] {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}
<b>Invalid Acknowledgement Fault</b> [msg] {4.4}	N/A		
<b>Expires exceeded</b> [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
<b>&lt;Seq Acknowledgement autonomously&gt;</b> [int] {3.6}	N/A	Xmit SeqAck [Same] {3.6}	Xmit SeqAck+Final [Same] {3.6}
<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}

2075 The following two tables apply only if the `MakeConnection` mechanism is utilized.

2076 Table 3 Sending Endpoint Message Transfer Engine

Event	None	Queued n=1	Queued, n>1
Message destined to anon endpoint when channel unavailable [int] {3.7}	Queue message [Queued n=1]	Queue message [Queued n>1]	Queue message [Queued n>1]
MakeConnection [msg] {3.7}		Send message [none]	Xmit message with MessagePending [if n=2 then (Queued n=1) else (Queued n>1)]

2077 Table 4 Receiving Endpoint Message Transfer Engine

Event	None	Polling
Expectation of unreceived message [int, unspecified]	No Action [Polling]	No Action [Same]
Polling trigger [int, unspecified]		Xmit MakeConnection [Polling] (3.7)

## 2078 E. Acknowledgments

2079 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following  
2080 authors:

2081 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),  
2082 Donald Ferguson(IBM), Christopher Ferris-Editor(BM), Tom Freund(IBM), Mary Ann Hondo(IBM),  
2083 John Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft),  
2084 Amelia Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don  
2085 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),  
2086 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony  
2087 Storey(IBM).

2088 The following individuals have provided invaluable input into the initial contribution:

2089 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen Brown  
2090 (Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM), Paul  
2091 Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott  
2092 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal  
2093 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter  
2094 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish  
2095 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2096 The following individuals were members of the committee during the development of this specification:

2097 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben Malek  
2098 (Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubez(Layer 7), Doug Bunting(Sun), Lloyd Burch  
2099 (Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul Cotton  
2100 (Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques Durand  
2101 (Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2), Robert  
2102 Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair Green(Choreology),  
2103 Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu), Chamikara Jayalath  
2104 (WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul Knight(Nortel), Dan  
2105 Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt Lovett(IBM), Ashok Malhotra  
2106 (Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff Mischkinsky(Oracle), Nilo Mitra  
2107 (Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku Nishiyama(Hitachi), Dave Orchard  
2108 (BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert Pilz(BEA), Martin Raepple(SAP),  
2109 Eric Rajkovic(Oracle), Stefan Rossmann(SAP), Tom Rutt(Fujitsu), Rich Salz(IBM), Shivajee  
2110 Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von Riegen(SAP), Pete Wenzel(Sun), Steve  
2111 Winkler(SAP), Ümit Yalçınalp(SAP), Nobuyuki Yamamoto(Hitachi).

## 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

Rev	Date	By Whom	What
wd-08	2005-12-27	Gilbert Pilz	Moved schema and WSDL files to their own artifacts. Converted source document to 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



Rev	Date	By Whom	What
wd-11	2006-03-08	Doug Davis	Issue 100 applied
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

Rev	Date	By Whom	What
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
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

## 2113 G. Notices

2114 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2115 might be claimed to pertain to the implementation or use of the technology described in this document or  
2116 the extent to which any license under such rights might or might not be available; neither does it represent  
2117 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
2118 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
2119 available for publication and any assurances of licenses to be made available, or the result of an attempt  
2120 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
2121 users of this specification, can be obtained from the OASIS Executive Director.

2122 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
2123 other proprietary rights which may cover technology that may be required to implement this specification.  
2124 Please address the information to the OASIS Executive Director.

2125 Copyright (C) OASIS Open (2006). All Rights Reserved.

2126 This document and translations of it may be copied and furnished to others, and derivative works that  
2127 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
2128 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
2129 this paragraph are included on all such copies and derivative works. However, this document itself may  
2130 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
2131 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
2132 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
2133 into languages other than English.

2134 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2135 or assigns.

2136 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2137 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2138 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
2139 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.