



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

3 **Committee Draft 05**

4 **1 February 2007**

5 **Specification URIs:**

6 **This Version:**

7 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-cd-05.pdf>

8 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-cd-05.html>

9 **Previous Version:**

10 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-spec-cd-04.pdf>

11 <http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-spec-cd-04.html>

12 **Latest Version:**

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

14 **Latest Approved Version:**

15 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-cd-05.pdf>

16 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-cd-05.html>

17 **Technical Committee:**

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

19 **Chairs:**

20 Paul Fremantle <paul@wso2.com>

21 Sanjay Patil <sanjay.patil@sap.com>

22 **Editors:**

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

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

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

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

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

28 **Related Work:**

29 This specification replaces or supercedes:

- 30
 - WS-ReliableMessaging v1.0

31 **Declared XML Namespaces:**

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

33 **Abstract:**

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

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

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

49 **Status:**

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

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

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

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

63 Notices

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

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

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

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

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

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

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

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

103 The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be
104 used only to refer to the organization and its official outputs. OASIS welcomes reference to, and
105 implementation and use of, specifications, while reserving the right to enforce its marks against misleading
106 uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

107 Table of Contents

108	1	Introduction	6
109	1.1	Terminology.....	6
110	1.2	Namespace	7
111	1.3	Conformance.....	7
112	2	Reliable Messaging Model.....	8
113	2.1	Glossary	8
114	2.2	Protocol Preconditions.....	9
115	2.3	Protocol Invariants.....	10
116	2.4	Delivery Assurances.....	10
117	2.5	Example Message Exchange	11
118	3	RM Protocol Elements	13
119	3.1	Considerations on the Use of Extensibility Points.....	13
120	3.2	Considerations on the Use of "Piggy-Backing"	13
121	3.3	Composition with WS-Addressing	13
122	3.4	Sequence Creation.....	14
123	3.5	Closing A Sequence.....	18
124	3.6	Sequence Termination	20
125	3.7	Sequences.....	22
126	3.8	Request Acknowledgement.....	23
127	3.9	Sequence Acknowledgement.....	24
128	4	Faults	27
129	4.1	SequenceFault Element	28
130	4.2	Sequence Terminated	29
131	4.3	Unknown Sequence	29
132	4.4	Invalid Acknowledgement.....	30
133	4.5	Message Number Rollover	30
134	4.6	Create Sequence Refused	31
135	4.7	Sequence Closed	31
136	4.8	WSRM Required.....	32
137	5	Security Threats and Countermeasures.....	33
138	5.1	Threats and Countermeasures.....	33
139	5.2	Security Solutions and Technologies	35
140	6	Securing Sequences.....	39
141	6.1	Securing Sequences Using WS-Security	39
142	6.2	Securing Sequences Using SSL/TLS.....	40
143	7	References.....	42
144	7.1	Normative	42
145	7.2	Non-Normative	43
146		Appendix A. Schema.....	45
147		Appendix B. WSDL	50
148		Appendix C. Message Examples	52
149		Appendix C.1 Create Sequence	52
150		Appendix C.2 Initial Transmission	52
151		Appendix C.3 First Acknowledgement	54

152	Appendix C.4 Retransmission	54
153	Appendix C.5 Termination	55
154	Appendix D. State Tables	57
155	Appendix A. Acknowledgments	62
156	Appendix B. Revision History.....	63

1 Introduction

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

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

1.1 Terminology

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

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

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

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

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

1.2 Namespace

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

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

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

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

Table 1

Prefix	Namespace
S	(Either SOAP 1.1 or 1.2)
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
wsrm	http://docs.oasis-open.org/ws-rx/wsrn/200702
wsa	http://www.w3.org/2005/08/addressing
wsam	http://www.w3.org/2007/02/addressing/metadata
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

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

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

1.3 Conformance

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

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

2 Reliable Messaging Model

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

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

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

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

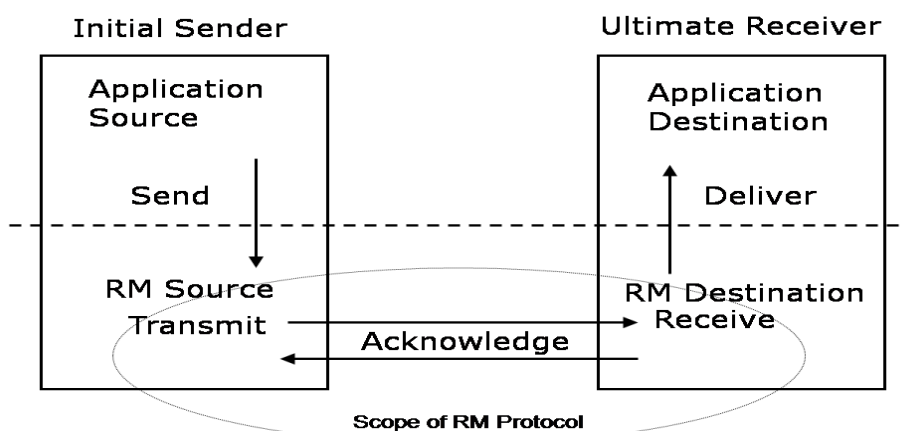


Figure 1: Reliable Messaging Model

2.1 Glossary

The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

272 2.2 Protocol Preconditions

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

- 275 • For any single message exchange the RM Source MUST have an endpoint reference that uniquely
276 identifies the RM Destination Endpoint.
- 277 • The RM Source MUST have successfully created a Sequence with the RM Destination.
- 278 • The RM Source MUST be capable of formulating messages that adhere to the RM Destination's
279 policies.
- 280 • If a secure exchange of messages is REQUIRED, then the RM Source and RM Destination MUST have
281 a security context.

2.3 Protocol Invariants

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

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

2.4 Delivery Assurances

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

AtLeastOnce

Each message is to be delivered at least once, or else an error MUST be raised by the RM Source and/or RM Destination. The requirement on an RM Source is that it SHOULD retry transmission of every message sent by the Application Source until it receives an acknowledgement from the RM Destination. The requirement on the RM Destination is that it SHOULD retry the transfer to the Application Destination of any message that it accepts from the RM Source, until that message has been successfully delivered. There is no requirement for the RM Destination to apply duplicate message filtering.

AtMostOnce

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

ExactlyOnce

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

InOrder

Messages from each individual sequence are to be delivered in the same order they have been sent by the Application Source. The requirement on an RM Source is that it MUST ensure that the ordinal position of each message in the sequence (as indicated by a message sequence number) is consistent with the

order in which the messages have been sent from the Application Source. The requirement on the RM Destination is that it MUST deliver received messages for each sequence in the order indicated by the message numbering. This DeliveryAssurance can be used in combination with any of the AtLeastOnce, AtMostOnce or ExactlyOnce assertions, and the requirements of those assertions MUST also be met. In particular if the AtLeastOnce or ExactlyOnce assertion applies and the RM Destination detects a gap in the sequence then the RM Destination MUST NOT deliver any subsequent messages from that sequence until the missing messages are received or until the sequence is closed.

2.5 Example Message Exchange

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

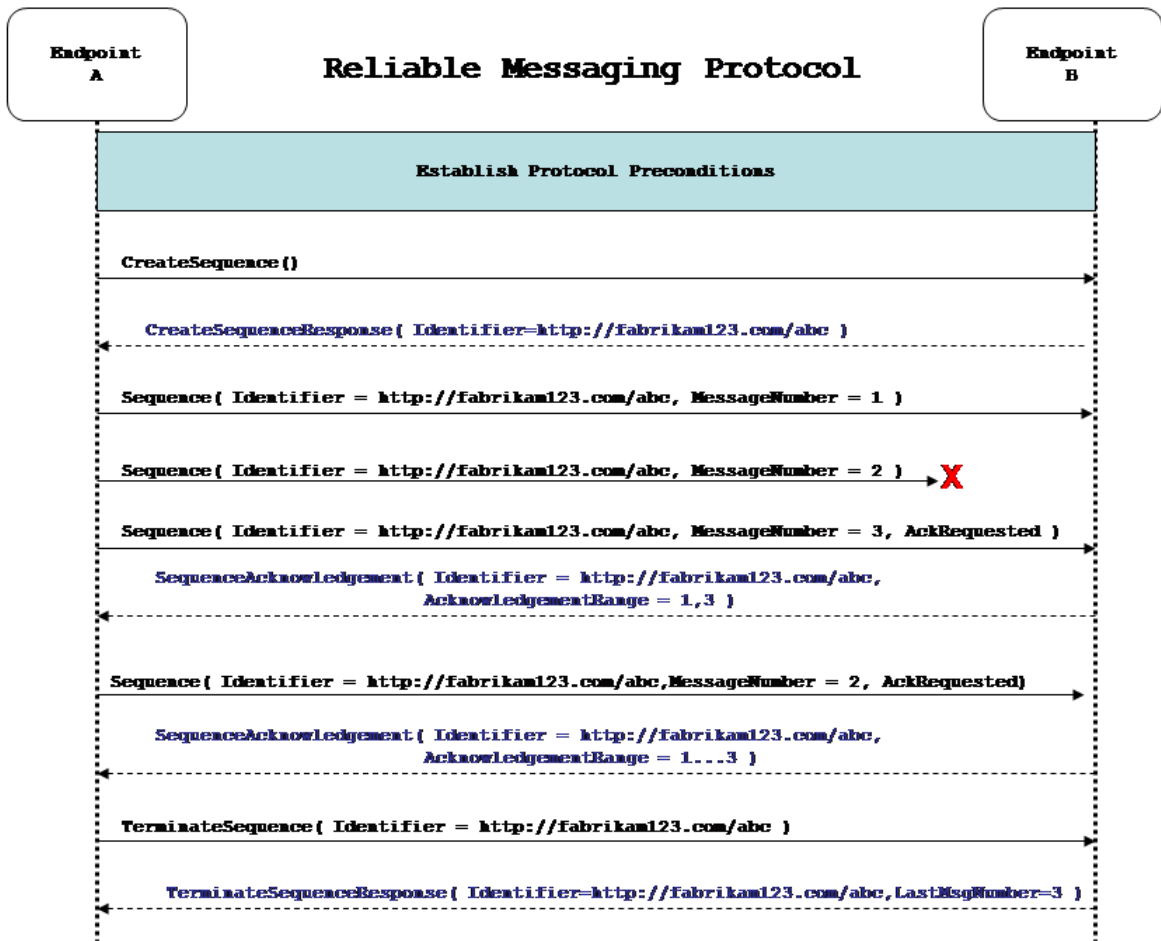


Figure 2: The WS-ReliableMessaging Protocol

1. The protocol preconditions are established. These include policy exchange, endpoint resolution, and establishing trust.
2. The RM Source requests creation of a new Sequence.
3. The RM Destination creates a new Sequence and returns its unique identifier.
4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1. In the figure above, the RM Source sends 3 messages in the Sequence.
5. The 2nd message in the Sequence is lost in transit.

- 340 6. The 3rd message is the last in this Sequence and the RM Source includes an `AckRequested`
341 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.
- 342 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the
343 RM Source's `AckRequested` header.
- 344 8. The RM Source retransmits the unacknowledged message with MessageNumber 2. This is a new
345 message from the perspective of the underlying transport, but it has the same Sequence Identifier
346 and MessageNumber so the RM Destination can recognize it as a duplicate of the earlier message,
347 in case the original and retransmitted messages are both Received. The RM Source includes an
348 `AckRequested` header in the retransmitted message so the RM Destination will expedite an
349 acknowledgement.
- 350 9. The RM Destination Receives the second transmission of the message with MessageNumber 2 and
351 acknowledges receipt of message numbers 1, 2, and 3.
- 352 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to the
353 RM Destination indicating that the Sequence is completed. The `TerminateSequence` message
354 indicates that message number 3 was the last message in the Sequence. The RM Destination then
355 reclaims any resources associated with the Sequence.
- 356 11. The RM Destination Receives the `TerminateSequence` message indicating that the RM Source will
357 not be sending any more messages. The RM Destination sends a `TerminateSequenceResponse`
358 message to the RM Source and reclaims any resources associated with the Sequence.
- 359 The RM Source will expect to Receive Acknowledgements from the RM Destination during the course of
360 a message exchange at occasions described in Section 3 below. Should an Acknowledgement not be
361 Received in a timely fashion, the RM Source MUST re-transmit the message since either the message or
362 the associated Acknowledgement might have been lost. Since the nature and dynamic characteristics of
363 the underlying transport and potential intermediaries are unknown in the general case, the timing of re-
364 transmissions cannot be specified. Additionally, over-aggressive re-transmissions have been
365 demonstrated to cause transport or intermediary flooding which are counterproductive to the intention of
366 providing a reliable exchange of messages. Consequently, implementers are encouraged to utilize
367 adaptive mechanisms that dynamically adjust re-transmission time and the back-off intervals that are
368 appropriate to the nature of the transports and intermediaries envisioned. For the case of TCP/IP
369 transports, a mechanism similar to that described as RTTM in RFC 1323 [[RTTM](#)] SHOULD be
370 considered.
- 371 Now that the basic model has been outlined, the details of the elements used in this protocol are now
372 provided in Section 3.

3 RM Protocol Elements

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

3.1 Considerations on the Use of Extensibility Points

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

3.2 Considerations on the Use of "Piggy-Backing"

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

3.3 Composition with WS-Addressing

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

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

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

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

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

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

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

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

3.4 Sequence Creation

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

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

The following exemplar defines the `CreateSequence` syntax:

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

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

`/wsrm:CreateSequence`

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

`/wsrm:CreateSequence/wsrm:AcksTo`

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

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

`/wsrm:CreateSequence/wsrm:Expires`

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

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

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

455 element.

456 `/wsrm:CreateSequence/wsrm:Offer`

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

459 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier`

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

462 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier/@{any}`

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

465 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Endpoint`

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

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

474 The Offer of an Endpoint containing the "http://www.w3.org/2005/08/addressing/anonymous" IRI as its
475 address is problematic due to the inability of a source to connect to this address and retry
476 unacknowledged messages (as described in Section 2.3). Note that this specification does not define any
477 mechanisms for providing this assurance. In the absence of an extension that addresses this issue, an
478 RM Destination MUST NOT accept (via the `/wsrm:CreateSequenceResponse/wsrm:Accept`
479 element described below) an Offer that contains the "http://www.w3.org/2005/08/addressing/anonymous"
480 IRI as its address.

481 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Expires`

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

485 `/wsrm:CreateSequence/wsrm:Offer/wsrm:Expires/@{any}`

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

488 `/wsrm:CreateSequence/wsrm:Offer/wsrm:IncompleteSequenceBehavior`

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

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

495 A value of "DiscardFollowingFirstGap" indicates that messages in the Sequence beyond the first gap

496 MUST be discarded when there are one or more gaps in the final `SequenceAcknowledgement`.

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

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

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

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

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

505 `/wsrm:CreateSequence/{any}`

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

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

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

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

515 The following exemplar defines the `CreateSequenceResponse` syntax:

```
516 <wsrm:CreateSequenceResponse ...>
517   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
518   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
519   <wsrm:IncompleteSequenceBehavior>
520     wsrm:IncompleteSequenceBehaviorType
521   </wsrm:IncompleteSequenceBehavior> ?
522   <wsrm:Accept ...>
523     <wsrm:AcksTo wsa:EndpointReferenceType </wsrm:AcksTo>
524     ...
525   </wsrm:Accept> ?
526   ...
527 </wsrm:CreateSequenceResponse>
```

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

529 `/wsrm:CreateSequenceResponse`

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

533 `/wsrm:CreateSequenceResponse/wsrm:Identifier`

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

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

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

540 /wsrm:CreateSequenceResponse/wsrm:Expires

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

549 /wsrm:CreateSequenceResponse/wsrm:Expires/@{any}

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

552 /wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior

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

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

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

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

563 /wsrm:CreateSequenceResponse/wsrm:Accept

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

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

569 /wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo

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

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

578 /wsrm:CreateSequenceResponse/wsrm:Accept/{any}

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

581 /wsrm:CreateSequenceResponse/wsrm:Accept/@{any}

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

584 `/wsrm:CreateSequenceResponse/{any}`

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

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

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

590 3.5 Closing A Sequence

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

596 If the RM Source wishes to close the Sequence, then it sends a `CloseSequence` element, in the body of
597 a message, to the RM Destination. This message indicates that the RM Destination MUST NOT accept
598 any new messages for the specified Sequence, other than those already accepted at the time the
599 `CloseSequence` element is interpreted by the RM Destination. Upon receipt of this message, or
600 subsequent to the RM Destination closing the Sequence of its own volition, the RM Destination MUST
601 include a final `SequenceAcknowledgement` (within which the RM Destination MUST include the `Final`
602 element) header block on any messages associated with the Sequence destined to the RM Source,
603 including the `CloseSequenceResponse` message or on any Sequence fault Transmitted to the RM
604 Source.

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

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

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

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

624 The following exemplar defines the `CloseSequence` syntax:

```

625 <wsrm:CloseSequence ...>
626   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
627   <wsrm:LastMsgNumber> wsrm:MessageNumberType </wsrm:LastMsgNumber> ?
628   ...
629 </wsrm:CloseSequence>

```

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

631 `/wsrm:CloseSequence`

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

635 `/wsrm:CloseSequence/wsrm:Identifier`

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

639 `/wsrm:CloseSequence/wsrm:LastMessageNumber`

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

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

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

646 `/wsrm:CloseSequence/{any}`

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

649 `/wsrm:CloseSequence@{any}`

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

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

654 The following exemplar defines the `CloseSequenceResponse` syntax:

```

655 <wsrm:CloseSequenceResponse ...>
656   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
657   ...
658 </wsrm:CloseSequenceResponse>

```

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

660 `/wsrm:CloseSequenceResponse`

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

663 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

667 /wsrm:CloseSequenceResponse/wsrm:Identifier/@{any}

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

670 /wsrm:CloseSequenceResponse/{any}

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

673 /wsrm:CloseSequenceResponse@{any}

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

676 3.6 Sequence Termination

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

684 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM
685 Source SHOULD include the `LastMsgNumber` element in any `TerminateSequence` messages it sends.
686 The RM Destination can use this information, for example, to implement the behavior indicated by
687 /wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior. The value of the
688 `LastMsgNumber` element in the `TerminateSequence` message MUST be equal to the value of the
689 `LastMsgNumber` element in any `CloseSequence` message(s) sent by the RM Source for the same
690 Sequence.

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

695 The following exemplar defines the `TerminateSequence` syntax:

```
696 <wsrm:TerminateSequence ...>  
697   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
698   <wsrm>LastMsgNumber> wsrm:MessageNumberType </wsrm>LastMsgNumber> ?  
699   ...  
700 </wsrm:TerminateSequence>
```

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

702 /wsrm:TerminateSequence

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

708 This element MAY also be sent by the RM Destination to indicate that it has unilaterally terminated the
709 Sequence. Upon sending this message the RM Destination MUST NOT accept any additional messages

710 (with the exception of the corresponding `TerminateSequenceResponse`) for this Sequence. Upon
711 receipt of a `TerminateSequence` the RM Source MUST NOT send any additional messages (with the
712 exception of the corresponding `TerminateSequenceResponse`) for this Sequence.

713 `/wsrm:TerminateSequence/wsrm:Identifier`

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

717 `/wsrm:TerminateSequence/wsrm:LastMsgNumber`

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

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

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

724 `/wsrm:TerminateSequence/{any}`

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

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

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

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

732 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
733 <wsrm:TerminateSequenceResponse ...>  
734   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
735   ...  
736 </wsrm:TerminateSequenceResponse>
```

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

738 `/wsrm:TerminateSequenceResponse`

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

742 `/wsrm:TerminateSequenceResponse/wsrm:Identifier`

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

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

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

749 `/wsrm:TerminateSequenceResponse/{any}`

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

752 /wsrm:TerminateSequenceResponse/{any}

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

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

758 3.7 Sequences

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

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

766 A following exemplar defines its syntax:

```
767 <wsrm:Sequence ...>  
768   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
769   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
770   ...  
771 </wsrm:Sequence>
```

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

773 /wsrm:Sequence

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

780 /wsrm:Sequence/wsrm:Identifier

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

784 /wsrm:Sequence/wsrm:Identifier/{any}

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

787 /wsrm:Sequence/wsrm:MessageNumber

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

792 /wsrm:Sequence/{any}

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

795 /wsrm:Sequence/@{any}

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

798 The following example illustrates a Sequence header block.

```
799 <wsrm:Sequence>  
800   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
801   <wsrm:MessageNumber>10</wsrm:MessageNumber>  
802 </wsrm:Sequence>
```

803 3.8 Request Acknowledgement

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

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

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

818 The following exemplar defines its syntax:

```
819 <wsrm:AckRequested ...>  
820   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
821   ...  
822 </wsrm:AckRequested>
```

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

824 /wsrm:AckRequested

825 This element requests an Acknowledgement for the identified Sequence.

826 /wsrm:AckRequested/wsrm:Identifier

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

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

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

833 /wsrm:AckRequested/{any}

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

836 /wsrm:AckRequested/@{any}

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

839 3.9 Sequence Acknowledgement

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

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

850 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
851 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
852 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
853 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
854 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
855 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
856 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`
857 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination
858 SHOULD respond on the protocol binding-specific back-channel provided by the Received message
859 containing the `AckRequested` header block.

860 The following exemplar defines its syntax:

```
861 <wsrm:SequenceAcknowledgement ...>
862   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
863   [ [ [ <wsrm:AcknowledgementRange ...
864       Upper="wsrm:MessageNumberType"
865       Lower="wsrm:MessageNumberType"/> +
866       | <wsrm:None/> ]
867     <wsrm:Final/> ? ]
868   | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]
869   ...
870 </wsrm:SequenceAcknowledgement>
```

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

873 /wsrm:SequenceAcknowledgement

874 This element contains the Sequence Acknowledgement information.

875 /wsrm:SequenceAcknowledgement/wsrm:Identifier

876 An RM Destination that includes a `SequenceAcknowledgement` header block in a SOAP envelope
877 MUST include this element in that header block. The RM Destination MUST set the value of this element

878 to the absolute URI (conformant with RFC3986) that uniquely identifies the Sequence. The RM
879 Destination MUST NOT include multiple `SequenceAcknowledgement` header blocks that share the
880 same value for `Identifier` within the same SOAP envelope.

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

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

884 `/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange`

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

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

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

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

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

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

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

899 `/wsrm:SequenceAcknowledgement/wsrm:None`

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

904 `/wsrm:SequenceAcknowledgement/wsrm:Final`

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

911 `/wsrm:SequenceAcknowledgement/wsrm:Nack`

912 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If
913 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing the
914 `MessageNumber` of an unreceived message in a Sequence. The RM Destination MUST NOT include a
915 `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of
916 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the
917 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`
918 containing a `Nack` for a message that it has previously acknowledged within an
919 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing
920 a `Nack` for a message that has previously been acknowledged within an `AcknowledgementRange`.

921 /wsrm:SequenceAcknowledgement/{any}

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

924 /wsrm:SequenceAcknowledgement/@{any}

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

927 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
929 <wsrm:SequenceAcknowledgement>  
930   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
931   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
932 </wsrm:SequenceAcknowledgement>
```

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

```
935 <wsrm:SequenceAcknowledgement>  
936   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
937   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
938   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
939   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
940 </wsrm:SequenceAcknowledgement>
```

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

```
942 <wsrm:SequenceAcknowledgement>  
943   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
944   <wsrm:Nack>3</wsrm:Nack>  
945 </wsrm:SequenceAcknowledgement>
```

4 Faults

Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. Create Sequence Refused is a possible fault reply for this operation. Unknown Sequence is a fault generated by Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences are detected. `WSRMRequired` is a fault generated an RM Destination that requires the use of WS-RM on a Received message that did not use the protocol. All other faults in this section relate to known Sequences. Destinations that generate faults related to known sequences SHOULD transmit those faults. If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages. Entities that generate WS-ReliableMessaging faults MUST include as the [action] property the default fault action IRI defined below. The value from the W3C Recommendation is below for informational purposes:

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

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

The definitions of faults use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

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

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

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

```
<S:Envelope>
  <S:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-rx/wsrn/200702/fault
    </wsa:Action>
    <!-- Headers elided for brevity. -->
  </S:Header>
  <S:Body>
    <S:Fault>
      <S:Code>
        <S:Value> [Code] </S:Value>
        <S:Subcode>
          <S:Value> [Subcode] </S:Value>
        </S:Subcode>
      </S:Code>
      <S:Reason>
        <S:Text xml:lang="en"> [Reason] </S:Text>
      </S:Reason>
      <S:Detail>
        [Detail]
      </S:Detail>
    </S:Fault>
  </S:Body>
</S:Envelope>
```

```

990     </S:Detail>
991   </S:Fault>
992 </S:Body>
993 </S:Envelope>

```

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

```

996 <S11:Envelope>
997   <S11:Header>
998     <wsrm:SequenceFault>
999       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1000       <wsrm:Detail> [Detail] </wsrm:Detail>
1001       ...
1002     </wsrm:SequenceFault>
1003     <!-- Headers elided for brevity. -->
1004   </S11:Header>
1005   <S11:Body>
1006     <S11:Fault>
1007       <faultcode> [Code] </faultcode>
1008       <faultstring> [Reason] </faultstring>
1009     </S11:Fault>
1010   </S11:Body>
1011 </S11:Envelope>

```

1012 The properties bind to a SOAP 1.1 fault as follows when the fault is generated as a result of processing a
 1013 CreateSequence request message:

```

1014 <S11:Envelope>
1015   <S11:Body>
1016     <S11:Fault>
1017       <faultcode> [Subcode] </faultcode>
1018       <faultstring> [Reason] </faultstring>
1019     </S11:Fault>
1020   </S11:Body>
1021 </S11:Envelope>

```

1022 4.1 SequenceFault Element

1023 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during the
 1024 reliable messaging specific processing of a message belonging to a Sequence. WS-ReliableMessaging
 1025 nodes MUST use the `SequenceFault` container only in conjunction with the SOAP 1.1 fault mechanism.
 1026 WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in conjunction with the
 1027 SOAP 1.2 binding.

1028 The following exemplar defines its syntax:

```

1029 <wsrm:SequenceFault ...>
1030   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
1031   <wsrm:Detail> ... </wsrm:Detail> ?
1032   ...
1033 </wsrm:SequenceFault>

```

1034 The following describes the content model of the `SequenceFault` element.

1035 /wsrm:SequenceFault

1036 This is the element containing Sequence information for WS-ReliableMessaging

1037 /wsrm:SequenceFault/wsrm:FaultCode

1038 WS-ReliableMessaging nodes that generate a `SequenceFault` MUST set the value of this element to a

1039 qualified name from the set of fault [Subcodes] defined below.

1040 /wsrm:SequenceFault/wsrm:Detail

1041 This element, if present, carries application specific error information related to the fault being described.

1042 /wsrm:SequenceFault/wsrm:Detail/{any}

1043 The application specific error information related to the fault being described.

1044 /wsrm:SequenceFault/wsrm:Detail/@{any}

1045 The application specific error information related to the fault being described.

1046 /wsrm:SequenceFault/{any}

1047 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
1048 to be passed.

1049 /wsrm:SequenceFault/@{any}

1050 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
1051 element.

1052 4.2 Sequence Terminated

1053 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
1054 Endpoint of this decision.

1055 Properties:

1056 [Code] Sender or Receiver

1057 [Subcode] wsrm:SequenceTerminated

1058 [Reason] The Sequence has been terminated due to an unrecoverable error.

1059 [Detail]

1060 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

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

1061 4.3 Unknown Sequence

1062 Properties:

1063 [Code] Sender

1064 [Subcode] wsrm:UnknownSequence

1065 [Reason] The value of wsrm:Identifier is not a known Sequence identifier.

1066 [Detail]

1067 `<wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>`

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

1068 4.4 Invalid Acknowledgement

1069 An example of when this fault is generated is when a message is Received by the RM Source containing
1070 a `SequenceAcknowledgement` covering messages that have not been sent.

1071 [Code] Sender

1072 [Subcode] `wsrm:InvalidAcknowledgement`

1073 [Reason] The `SequenceAcknowledgement` violates the cumulative Acknowledgement invariant.

1074 [Detail]

1075 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

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

1076 4.5 Message Number Rollover

1077 If the condition listed below is reached, the RM Destination MUST generate this fault.

1078 Properties:

1079 [Code] Sender

1080 [Subcode] `wsrm:MessageNumberRollover`

1081 [Reason] The maximum value for `wsrm:MessageNumber` has been exceeded.

1082 [Detail]

```
1083 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
1084 <wsrm:MaxMessageNumber> wsrm:MessageNumberType </wsrm:MaxMessageNumber>
```

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

1085 4.6 Create Sequence Refused

1086 Properties:

1087 [Code] Sender or Receiver

1088 [Subcode] wsrm:CreateSequenceRefused

1089 [Reason] The Create Sequence request has been refused by the RM Destination.

1090 [Detail]

```
1091 xs:any
```

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

1092 4.7 Sequence Closed

1093 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1094 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1095 is closed.

1096 Properties:

1097 [Code] Sender

1098 [Subcode] wsrm:SequenceClosed

1099 [Reason] The Sequence is closed and cannot accept new messages.

1100 [Detail]

1101 `<wsrm:Identifier...> xs:anyURI </wsrm:Identifier>`

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

1102 4.8 WSRM Required

1103 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1104 message that did not use this protocol.

1105 Properties:

1106 [Code] Sender

1107 [Subcode] wsrm:WSRMRequired

1108 [Reason] The RM Destination requires the use of WSRM.

1109 [Detail]

1110 `xs:any`

5 Security Threats and Countermeasures

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

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

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

Implementers are also advised that there is a core tension between security and reliable messaging that can be problematic if not addressed by implementations; one aspect of security is to prevent message replay but one of the invariants of this protocol is to resend messages until they are acknowledged.

Consequently, if the security sub-system processes a message but a failure occurs before the reliable messaging sub-system Receives that message, then it is possible (and likely) that the security sub-system will treat subsequent copies as replays and discard them. At the same time, the reliable messaging sub-system will likely continue to expect and even solicit the missing message(s). Care should be taken to avoid and prevent this condition.

5.1 Threats and Countermeasures

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

5.1.1 Integrity Threats

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

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

5.1.1.1 Countermeasures

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

5.1.2 Resource Consumption Threats

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

5.1.2.1 Countermeasures

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

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

5.1.3 Sequence Spoofing Threats

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

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

5.1.3.1 Sequence Hijacking

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

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

5.1.3.2 Countermeasures

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

1191 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1192 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1193 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1194 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1195 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1196 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1197 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1198 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1199 sequence peer it MUST be able to identify and authenticate the entity that sent the
1200 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1201 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1202 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1203 creation time.

1204 5.2 Security Solutions and Technologies

1205 The security threats described in the previous sections are neither new nor unique. The solutions that
1206 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1207 section maps the facilities provided by common web services security solutions against countermeasures
1208 described in the previous sections.

1209 Before continuing this discussion, however, some examination of the underlying requirements of the
1210 previously described countermeasures is necessary. Specifically it should be noted that the technique
1211 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1212 the issuer of a `CreateSequence` message. Secondly, the RM Destination performs an authorization check
1213 against this authenticated identity and determines if the RM Source is permitted to create Sequences with
1214 the RM Destination. Since the facilities for performing this authorization check (runtime infrastructure,
1215 policy frameworks, etc.) lie completely within the domain of individual implementations, any discussion of
1216 such facilities is considered to be beyond the scope of this specification.

1217 5.2.1 Transport Layer Security

1218 This section describes how the facilities provided by SSL/TLS [RFC 4346] can be used to implement the
1219 countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1220 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

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

1226 5.2.1.1 Model

1227 The basic model for using SSL/TLS is as follows:

- 1228 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1229 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1230 Destination.
- 1231 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an

1232 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1233 synchronous `CreateSequenceResponse` using the session established in (1).

1234 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1235 any and all messages or faults that refer to that Sequence.

1236 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1237 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1238 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1239 5.2.1.2 Countermeasure Implementation

1240 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1241 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1242 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1243 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1244 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1245 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1246 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1247 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1248 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1249 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1250 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1251 establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
1252 using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
1253 to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
1254 Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
1255 Acknowledgement) using BasicAuth.
- 1256 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1257 connection authenticates itself to the party accepting the connection using an X.509 certificate
1258 that is exchanged during the SSL/TLS handshake.

1259 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1260 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1261 Source is authorized to create a Sequence with the RM Destination.

1262 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1263 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1264 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1265 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1266 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1267 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1268 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1269 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1270 to protect that Sequence.

1271 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1272 countermeasures (such as associating specific authentication information with a Sequence) although such
1273 methods are not covered by this document.

1274 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1275 session) are outside the scope of this specification.

5.2.2 SOAP Message Security

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

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

5.2.2.1 Model

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

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

5.2.2.2 Countermeasure Implementation

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

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

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

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

1318 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
1319 amending or renewing contexts) are outside the scope of this specification.

6 Securing Sequences

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

6.1 Securing Sequences Using WS-Security

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

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

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

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

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

`/wsrm:CreateSequence/wsse:SecurityTokenReference`

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

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

1367 the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This
1368 element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source
1369 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1370 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1371 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1372 in WS-Security still applies.

1373 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1374 <wsrm:UsesSequenceSTR ... />
```

1375 The following describes the content model of the `UsesSequenceSTR` header block.

1376 `/wsrm:UsesSequenceSTR`

1377 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1378 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1379 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1380 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1381 Sequence creation.

1382 The following is an example of a `CreateSequence` message using the
1383 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1384 <soap:Envelope ...>  
1385   <soap:Header>  
1386     ...  
1387     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />  
1388     ...  
1389   </soap:Header>  
1390   <soap:Body>  
1391     <wsrm:CreateSequence>  
1392       <wsrm:AcksTo>  
1393         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>  
1394       </wsrm:AcksTo>  
1395       <wsse:SecurityTokenReference>  
1396         ...  
1397       </wsse:SecurityTokenReference>  
1398     </wsrm:CreateSequence>  
1399   </soap:Body>  
1400 </soap:Envelope>
```

1401 6.2 Securing Sequences Using SSL/TLS

1402 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1403 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1404 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1405 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1406 SOAP header block within the `CreateSequence` message.

1407 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1408 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1409 The following describes the content model of the `UsesSequenceSSL` header block.

1410 `/wsrm:UsesSequenceSSL`

1411 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1412 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that

1413 was used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header
1414 with a `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST
1415 understand and correctly implement the functionality described in Section 5.2.1 or else generate a
1416 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1417 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1418 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1419 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1420 `CreateSequenceResponse` message.

1421 7 References

1422 7.1 Normative

1423 [KEYWORDS]

1424 S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University,
1425 March 1997

1426 <http://www.ietf.org/rfc/rfc2119.txt>

1427 [WS-RM Policy]

1428 OASIS WS-RX Technical Committee Draft, "Web Services ReliableMessaging Policy Assertion(WS-RM
1429 Policy)" February 2007

1430 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-cd-05.pdf>

1431 [SOAP 1.1]

1432 W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.

1433 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1434 [SOAP 1.2]

1435 W3C Recommendation, "SOAP Version 1.2 Part 1: Messaging Framework" June 2003.

1436 <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>

1437 [URI]

1438 T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 3986,
1439 MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005.

1440 <http://ietf.org/rfc/rfc3986>

1441 [UUID]

1442 P. Leach, M. Mealling, R. Salz, "A Universally Unique Identifier (UUID) URN Namespace," RFC 4122,
1443 Microsoft, Refactored Networks - LLC, DataPower Technology Inc, July 2005

1444 <http://www.ietf.org/rfc/rfc4122.txt>

1445 [XML]

1446 W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", September 2006.

1447 <http://www.w3.org/TR/REC-xml/>

1448 [XML-ns]

1449 W3C Recommendation, "Namespaces in XML," 14 January 1999.

1450 <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

1451 [XML-Schema Part1]

1452 W3C Recommendation, "XML Schema Part 1: Structures," October 2004.

1453 <http://www.w3.org/TR/xmlschema-1/>

1454 **[XML-Schema Part2]**
1455 W3C Recommendation, "XML Schema Part 2: Datatypes," October 2004.
1456 <http://www.w3.org/TR/xmlschema-2/>
1457 **[XPath 1.0]**
1458 W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16 November 1999.
1459 <http://www.w3.org/TR/xpath>
1460 **[WSDL 1.1]**
1461 W3C Note, "Web Services Description Language (WSDL 1.1)," 15 March 2001.
1462 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
1463 **[WS-Addressing]**
1464 W3C Recommendation, "Web Services Addressing 1.0 - Core", May 2006.
1465 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
1466 W3C Recommendation, "Web Services Addressing 1.0 – SOAP Binding", May 2006.
1467 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>

1468 **7.2 Non-Normative**
1469 **[BSP 1.0]**
1470 WS-I Working Group Draft. "Basic Security Profile Version 1.0," August 2006
1471 <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html>
1472 **[RDDL 2.0]**
1473 Jonathan Borden, Tim Bray, eds. "Resource Directory Description Language (RDDL) 2.0," January 2004
1474 <http://www.openhealth.org/RDDL/20040118/rddl-20040118.html>
1475 **[RFC 2617]**
1476 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "HTTP
1477 Authentication: Basic and Digest Access Authentication," June 1999.
1478 <http://www.ietf.org/rfc/rfc2617.txt>
1479 **[RFC 4346]**
1480 T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," April 2006.
1481 <http://www.ietf.org/rfc/rfc4346.txt>
1482 **[WS-Policy]**
1483 W3C Member Submission, "Web Services Policy Framework (WS-Policy)," April 2006.
1484 <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
1485 **[WS-PolicyAttachment]**
1486 W3C Member Submission, "Web Services Policy Attachment (WS-PolicyAttachment)," April 2006.
1487 [http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)
1488 [20060425/](http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/)

1489 **[WS-Security]**

1490 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1491 SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard 200401, March 2004.

1492 <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

1493 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "OASIS Web Services Security:
1494 SOAP Message Security 1.1 (WS-Security 2004)", OASIS Standard 200602, February 2006.

1495 <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

1496 **[RTTM]**

1497 V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", RFC 1323, May
1498 1992.

1499 <http://www.rfc-editor.org/rfc/rfc1323.txt>

1500 **[SecurityPolicy]**

1501 G. Della-Libra, et. al. "Web Services Security Policy Language (WS-SecurityPolicy)", July 2005

1502 <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>

1503 **[SecureConversation]**

1504 S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February
1505 2005.

1506 <http://schemas.xmlsoap.org/ws/2004/04/sc/>

1507 **[Trust]**

1508 S. Anderson, et al, "Web Services Trust Language (WS-Trust)," February 2005.

1509 <http://schemas.xmlsoap.org/ws/2005/02/trust>

1510 Appendix A. Schema

1511 The normative schema that is defined for WS-ReliableMessaging using [XML-Schema Part1] and [XML-
1512 Schema Part2] is located at:

1513 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-200702.xsd>

1514 The following copy is provided for reference.

```
1515 <?xml version="1.0" encoding="UTF-8"?>
1516 <!-- Copyright (C) OASIS (R) 1993-2007. All Rights Reserved.
1517 OASIS trademark, IPR and other policies apply. -->
1518 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
1519 xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsrm="http://docs.oasis-
1520 open.org/ws-rx/wsrn/200702" targetNamespace="http://docs.oasis-open.org/ws-
1521 rx/wsrn/200702" elementFormDefault="qualified"
1522 attributeFormDefault="unqualified">
1523 <xs:import namespace="http://www.w3.org/2005/08/addressing"
1524 schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
1525 <!-- Protocol Elements -->
1526 <xs:complexType name="SequenceType">
1527 <xs:sequence>
1528 <xs:element ref="wsrm:Identifier"/>
1529 <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
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="Sequence" type="wsrm:SequenceType"/>
1536 <xs:element name="SequenceAcknowledgement">
1537 <xs:complexType>
1538 <xs:sequence>
1539 <xs:element ref="wsrm:Identifier"/>
1540 <xs:choice>
1541 <xs:sequence>
1542 <xs:choice>
1543 <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1544 <xs:complexType>
1545 <xs:sequence/>
1546 <xs:attribute name="Upper" type="xs:unsignedLong"
1547 use="required"/>
1548 <xs:attribute name="Lower" type="xs:unsignedLong"
1549 use="required"/>
1550 <xs:anyAttribute namespace="##other" processContents="lax"/>
1551 </xs:complexType>
1552 </xs:element>
1553 <xs:element name="None">
1554 <xs:complexType>
1555 <xs:sequence/>
1556 </xs:complexType>
1557 </xs:element>
1558 </xs:choice>
1559 <xs:element name="Final" minOccurs="0">
1560 <xs:complexType>
1561 <xs:sequence/>
1562 </xs:complexType>
1563 </xs:element>
1564 </xs:sequence>
1565 <xs:element name="Nack" type="xs:unsignedLong"
1566 maxOccurs="unbounded"/>
```

```

1567         </xs:choice>
1568         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1569 maxOccurs="unbounded"/>
1570     </xs:sequence>
1571     <xs:anyAttribute namespace="##other" processContents="lax"/>
1572 </xs:complexType>
1573 </xs:element>
1574 <xs:complexType name="AckRequestedType">
1575     <xs:sequence>
1576         <xs:element ref="wsrm:Identifier"/>
1577         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1578 maxOccurs="unbounded"/>
1579     </xs:sequence>
1580     <xs:anyAttribute namespace="##other" processContents="lax"/>
1581 </xs:complexType>
1582 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1583 <xs:element name="Identifier">
1584     <xs:complexType>
1585         <xs:annotation>
1586             <xs:documentation>
1587                 This type is for elements whose [children] is an anyURI and can have
1588 arbitrary attributes.
1589             </xs:documentation>
1590         </xs:annotation>
1591         <xs:simpleContent>
1592             <xs:extension base="xs:anyURI">
1593                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1594             </xs:extension>
1595         </xs:simpleContent>
1596     </xs:complexType>
1597 </xs:element>
1598 <xs:element name="Address">
1599     <xs:complexType>
1600         <xs:simpleContent>
1601             <xs:extension base="xs:anyURI">
1602                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1603             </xs:extension>
1604         </xs:simpleContent>
1605     </xs:complexType>
1606 </xs:element>
1607 <xs:simpleType name="MessageNumberType">
1608     <xs:restriction base="xs:unsignedLong">
1609         <xs:minInclusive value="1"/>
1610         <xs:maxInclusive value="9223372036854775807"/>
1611     </xs:restriction>
1612 </xs:simpleType>
1613 <!-- Fault Container and Codes -->
1614 <xs:simpleType name="FaultCodes">
1615     <xs:restriction base="xs:QName">
1616         <xs:enumeration value="wsrm:SequenceTerminated"/>
1617         <xs:enumeration value="wsrm:UnknownSequence"/>
1618         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1619         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1620         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1621         <xs:enumeration value="wsrm:SequenceClosed"/>
1622         <xs:enumeration value="wsrm:WSRMRequired"/>
1623         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1624     </xs:restriction>
1625 </xs:simpleType>
1626 <xs:complexType name="SequenceFaultType">
1627     <xs:sequence>
1628         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1629         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1630         <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:complexType name="DetailType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
<xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
<xs:element name="CreateSequenceResponse"
type="wsrm:CreateSequenceResponseType"/>
<xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
<xs:element name="CloseSequenceResponse"
type="wsrm:CloseSequenceResponseType"/>
<xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
<xs:element name="TerminateSequenceResponse"
type="wsrm:TerminateSequenceResponseType"/>
<xs:complexType name="CreateSequenceType">
  <xs:sequence>
    <xs:element ref="wsrm:AcksTo"/>
    <xs:element ref="wsrm:Expires" minOccurs="0"/>
    <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>
          It is the authors intent that this extensibility be used to
transfer a Security Token Reference as defined in WS-Security.
        </xs:documentation>
      </xs:annotation>
    </xs:any>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:complexType name="CreateSequenceResponseType">
  <xs:sequence>
    <xs:element ref="wsrm:Identifier"/>
    <xs:element ref="wsrm:Expires" minOccurs="0"/>
    <xs:element name="IncompleteSequenceBehavior"
type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
    <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:complexType name="CloseSequenceType">
  <xs:sequence>
    <xs:element ref="wsrm:Identifier"/>
    <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:complexType name="CloseSequenceResponseType">
  <xs:sequence>
    <xs:element ref="wsrm:Identifier"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"

```

```

1695 maxOccurs="unbounded"/>
1696 </xs:sequence>
1697 <xs:anyAttribute namespace="##other" processContents="lax"/>
1698 </xs:complexType>
1699 <xs:complexType name="TerminateSequenceType">
1700 <xs:sequence>
1701 <xs:element ref="wsrm:Identifier"/>
1702 <xs:element name="LastMsgNumber" type="wsrm:MessageNumberType"
1703 minOccurs="0"/>
1704 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1705 maxOccurs="unbounded"/>
1706 </xs:sequence>
1707 <xs:anyAttribute namespace="##other" processContents="lax"/>
1708 </xs:complexType>
1709 <xs:complexType name="TerminateSequenceResponseType">
1710 <xs:sequence>
1711 <xs:element ref="wsrm:Identifier"/>
1712 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1713 maxOccurs="unbounded"/>
1714 </xs:sequence>
1715 <xs:anyAttribute namespace="##other" processContents="lax"/>
1716 </xs:complexType>
1717 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1718 <xs:complexType name="OfferType">
1719 <xs:sequence>
1720 <xs:element ref="wsrm:Identifier"/>
1721 <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1722 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1723 <xs:element name="IncompleteSequenceBehavior"
1724 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1725 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1726 maxOccurs="unbounded"/>
1727 </xs:sequence>
1728 <xs:anyAttribute namespace="##other" processContents="lax"/>
1729 </xs:complexType>
1730 <xs:complexType name="AcceptType">
1731 <xs:sequence>
1732 <xs:element ref="wsrm:AcksTo"/>
1733 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1734 maxOccurs="unbounded"/>
1735 </xs:sequence>
1736 <xs:anyAttribute namespace="##other" processContents="lax"/>
1737 </xs:complexType>
1738 <xs:element name="Expires">
1739 <xs:complexType>
1740 <xs:simpleContent>
1741 <xs:extension base="xs:duration">
1742 <xs:anyAttribute namespace="##other" processContents="lax"/>
1743 </xs:extension>
1744 </xs:simpleContent>
1745 </xs:complexType>
1746 </xs:element>
1747 <xs:simpleType name="IncompleteSequenceBehaviorType">
1748 <xs:restriction base="xs:string">
1749 <xs:enumeration value="DiscardEntireSequence"/>
1750 <xs:enumeration value="DiscardFollowingFirstGap"/>
1751 <xs:enumeration value="NoDiscard"/>
1752 </xs:restriction>
1753 </xs:simpleType>
1754 <xs:element name="UsesSequenceSTR">
1755 <xs:complexType>
1756 <xs:sequence>
1757 <xs:anyAttribute namespace="##other" processContents="lax"/>
1758 </xs:complexType>

```



```
1759 </xs:element>
1760 <xs:element name="UsesSequenceSSL">
1761   <xs:complexType>
1762     <xs:sequence/>
1763     <xs:anyAttribute namespace="##other" processContents="lax"/>
1764   </xs:complexType>
1765 </xs:element>
1766 <xs:element name="UnsupportedElement">
1767   <xs:simpleType>
1768     <xs:restriction base="xs:QName"/>
1769   </xs:simpleType>
1770 </xs:element>
1771 </xs:schema>
```

1772 Appendix B. WSDL

1773 This WSDL describes the WS-RM protocol from the point of view of an RM Destination. In the case where
1774 an endpoint acts both as an RM Destination and an RM Source, note that additional messages may be
1775 present in exchanges with that endpoint.

1776 Also note that this WSDL is intended to describe the internal structure of the WS-RM protocol, and will not
1777 generally appear in a description of a WS-RM-capable Web service. See WS-RM Policy [WS-RM Policy]
1778 for a higher-level mechanism to indicate that WS-RM is engaged.

1779 The normative WSDL 1.1 definition for WS-ReliableMessaging is located at:

1780 <http://docs.oasis-open.org/ws-rx/wsrn/200702/wsd/wsrn-1.1-wsdl-200702.wsdl>

1781 The following non-normative copy is provided for reference.

```
1782 <?xml version="1.0" encoding="utf-8"?>
1783 <!-- Copyright(C) OASIS(R) 1993-2007. All Rights Reserved.
1784 OASIS trademark, IPR and other policies apply. -->
1785 <wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
1786 xmlns:xs="http://www.w3.org/2001/XMLSchema"
1787 xmlns:wsa="http://www.w3.org/2005/08/addressing"
1788 xmlns:wsam="http://www.w3.org/2007/02/addressing/metadata"
1789 xmlns:rm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1790 xmlns:tns="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsd"
1791 targetNamespace="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsd">
1792
1793   <wSDL:types>
1794     <xs:schema
1795       <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1796       schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-schema-
1797       200702.xsd"/>
1798     </xs:schema>
1799   </wSDL:types>
1800
1801   <wSDL:message name="CreateSequence">
1802     <wSDL:part name="create" element="rm:CreateSequence"/>
1803   </wSDL:message>
1804   <wSDL:message name="CreateSequenceResponse">
1805     <wSDL:part name="createResponse" element="rm:CreateSequenceResponse"/>
1806   </wSDL:message>
1807   <wSDL:message name="CloseSequence">
1808     <wSDL:part name="close" element="rm:CloseSequence"/>
1809   </wSDL:message>
1810   <wSDL:message name="CloseSequenceResponse">
1811     <wSDL:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1812   </wSDL:message>
1813   <wSDL:message name="TerminateSequence">
1814     <wSDL:part name="terminate" element="rm:TerminateSequence"/>
1815   </wSDL:message>
1816   <wSDL:message name="TerminateSequenceResponse">
1817     <wSDL:part name="terminateResponse"
1818     element="rm:TerminateSequenceResponse"/>
1819   </wSDL:message>
1820
1821   <wSDL:portType name="SequenceAbstractPortType">
1822     <wSDL:operation name="CreateSequence">
1823       <wSDL:input message="tns:CreateSequence" wsam:Action="http://docs.oasis-
1824       open.org/ws-rx/wsrn/200702/CreateSequence"/>
1825       <wSDL:output message="tns:CreateSequenceResponse"
```

```
1826 wsam:Action="http://docs.oasis-open.org/ws-
1827 rx/wsrn/200702/CreateSequenceResponse"/>
1828 </wsdl:operation>
1829 <wsdl:operation name="CloseSequence">
1830 <wsdl:input message="tns:CloseSequence" wsam:Action="http://docs.oasis-
1831 open.org/ws-rx/wsrn/200702/CloseSequence"/>
1832 <wsdl:output message="tns:CloseSequenceResponse"
1833 wsam:Action="http://docs.oasis-open.org/ws-
1834 rx/wsrn/200702/CloseSequenceResponse"/>
1835 </wsdl:operation>
1836 <wsdl:operation name="TerminateSequence">
1837 <wsdl:input message="tns:TerminateSequence"
1838 wsam:Action="http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence"/>
1839 <wsdl:output message="tns:TerminateSequenceResponse"
1840 wsam:Action="http://docs.oasis-open.org/ws-
1841 rx/wsrn/200702/TerminateSequenceResponse"/>
1842 </wsdl:operation>
1843 </wsdl:portType>
1844
1845 </wsdl:definitions>
```

1846 Appendix C. Message Examples

1847 Appendix C.1 Create Sequence

1848 Create Sequence

```
1849 <?xml version="1.0" encoding="UTF-8"?>
1850 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1851 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1852 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1853   <S:Header>
1854     <wsa:MessageID>
1855       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
1856     </wsa:MessageID>
1857     <wsa:To>http://example.com/serviceB/123</wsa:To>
1858     <wsa:Action>http://docs.oasis-open.org/ws-
1859 rx/wsmr/200702/CreateSequence</wsa:Action>
1860     <wsa:ReplyTo>
1861       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1862     </wsa:ReplyTo>
1863   </S:Header>
1864   <S:Body>
1865     <wsmr>CreateSequence>
1866       <wsmr:AcksTo>
1867         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1868       </wsmr:AcksTo>
1869     </wsmr>CreateSequence>
1870   </S:Body>
1871 </S:Envelope>
```

1872 Create Sequence Response

```
1873 <?xml version="1.0" encoding="UTF-8"?>
1874 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1875 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1876 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1877   <S:Header>
1878     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1879     <wsa:RelatesTo>
1880       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8a7c2eb546817
1881     </wsa:RelatesTo>
1882     <wsa:Action>
1883       http://docs.oasis-open.org/ws-rx/wsmr/200702/CreateSequenceResponse
1884     </wsa:Action>
1885   </S:Header>
1886   <S:Body>
1887     <wsmr>CreateSequenceResponse>
1888       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1889     </wsmr>CreateSequenceResponse>
1890   </S:Body>
1891 </S:Envelope>
```

1892 Appendix C.2 Initial Transmission

1893 The following example WS-ReliableMessaging headers illustrate the message exchange in the above
1894 figure. The three messages have the following headers; the third message is identified as the last
1895 message in the Sequence:

1896 Message 1

```
1897 <?xml version="1.0" encoding="UTF-8"?>
1898 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1899 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1900 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1901   <S:Header>
1902     <wsa:MessageID>
1903       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfcbc9e
1904     </wsa:MessageID>
1905     <wsa:To>http://example.com/serviceB/123</wsa:To>
1906     <wsa:From>
1907       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1908     </wsa:From>
1909     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1910     <wsmr:Sequence>
1911       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1912       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1913     </wsmr:Sequence>
1914   </S:Header>
1915   <S:Body>
1916     <!-- Some Application Data -->
1917   </S:Body>
1918 </S:Envelope>
```

1919 Message 2

```
1920 <?xml version="1.0" encoding="UTF-8"?>
1921 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1922 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1923 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1924   <S:Header>
1925     <wsa:MessageID>
1926       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1927     </wsa:MessageID>
1928     <wsa:To>http://example.com/serviceB/123</wsa:To>
1929     <wsa:From>
1930       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1931     </wsa:From>
1932     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1933     <wsmr:Sequence>
1934       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1935       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1936     </wsmr:Sequence>
1937   </S:Header>
1938   <S:Body>
1939     <!-- Some Application Data -->
1940   </S:Body>
1941 </S:Envelope>
```

1942 Message 3

```
1943 <?xml version="1.0" encoding="UTF-8"?>
1944 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1945 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200702"
1946 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1947   <S:Header>
1948     <wsa:MessageID>
1949       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1950     </wsa:MessageID>
1951     <wsa:To>http://example.com/serviceB/123</wsa:To>
1952     <wsa:From>
1953       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1954     </wsa:From>
```

```

1955 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1956 <wsrm:Sequence>
1957 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1958 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1959 </wsrm:Sequence>
1960 <wsrm:AckRequested>
1961 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1962 </wsrm:AckRequested>
1963 </S:Header>
1964 <S:Body>
1965 <!-- Some Application Data -->
1966 </S:Body>
1967 </S:Envelope>

```

1968 **Appendix C.3 First Acknowledgement**

1969 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1970 responds with an Acknowledgement for messages 1 and 3:

```

1971 <?xml version="1.0" encoding="UTF-8"?>
1972 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1973 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
1974 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1975 <S:Header>
1976 <wsa:MessageID>
1977 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1978 </wsa:MessageID>
1979 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1980 <wsa:From>
1981 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1982 </wsa:From>
1983 <wsa:Action>
1984 http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
1985 </wsa:Action>
1986 <wsrm:SequenceAcknowledgement>
1987 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1988 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1989 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1990 </wsrm:SequenceAcknowledgement>
1991 </S:Header>
1992 <S:Body/>
1993 </S:Envelope>

```

1994 **Appendix C.4 Retransmission**

1995 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
1996 requests an Acknowledgement:

```

1997 <?xml version="1.0" encoding="UTF-8"?>
1998 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1999 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2000 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2001 <S:Header>
2002 <wsa:MessageID>
2003 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2004 </wsa:MessageID>
2005 <wsa:To>http://example.com/serviceB/123</wsa:To>
2006 <wsa:From>
2007 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2008 </wsa:From>
2009 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>

```

```

2010 <wsrm:Sequence>
2011   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2012   <wsrm:MessageNumber>2</wsrm:MessageNumber>
2013 </wsrm:Sequence>
2014 <wsrm:AckRequested>
2015   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2016 </wsrm:AckRequested>
2017 </S:Header>
2018 <S:Body>
2019   <!-- Some Application Data -->
2020 </S:Body>
2021 </S:Envelope>

```

2022 Appendix C.5 Termination

2023 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
 2024 be terminated:

```

2025 <?xml version="1.0" encoding="UTF-8"?>
2026 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2027 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2028 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2029   <S:Header>
2030     <wsa:MessageID>
2031       http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2032     </wsa:MessageID>
2033     <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2034     <wsa:From>
2035       <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2036     </wsa:From>
2037     <wsa:Action>
2038       http://docs.oasis-open.org/ws-rx/wsrn/200702/SequenceAcknowledgement
2039     </wsa:Action>
2040     <wsrm:SequenceAcknowledgement>
2041       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2042       <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2043     </wsrm:SequenceAcknowledgement>
2044   </S:Header>
2045   <S:Body/>
2046 </S:Envelope>

```

2047 Terminate Sequence

```

2048 <?xml version="1.0" encoding="UTF-8"?>
2049 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2050 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2051 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2052   <S:Header>
2053     <wsa:MessageID>
2054       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2055     </wsa:MessageID>
2056     <wsa:To>http://example.com/serviceB/123</wsa:To>
2057     <wsa:Action>
2058       http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequence
2059     </wsa:Action>
2060     <wsa:From>
2061       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2062     </wsa:From>
2063   </S:Header>
2064   <S:Body>
2065     <wsrm:TerminateSequence>
2066       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>

```

```

2067     <wsrm:LastMsgNumber> 3 </wsrm:LastMsgNumber>
2068   </wsrm:TerminateSequence>
2069 </S:Body>
2070 </S:Envelope>

```

2071 Terminate Sequence Response

```

2072 <?xml version="1.0" encoding="UTF-8"?>
2073 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2074   xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200702"
2075   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2076   <S:Header>
2077     <wsa:MessageID>
2078       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2079     </wsa:MessageID>
2080     <wsa:To>http://example.com/serviceA/789</wsa:To>
2081     <wsa:Action>
2082       http://docs.oasis-open.org/ws-rx/wsrn/200702/TerminateSequenceResponse
2083     </wsa:Action>
2084     <wsa:RelatesTo>
2085       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2086     </wsa:RelatesTo>
2087     <wsa:From>
2088       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2089     </wsa:From>
2090   </S:Header>
2091   <S:Body>
2092     <wsrm:TerminateSequenceResponse>
2093       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2094     </wsrm:TerminateSequenceResponse>
2095   </S:Body>
2096 </S:Envelope>

```


2097 Appendix D. State Tables

2098 This appendix specifies the non-normative state transition tables for RM Source and RM Destination.

2099 The state tables describe the lifetime of a sequence in both the RM Source and the RM Destination

2100 Legend:

2101 The first column of these tables contains the motivating event and has the following format:

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

2102 Where:

- 2103 • Event Name: indicates the name of the event. Event Names surrounded by "<>" are optional as
- 2104 described by the specification.
- 2105 • [source]: indicates the source of the event; one of:
- 2106 • [msg] a Received message
- 2107 • [int]: an internal event such as the firing of a timer
- 2108 • [app]: the application
- 2109 • [unspec]: the source is unspecified

2110 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}

2111 Where:

- 2112 • action to take: indicates that the state machine performs the following action. Actions surrounded
- 2113 by "<>" are optional as described by the specification. "Xmit" is used as a short form for the word
- 2114 "Transmit"
- 2115 • [next state]: indicates the state to which the state machine will advance upon the performance of
- 2116 the action. For ease of reading the next state "same" indicates that the state does not change.
- 2117 • {ref} is a reference to the document section describing the behavior in this cell

2118 "N/A" in a cell indicates a state / event combination self-inconsistent with the state machine; should these
2119 conditions occur, it would indicate an implementation error. A blank cell indicates that the behavior is not
2120 described in this specification and does not indicate normal protocol operation. Implementations MAY
2121 generate a Sequence Terminated fault (see section 4.2) in these circumstances. Robust implementations
2122 MUST be able to operate in a stable manner despite the occurrence of unspecified event / state
2123 combinations.

2124 Table 1 RM Source Sequence State Transition Table

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

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
SeqAck (final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Closed] {3.9}	Process Ack ranges [Same]	Process Ack ranges [Same]
Sequence Closed Fault [msg] {4.7}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]
Unknown Sequence Fault [msg] {4.3}			Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
Sequence Terminated Fault [msg] {4.2}	N/A		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}
TerminateSequence [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}
Terminate Sequence [int]	N/A	No action [None] {unspec}	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	Xmit Terminate Sequence [Terminating]	N/A
Terminate Sequence Response [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}				Terminate Sequence [None] {3.6}
Expires exceeded [int]	N/A	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}	Terminate Sequence [None] {3.7}
Invalid Acknowledgement [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

2125 Table 2 RM Destination Sequence State Transition Table

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

Events	Sequence States			
	None	Created	Closed	Terminating
CreateSequence (unsuccessful) [msg/int] {3.4}	Generate Create Sequence Refused Fault [None] {3.4}	N/A	N/A	
Message (with message number within range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Accept Message; <Xmit SeqAck> [Same]	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
Message (with message number outside of range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.7}{4.5}	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
<AckRequested> [msg] {3.8}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.8}	Xmit SeqAck+Final [Same] {3.9}	Generate Sequence Terminated Fault [Same] {4.2}
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
<CloseSequence autonomously> [int]		Xmit CloseSequence with SeqAck+Final [Closed] {3.5}	Xmit CloseSequence with SeqAck+Final [Same] {3.5}	
CloseSequenceResponse [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}		No Action [Closed] {3.5}	Generate Sequence Terminated Fault [Same] {4.2}
TerminateSequence [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}	Xmit Terminate Sequence Response [None] {3.6}
<TerminateSequence autonomously> [int]		Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}	Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}	Xmit TerminateSequence with SeqAck+Final [Terminating] {3.6}
TerminateSequenceResponse [msg]	Generate Unknown Sequence Fault [Same] {4.3}			Terminate Sequence [None]
UnknownSequence Fault [msg] {4.3}		Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}	Terminate Sequence [None] {4.3}
SequenceTerminated Fault [msg] {4.2}		Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.2}	Terminate Sequence [None] {4.3}
Invalid Acknowledgement Fault [msg] {4.4}	N/A			
Expires exceeded [int]	N/A	Terminate Sequence [None]	Terminate Sequence [None]	

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

2126 **Appendix A. Acknowledgments**

2127 This document is based on initial contribution to OASIS WS-RX Technical Committee by the following
2128 authors:

2129 Ruslan Bilorusets(BEA), Don Box(Microsoft), Luis Felipe Cabrera(Microsoft), Doug Davis(IBM),
2130 Donald Ferguson(IBM), Christopher Ferris(IBM), Tom Freund(IBM), Mary Ann Hondo(IBM), John
2131 Ibbotson(IBM), Lei Jin(BEA), Chris Kaler(Microsoft), David Langworthy-Editor(Microsoft), Amelia
2132 Lewis(TIBCO Software), Rodney Limprecht(Microsoft), Steve Lucco(Microsoft), Don
2133 Mullen(TIBCO Software), Anthony Nadalin(IBM), Mark Nottingham(BEA), David Orchard(BEA),
2134 Jamie Roots(IBM), Shivajee Samdarshi(TIBCO Software), John Shewchuk(Microsoft), Tony
2135 Storey(IBM).

2136 The following individuals have provided invaluable input into the initial contribution:

2137 Keith Ballinger(Microsoft), Stefan Batres(Microsoft), Rebecca Bergersen(Iona), Allen
2138 Brown(Microsoft), Michael Conner(IBM), George Copeland(Microsoft), Francisco Curbera(IBM),
2139 Paul Fremantle(IBM), Steve Graham(IBM), Pat Helland(Microsoft), Rick Hill(Microsoft), Scott
2140 Hinkelman(IBM), Tim Holloway(IBM), Efim Hudis(Microsoft), David Ingham(Microsoft), Gopal
2141 Kakivaya(Microsoft), Johannes Klein(Microsoft), Frank Leymann(IBM), Martin Nally(IBM), Peter
2142 Niblett(IBM), Jeffrey Schlimmer(Microsoft), James Snell(IBM), Keith Stobie(Microsoft), Satish
2143 Thatte(Microsoft), Stephen Todd(IBM), Sanjiva Weerawarana(IBM), Roger Wolter(Microsoft).

2144 The following individuals were members of the committee during the development of this specification:

2145 Abbie Barbir(Nortel), Charlton Barreto(Adobe), Stefan Batres(Microsoft), Hamid Ben
2146 Malek(Fujitsu), Andreas Bjarlestam(Ericsson), Toufic Boubbez(Layer 7), Doug Bunting(Sun), Lloyd
2147 Burch(Novell), Steve Carter(Novell), Martin Chapman(Oracle), Dave Chappell(Sonic), Paul
2148 Cotton(Microsoft), Glen Daniels(Sonic), Doug Davis(IBM), Blake Dournaee(Intel), Jacques
2149 Durand(Fujitsu), Colleen Evans(Microsoft), Christopher Ferris(IBM), Paul Fremantle(WSO2),
2150 Robert Freund(Hitachi), Peter Furniss(Erebor), Marc Goodner(Microsoft), Alastair
2151 Green(Choreology), Mike Grogan(Sun), Ondrej Hrebicek(Microsoft), Kazunori Iwasa(Fujitsu),
2152 Chamikara Jayalath(WSO2), Lei Jin(BEA), Ian Jones(BTplc), Anish Karmarkar(Oracle), Paul
2153 Knight(Nortel), Dan Leshchiner(Tibco), Mark Little(JBoss), Lily Liu(webMethods), Matt
2154 Lovett(IBM), Ashok Malhotra(Oracle), Jonathan Marsh(Microsoft), Daniel Millwood(IBM), Jeff
2155 Mischkinsky(Oracle), Nilo Mitra(Ericsson), Peter Niblett(IBM), Duane Nickull(Adobe), Eisaku
2156 Nishiyama(Hitachi), Dave Orchard(BEA), Chouthri Palanisamy(NEC), Sanjay Patil(SAP), Gilbert
2157 Pilz(BEA), Martin Raeppele(SAP), Eric Rajkovic(Oracle), Stefan Rossmannith(SAP), Tom
2158 Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(SAP), Claus von
2159 Riegen(SAP), Pete Wenzel(Sun), Steve Winkler(SAP), Ümit Yalçınalp(SAP), Nobuyuki
2160 Yamamoto(Hitachi).

2161 Appendix B. 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 'optional'/'required' to 'OPTIONAL'/'REQUIRED') added
wd-05	2005-09-30	Anish Karmarkar	i017 (Change NS to http://docs.oasis-open.org/wsrn/200510/)
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

Rev	Date	By Whom	What
wd-07	2005-11-28	Doug Davis	i044
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

Rev	Date	By Whom	What
wd-08	2005-12-27	Doug Davis	i060
wd-08	2005-12-27	Gilbert Pilz	Moved schema and WSDL files to their own artifacts. Converted source document to 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

Rev	Date	By Whom	What
wd-11	2006-03-08	Doug Davis	Issue 091 applied
wd-11	2006-03-08	Doug Davis	Issue 092 applied
wd-11	2006-03-08	Doug Davis	Issue 100 applied
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

Rev	Date	By Whom	What
wd-14	2006-06-12	Doug Davis	Fix for several RFC2119 keywords – per Anish
wd-15	2006-06-12	Doug Davis	Accept all changed, dump WD number
wd-15	2006-06-12	Doug Davis	Move WSDL after Schema
wd-15	2006-06-12	Doug Davis	Nits – remove tabs, extra [yyy]'s ...
wd-15	2006-06-14	Doug Davis	Remove extra "OPTIONAL"s – Matt Lovett
wd-15	2006-06-14	Doug Davis	Remove blank rows/columns from state table. Fix italics in state table
wd-15	2006-06-15	Doug Davis	Typo – section D was empty
wd-15	2006-06-16	Doug Davis	Issue 125 applied
wd-15	2006-06-16	Doug Davis	Issue 126 applied
wd-15	2006-06-16	Doug Davis	Issue 127 applied
wd-15	2006-06-16	Doug Davis	Issue 133 applied
wd-15	2006-06-16	Doug Davis	Issue 136 applied
wd-15	2006-06-16	Doug Davis	Issue 138 applied
wd-15	2006-06-16	Doug Davis	Issue 135 applied
wd-15	2006-06-20	Doug Davis	Added all TC members to the ack list
wd-15	2006-06-22	Doug Davis	Issue 129 applied
wd-15	2006-06-22	Doug Davis	Issue 130 applied
wd-15	2006-06-22	Doug Davis	Issue 137 applied
wd-15	2006-06-26	Doug Davis	Issue 111 applied
wd-15	2006-06-26	Doug Davis	Missed a part of issue 129
wd-15	2006-06-30	Doug Davis	Fixed a typo in schema
wd-15	2006-06-30	Doug Davis	Issue 141 applied
wd-15	2006-06-30	Doug Davis	Issue 142 applied
wd-15	2006-06-30	Doug Davis	Issue 148 applied
wd-15	2006-06-30	Doug Davis	Issue 149 applied
wd-15	2006-06-30	Doug Davis	Issue 150 applied
wd-15	2006-07-06	Doug Davis	Issue 121 applied
wd-15	2006-07-21	Doug Davis	Issue 139 applied
wd-15	2006-07-21	Doug Davis	Issue 144 applied
wd-15	2006-07-21	Doug Davis	Issue 147 applied
wd-15	2006-07-21	Doug Davis	Issues 122-124 applied
wd-15	2006-07-27	Doug Davis	Updated list of oasis TC members (i134)
wd-15	2006-07-27	Doug Davis	Issue 140 applied
wd-15	2006-07-27	Doug Davis	Issue 145 applied
wd-15	2006-07-27	Doug Davis	Issue 143 applied
wd-15	2006-07-28	Doug Davis	Lots of minor typos found by Matt L.
wd-15	2006-07-28	Doug Davis	Issue 113 applied
wd-15	2006-08-04	Doug Davis	Update old namespaces – found by PaulC
wd-15	2006-08-04	Doug Davis	Issue 150 applied

Rev	Date	By Whom	What
wd-15	2006-08-04	Doug Davis	Minor typos – found by PeterN
wd-15	2006-08-04	Doug Davis	Verify all [refs]
wd-15	2006-08-04	Doug Davis	Change namespace to 2006/08
wd-15	2006-08-04	Doug Davis	Issue 148 applied
wd-15	2006-08-07	Doug Davis	Add some new glossary terms – per GilP
cd-04	2006-08-10	Gilbert Pilz	Formatting changes for better HTML rendering.
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.
wd-16	2006-10-25	Doug Davis	Accept all changes, update to wd16
wd-16	2006-10-26	Doug Davis	PR002 applied
wd-16	2006-10-26	Doug Davis	PR003 applied
wd-16	2006-10-26	Doug Davis	PR004 applied
wd-16	2006-10-27	Doug Davis	PR005 applied
wd-16	2006-10-27	Doug Davis	PR006 applied
wd-16	2006-10-27	Doug Davis	PR024 applied
wd-16	2006-11-13	Doug Davis	PR010 applied
wd-16	2006-11-13	Doug Davis	PR011 applied (technically as part of PR004)
wd-16	2006-11-13	Doug Davis	PR016 applied
wd-16	2006-11-13	Doug Davis	PR032 applied
wd-16	2006-11-20	Doug Davis	PR025 applied
wd-16	2006-11-20	Doug Davis	PR023 applied
wd-16	2006-12-03	Doug Davis	PR036 applied
wd-16	2006-12-03	Doug Davis	PR017 applied
wd-16	2006-12-11	Doug Davis	PR012 applied (and PR013)
wd-16	2006-12-14	Doug Davis	PR033 applied – changed a 'return' to 'generate' when talking about a fault
wd-16	2007-01-04	Doug Davis	PR018 applied
wd-16	2007-01-05	Doug Davis	Moved MakeConnection to new spec
wd-16	2007-01-17	Doug Davis	PR026 applied
wd-16	2007-01-18	Doug Davis	PR021 applied
wd-16	2007-01-18	Doug Davis	PR022 applied
wd-16	2007-01-18	Doug Davis	Fixed a few typos (Doug,Gil)
wd-16	2007-01-18	Gilbert Pilz	PR007 applied
wd-16	2007-01-25	Doug Davis	PR039 applied
wd-17	2007-01-31	Doug Davis	Lots of typos from MarcG Updated WD number and date
wd-17	2007-02-01	Doug Davis	PR038 applied
wd-17	2007-02-01	Doug Davis	PR035 (009,020 dups) applied

Rev	Date	By Whom	What
			Fixed typo in state table