



Web Services Reliable Messaging (WS-ReliableMessaging)

Committee Draft 04, August 11, 2006

Document identifier:

wsrm-1.1-spec-cd-04

Location:

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

Editors:

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

Contributors:

See the Acknowledgments (Appendix E).

Abstract:

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

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

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

Status:

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

Table of Contents

1	Introduction.....	4
1.1	Notational Conventions.....	4
1.2	Namespace.....	5
1.3	Compliance.....	5
2	Reliable Messaging Model.....	6
2.1	Glossary.....	6
2.2	Protocol Preconditions.....	7
2.3	Protocol Invariants.....	7
2.4	Example Message Exchange.....	8
3	RM Protocol Elements.....	10
3.1	Considerations on the Use of Extensibility Points.....	10
3.2	Considerations on the Use of "Piggy-Backing".....	10
3.3	Composition with WS-Addressing.....	10
3.4	Sequence Creation.....	10
3.5	Closing A Sequence.....	15
3.6	Sequence Termination.....	16
3.7	Sequences.....	18
3.8	Request Acknowledgement.....	19
3.9	Sequence Acknowledgement.....	20
3.10	MakeConnection.....	22
3.11	MessagePending.....	24
4	Faults.....	25
4.1	SequenceFault Element.....	26
4.2	Sequence Terminated.....	27
4.3	Unknown Sequence.....	27
4.4	Invalid Acknowledgement.....	28
4.5	Message Number Rollover.....	28
4.6	Create Sequence Refused.....	29
4.7	Sequence Closed.....	29
4.8	WSRM Required.....	30
4.9	Unsupported Selection.....	30
5	Security Threats and Countermeasures.....	32
5.1	Threats and Countermeasures.....	32
5.1.1	Integrity Threats.....	32
5.1.1.1	Countermeasures.....	32
5.1.2	Resource Consumption Threats.....	33
5.1.2.1	Countermeasures.....	33

80	5.1.3 Sequence Spoofing Threats.....	33
81	5.1.3.1 Sequence Hijacking.....	33
82	5.1.3.2 Countermeasures.....	33
83	5.2 Security Solutions and Technologies.....	34
84	5.2.1 Transport Layer Security.....	34
85	5.2.1.1 Model.....	34
86	5.2.1.2 Countermeasure Implementation.....	35
87	5.2.2 SOAP Message Security.....	36
88	5.2.2.1 Model.....	36
89	5.2.2.2 Countermeasure Implementation.....	36
90	6 Securing Sequences.....	38
91	6.1 Securing Sequences Using WS-Security.....	38
92	6.2 Securing Sequences Using SSL/TLS.....	39
93	7 References.....	41
94	7.1 Normative.....	41
95	7.2 Non-Normative.....	41
96	Appendix A. Schema.....	43
97	Appendix B. WSDL.....	48
98	Appendix C. Message Examples.....	50
99	Appendix C.1 Create Sequence.....	50
100	Appendix C.2 Initial Transmission.....	50
101	Appendix C.3 First Acknowledgement.....	52
102	Appendix C.4 Retransmission.....	52
103	Appendix C.5 Termination.....	53
104	Appendix C.6 MakeConnection.....	54
105	Appendix D. State Tables.....	58
106	Appendix E. Acknowledgments.....	63
107	Appendix F. Revision History.....	64
108	Appendix G. Notices.....	70

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 Notational Conventions

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/200608>

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/200608
wsa	http://www.w3.org/2005/08/addressing
wsaw	http://www.w3.org/2006/05/addressing/wsdl
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 Compliance

An implementation is not compliant 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 compliant 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 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.

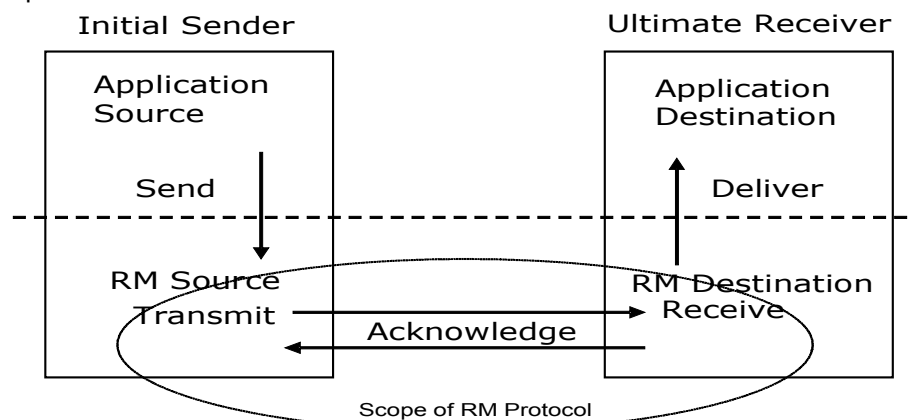


Figure 1: Reliable Messaging Model

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.

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

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

200 **Acknowledgement Request:** A message containing a `AckRequested` header. Acknowledgement
201 Requests may or may not contain a SOAP body.

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

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

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

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

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

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

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

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

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

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

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

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

219 2.2 Protocol Preconditions

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

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

229 2.3 Protocol Invariants

230 During the lifetime of a Sequence, two 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.

2.4 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.

247 5. The 2nd message in the Sequence is lost in transit.

248 6. The 3rd message is the last in this Sequence and the RM Source includes an `AckRequested`
249 header to ensure that it gets a timely `SequenceAcknowledgement` for the Sequence.

250 7. The RM Destination acknowledges receipt of message numbers 1 and 3 as a result of receiving the
251 RM Source's `AckRequested` header.

252 8. The RM Source retransmits the unacknowledged message with `MessageNumber` 2. This is a new
253 message from the perspective of the underlying transport, but it has the same `Sequence Identifier`
254 and `MessageNumber` so the RM Destination can recognize it as a duplicate of the earlier message,
255 in case the original and retransmitted messages are both Received. The RM Source includes an
256 `AckRequested` header in the retransmitted message so the RM Destination will expedite an
257 acknowledgement.

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

260 10. The RM Source Receives this Acknowledgement and sends a `TerminateSequence` message to the
261 RM Destination indicating that the Sequence is completed and reclaims any resources associated
262 with the Sequence.

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

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

278 Now that the basic model has been outlined, the details of the elements used in this protocol are now
279 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 header blocks may be added to messages that happen to be 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.

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 section 3 below, 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 3.1 below, the value of the `wsa:Action` IRI would be:

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

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

319 The following exemplar defines the CreateSequence syntax:

```
320 <wsrm:CreateSequence ...>
321   <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
322   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
323   <wsrm:Offer ...>
324     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
325     <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
326     <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
327     <wsrm:IncompleteSequenceBehavior>
328       wsrml:IncompleteSequenceBehaviorType
329     </wsrm:IncompleteSequenceBehavior> ?
330     ...
331   </wsrm:Offer> ?
332   ...
333 </wsrm:CreateSequence>
```

334 /wsrm:CreateSequence

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

339 /wsrm:CreateSequence/wsrm:AcksTo

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

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

349 /wsrm:CreateSequence/wsrm:Expires

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

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

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

357 /wsrm:CreateSequence/wsrm:Offer

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

360 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier

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

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

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

366 /wsrm:CreateSequence/wsrm:Offer/wsrm:Endpoint

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

371 Implementations MUST NOT use an endpoint reference in the Endpoint element that would prevent the
372 sending of Sequence Lifecycle Message, Sequence Traffic Message, etc. For example, using the WS-
373 Addressing "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM
374 Destination to ever send Sequence Lifecycle Messages (e.g. `TerminateSequence`) to the RM Source
375 for the Offered Sequence. Implementations MAY use the WS-RM anonymous URI template and doing so
376 implies that messages will be retrieved using a mechanism such as the `MakeConnection` message (see
377 section 3.7).

378 /wsrm:CreateSequence/wsrm:Offer/wsrm:Expires

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

382 /wsrm:CreateSequence/wsrm:Offer/wsrm:Expires/@{any}

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

385 /wsrm:CreateSequence/wsrm:Offer/wsrm:IncompleteSequenceBehavior

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

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

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

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

396 /wsrm:CreateSequence/wsrm:Offer/{any}

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

399 /wsrm:CreateSequence/wsrm:Offer/@{any}

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

402 /wsrm:CreateSequence/{any}

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

405 /wsrm:CreateSequence/@{any}

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

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

412 The following exemplar defines the `CreateSequenceResponse` syntax:

```
413 <wsrm:CreateSequenceResponse ...>
414   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
415   <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
416   <wsrm:IncompleteSequenceBehavior>
417     wsrm:IncompleteSequenceBehaviorType
418   </wsrm:IncompleteSequenceBehavior> ?
419   <wsrm:Accept ...>
420     <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
421     ...
422   </wsrm:Accept> ?
423   ...
424 </wsrm:CreateSequenceResponse>
```

425 /wsrm:CreateSequenceResponse

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

429 /wsrm:CreateSequenceResponse/wsrm:Identifier

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

433 /wsrm:CreateSequenceResponse/wsrm:Identifier/@{any}

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

436 /wsrm:CreateSequenceResponse/wsrm:Expires

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

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

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

448 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`
 449 This element, if present, specifies the behavior that the destination will exhibit upon the closure or
 450 termination of an incomplete Sequence. For the purposes of defining the values used, the term "discard"
 451 refers to behavior equivalent to the Application Destination never processing a particular message.

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

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

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

459 `/wsrm:CreateSequenceResponse/wsrm:Accept`
 460 This element, if present, enables an RM Destination to accept the offer of a corresponding Sequence for
 461 the reliable exchange of messages Transmitted from RM Destination to RM Source.

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

465 `/wsrm:CreateSequenceResponse/wsrm:Accept/wsrm:AcksTo`
 466 The RM Destination MUST include this element, of type `wsa:EndpointReferenceType` (as specified
 467 by WS-Addressing). It specifies the endpoint reference to which messages containing
 468 `SequenceAcknowledgement` header blocks and faults related to the created Sequence are to be sent,
 469 unless otherwise noted in this specification (for example, see Section 3.2).

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

474 `/wsrm:CreateSequenceResponse/wsrm:Accept/{any}`
 475 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
 476 to be passed.

477 `/wsrm:CreateSequenceResponse/wsrm:Accept/@{any}`
 478 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
 479 to be passed.

480 `/wsrm:CreateSequenceResponse/{any}`
 481 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
 482 to be passed.

483 `/wsrm:CreateSequenceResponse/@{any}`
 484 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
 485 element.

3.5 Closing A Sequence

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

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

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

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

The following exemplar defines the `CloseSequence` syntax:

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

`/wsrm:CloseSequence`

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

`/wsrm:CloseSequence/wsrm:Identifier`

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

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

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

`/wsrm:CloseSequence/{any}`

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

`/wsrm:CloseSequence@{any}`

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

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

534 The following exemplar defines the `CloseSequenceResponse` syntax:

```
535 <wsrm:CloseSequenceResponse ...>  
536   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
537   ...  
538 </wsrm:CloseSequenceResponse>
```

539 `/wsrm:CloseSequenceResponse`

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

542 `/wsrm:CloseSequenceResponse/wsrm:Identifier`

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

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

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

549 `/wsrm:CloseSequenceResponse/{any}`

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

552 `/wsrm:CloseSequenceResponse@{any}`

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

555 3.6 Sequence Termination

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

563 The following exemplar defines the `TerminateSequence` syntax:

```
564 <wsrm:TerminateSequence ...>  
565   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
566   ...  
567 </wsrm:TerminateSequence>
```

568 `/wsrm:TerminateSequence`

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

574 /wsrm:TerminateSequence/wsrm:Identifier

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

578 /wsrm:TerminateSequence/wsrm:Identifier/@{any}

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

581 /wsrm:TerminateSequence/{any}

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

584 /wsrm:TerminateSequence/@{any}

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

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

590 The following exemplar defines the `TerminateSequenceResponse` syntax:

```
591 <wsrm:TerminateSequenceResponse ...>  
592   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
593   ...  
594 </wsrm:TerminateSequenceResponse>
```

595 /wsrm:TerminateSequenceResponse

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

599 /wsrm:TerminateSequenceResponse/wsrm:Identifier

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

603 /wsrm:TerminateSequenceResponse/wsrm:Identifier/@{any}

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

606 /wsrm:TerminateSequenceResponse/{any}

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

609 /wsrm:TerminateSequenceResponse/@{any}

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

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

615 3.7 Sequences

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

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

623 A following exemplar defines its syntax:

```
624 <wsrm:Sequence ...>  
625   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
626   <wsrm:MessageNumber> wsrm:MessageNumberType </wsrm:MessageNumber>  
627   ...  
628 </wsrm:Sequence>
```

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

630 `/wsrm:Sequence`

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

637 `/wsrm:Sequence/wsrm:Identifier`

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

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

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

644 `/wsrm:Sequence/wsrm:MessageNumber`

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

649 `/wsrm:Sequence/{any}`

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

652 /wsrm:Sequence/@{any}

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

655 The following example illustrates a Sequence header block.

```
656 <wsrm:Sequence>  
657   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
658   <wsrm:MessageNumber>10</wsrm:MessageNumber>  
659 </wsrm:Sequence>
```

660 3.8 Request Acknowledgement

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

663 The RM Source MAY request an Acknowledgement Message from the RM Destination at any time by
664 including an `AckRequested` header block in any message targeted to the RM Destination. An RM
665 Destination that Receives a message that contains an `AckRequested` header block MUST send a
666 message containing a `SequenceAcknowledgement` header block to the `AcksTo` endpoint reference
667 (see Section 3.1) for a known Sequence or else generate an `UnknownSequence` fault. If a non-
668 `mustUnderstand` fault occurs when processing an RM header that was piggy-backed on another
669 message, a fault MUST be generated, but the processing of the original message MUST NOT be
670 affected. It is RECOMMENDED that the RM Destination return a `AcknowledgementRange` or `None`
671 element instead of a `Nack` element (see Section 3.6).

672 The following exemplar defines its syntax:

```
673 <wsrm:AckRequested ...>  
674   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>  
675   ...  
676 </wsrm:AckRequested>
```

677 /wsrm:AckRequested

678 This element requests an Acknowledgement for the identified Sequence.

679 /wsrm:AckRequested/wsrm:Identifier

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

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

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

686 /wsrm:AckRequested/{any}

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

689 /wsrm:AckRequested/@{any}

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

692 3.9 Sequence Acknowledgement

693 The RM Destination informs the RM Source of successful message receipt using a
694 `SequenceAcknowledgement` header block. The RM Destination MAY Transmit the
695 `SequenceAcknowledgement` header block independently or it MAY include the
696 `SequenceAcknowledgement` header block on any message targeted to the `AcksTo` EPR.
697 Acknowledgements can be explicitly requested using the `AckRequested` directive (see Section 3.5). If a
698 non-mustUnderstand fault occurs when processing an RM header that was piggy-backed on another
699 message, a fault MUST be generated, but the processing of the original message MUST NOT be
700 affected.

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

703 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
704 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
705 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
706 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
707 on the protocol binding-specific channel. Such a channel is provided by the context of a Received
708 message containing a SOAP envelope that contains a `Sequence` header block and/or a `AckRequested`
709 header block for that same Sequence identifier.

710 The following exemplar defines its syntax:

```
711 <wsrm:SequenceAcknowledgement ...>
712   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
713   [ [ [ <wsrm:AcknowledgementRange ...
714         Upper="wsrm:MessageNumberType"
715         Lower="wsrm:MessageNumberType"/> +
716         | <wsrm:None/> ]
717     <wsrm:Final/> ? ]
718   | <wsrm:Nack> wsrm:MessageNumberType </wsrm:Nack> + ]
719   ...
720 </wsrm:SequenceAcknowledgement>
```

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

723 `/wsrm:SequenceAcknowledgement`

724 This element contains the Sequence Acknowledgement information.

725 `/wsrm:SequenceAcknowledgement/wsrm:Identifier`

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

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

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

734 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange

735 The RM Destination MAY include one or more instances of this element within a

736 SequenceAcknowledgement header block. It contains a range of Sequence MessageNumbers

737 successfully accepted by the RM Destination. The ranges SHOULD NOT overlap. The RM Destination

738 MUST NOT include this element if a sibling Nack or None element is also present as a child of

739 SequenceAcknowledgement.

740 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper

741 The RM Destination MUST set the value of this attribute equal to the message number of the highest

742 contiguous message in a Sequence range accepted by the RM Destination.

743 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower

744 The RM Destination MUST set the value of this attribute equal to the message number of the lowest

745 contiguous message in a Sequence range accepted by the RM Destination.

746 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@{any}

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

748 element.

749 /wsrm:SequenceAcknowledgement/wsrm:None

750 The RM Destination MUST include this element within a SequenceAcknowledgement header block if

751 the RM Destination has not accepted any messages for the specified Sequence. The RM Destination

752 MUST NOT include this element if a sibling AcknowledgementRange or Nack element is also present

753 as a child of the SequenceAcknowledgement.

754 /wsrm:SequenceAcknowledgement/wsrm:Final

755 The RM Destination MAY include this element within a SequenceAcknowledgement header block. This

756 element indicates that the RM Destination is not receiving new messages for the specified Sequence. The

757 RM Source can be assured that the ranges of messages acknowledged by this

758 SequenceAcknowledgement header block will not change in the future. The RM Destination MUST

759 include this element when the Sequence is closed. The RM Destination MUST NOT include this element

760 when sending a Nack; it can only be used when sending AcknowledgementRange elements or a None.

761 /wsrm:SequenceAcknowledgement/wsrm:Nack

762 The RM Destination MAY include this element within a SequenceAcknowledgement header block. If

763 used, the RM Destination MUST set the value of this element to a MessageNumberType representing

764 the MessageNumber of an unreceived message in a Sequence. The RM Destination MUST NOT include

765 a Nack element if a sibling AcknowledgementRange or None element is also present as a child of

766 SequenceAcknowledgement. Upon the receipt of a Nack, an RM Source SHOULD retransmit the

767 message identified by the Nack. The RM Destination MUST NOT issue a SequenceAcknowledgement

768 containing a Nack for a message that it has previously acknowledged within a

769 AcknowledgementRange. The RM Source SHOULD ignore a SequenceAcknowledgement containing

770 a Nack for a message that has previously been acknowledged within a AcknowledgementRange.

771 /wsrm:SequenceAcknowledgement/{any}

772 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,

773 to be passed.

774 /wsrm:SequenceAcknowledgement/@{any}

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

777 The following examples illustrate `SequenceAcknowledgement` elements:

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

```
779 <wsrm:SequenceAcknowledgement>  
780   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
781   <wsrm:AcknowledgementRange Upper="10" Lower="1"/>  
782 </wsrm:SequenceAcknowledgement>
```

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

```
785 <wsrm:SequenceAcknowledgement>  
786   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
787   <wsrm:AcknowledgementRange Upper="2" Lower="1"/>  
788   <wsrm:AcknowledgementRange Upper="6" Lower="4"/>  
789   <wsrm:AcknowledgementRange Upper="10" Lower="8"/>  
790 </wsrm:SequenceAcknowledgement>
```

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

```
792 <wsrm:SequenceAcknowledgement>  
793   <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>  
794   <wsrm:Nack>3</wsrm:Nack>  
795 </wsrm:SequenceAcknowledgement>
```

796 3.10 MakeConnection

797 When an Endpoint is not directly addressable (e.g. behind a firewall or not able to allow incoming
798 connections), an anonymous URI in the EPR address property can indicate such an Endpoint. The WS-
799 Addressing anonymous URI is one such anonymous URI. This specification defines a URI template (the
800 WS-RM anonymous URI) which may be used to uniquely identify anonymous Endpoints.

```
801 http://docs.oasis-open.org/ws-rx/wsr/200608/anonymous?id={uuid}
```

802 This URI template in an EPR indicates a protocol-specific back-channel will be established through a
803 mechanism such as `MakeConnection`, defined below. When using this URI template, "{uuid}" MUST be
804 replaced by a UUID value as defined by RFC4122[UUID]. This UUID value uniquely distinguishes the
805 Endpoint. A sending Endpoint SHOULD Transmit messages at Endpoints identified with the URI template
806 using a protocol-specific back-channel, including but not limited to those established with a
807 `MakeConnection` message. Note, this URI is semantically similar to the WS-Addressing anonymous
808 URI if a protocol-specific back-channel is available.

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

814 The following exemplar defines the `MakeConnection` syntax:

```
815 <wsrm:MakeConnection ...>  
816   <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier> ?  
817   <wsrm:Address ...> xs:anyURI </wsrm:Address> ?  
818   ...  
819 </wsrm:MakeConnection>
```

820 /wsrm:MakeConnection

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

823 /wsrm:MakeConnection/wsrm:Identifier

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

829 /wsrm:MakeConnection/wsrm:Identifier/@{any}

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

832 /wsrm:MakeConnection/wsrm:Address

833 This element specifies the URI (`wsa:Address`) of the initiating Endpoint. Endpoints MUST NOT return
834 messages on the transport-specific back-channel unless they have been addressed to this URI. This
835 `Address` property and a message's WS-Addressing destination property are considered identical when
836 they are exactly the same character-for-character. Note that URIs which are not identical in this sense
837 may in fact be functionally equivalent. Examples include URI references which differ only in case, or
838 which are in external entities which have different effective base URIs. If this element is omitted from the
839 message then the `Identifier` MUST be included in the message.

840 /wsrm:MakeConnection/wsrm:Address/@{any}

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

843 /wsrm:MakeConnection/{any}

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

848 /wsrm:MakeConnection/@{any}

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

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

854 The management of messages that are awaiting the establishment of a back-channel to their receiving
855 Endpoint is an implementation detail that is outside the scope of this specification. Note, however, that
856 these messages form a class of asynchronous messages that is not dissimilar from "ordinary"
857 asynchronous messages that are waiting for the establishment of a connection to their destination
858 Endpoints.

859 This specification places no constraint on the types of messages that can be returned on the transport-
860 specific back-channel. As in an asynchronous environment, it is up to the recipient of the
861 `MakeConnection` message to decide which messages are appropriate for transmission to any particular

862 Endpoint. However, the Endpoint processing the `MakeConnection` message MUST insure that the
863 messages match the selection criteria as specified by the child elements of the `MakeConnection`
864 element.

865 Since the message exchange pattern use by `MakeConnection` may be new to some, the following points
866 need to be reiterated for clarification:

- 867 ● The `MakeConnection` message is a one-way operation: no reply message is generated from this
868 message.
- 869 ● The presence, or value, of any `wsa:ReplyTo` Header in the `MakeConnection` message has no
870 impact on the messages exchanged since there is no reply to `MakeConnection`. As such, the
871 WS-Addressing processing rules for reply processing do not apply.
- 872 ● In the absence of any pending message, there will be no message transmitted on the transport
873 back-channel. E.g. In the HTTP case, just an HTTP 202 Accepted will be returned without any
874 SOAP envelope in the HTTP response message.
- 875 ● When there is a message pending, the endpoint receiving the `MakeConnection` places the
876 message onto the transport back-channel. By doing this it is reusing an otherwise empty stream.
- 877 ● While the `MakeConnection` allows for a message to be sent on an otherwise empty back-
878 channel, it does not request a response – again, it is a one-way operation. However, to avoid
879 confusion, the `MakeConnection` operation is excluded from the WSDL.

880 **MessagePending**

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

885 The following exemplar defines the `MessagePending` syntax:

```
886 <wsrm:MessagePending pending="xs:boolean" ...>  
887   ...  
888 </wsrm:MessagePending>
```

889 `/wsrm:MessagePending`

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

891 `/wsrm:MessagePending@pending`

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

894 `/wsrm:MessagePending/{any}`

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

897 `/wsrm:MessagePending/@{any}`

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

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

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. WSRM Required 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. RM Destinations that generate Sequence faults SHOULD send those faults 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/200608/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/200608/fault
    </wsa:Action>
    <!-- Headers elided for clarity. -->
  </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]
```

```

945     ...
946     </S:Detail>
947     </S:Fault>
948     </S:Body>
949     </S:Envelope>

```

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

```

952 <S11:Envelope>
953   <S11:Header>
954     <wsrm:SequenceFault>
955       <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
956       <wsrm:Detail> [Detail] </wsrm:Detail>
957       ...
958     </wsrm:SequenceFault>
959     <!-- Headers elided for clarity. -->
960   </S11:Header>
961   <S11:Body>
962     <S11:Fault>
963       <faultcode> [Code] </faultcode>
964       <faultstring> [Reason] </faultstring>
965     </S11:Fault>
966   </S11:Body>
967 </S11:Envelope>

```

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

```

970 <S11:Envelope>
971   <S11:Body>
972     <S11:Fault>
973       <faultcode> [Subcode] </faultcode>
974       <faultstring> [Reason] </faultstring>
975     </S11:Fault>
976   </S11:Body>
977 </S11:Envelope>

```

978 4.1 SequenceFault Element

979 The purpose of the `SequenceFault` element is to carry the specific details of a fault generated during
 980 the reliable messaging specific processing of a message belonging to a Sequence. WS-
 981 ReliableMessaging nodes MUST use the `SequenceFault` container only in conjunction with the SOAP
 982 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the `SequenceFault` container in
 983 conjunction with the SOAP 1.2 binding.

984 The following exemplar defines its syntax:

```

985 <wsrm:SequenceFault ...>
986   <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
987   <wsrm:Detail> ... </wsrm:Detail> ?
988   ...
989 </wsrm:SequenceFault>

```

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

991 /wsrm:SequenceFault

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

993 /wsrm:SequenceFault/wsrm:FaultCode

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

996 `/wsrm:SequenceFault/wsrm:Detail`

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

998 `/wsrm:SequenceFault/wsrm:Detail/{any}`

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

1000 `/wsrm:SequenceFault/wsrm:Detail/@{any}`

1001 The application specific error information related to the fault being described.

1002 `/wsrm:SequenceFault/{any}`

1003 This is an extensibility mechanism to allow different (extensible) types of information, based on a schema,
1004 to be passed.

1005 `/wsrm:SequenceFault/@{any}`

1006 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
1007 element.

1008 4.2 Sequence Terminated

1009 The Endpoint that generates this fault SHOULD make every reasonable effort to notify the corresponding
1010 Endpoint of this decision.

1011 Properties:

1012 [Code] Sender or Receiver

1013 [Subcode] `wsrm:SequenceTerminated`

1014 [Reason] The Sequence has been terminated due to an unrecoverable error.

1015 [Detail]

1016 `<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.

1017 4.3 Unknown Sequence

1018 Properties:

1019 [Code] Sender

1020 [Subcode] `wsrm:UnknownSequence`

1021 [Reason] The value of wsrn:Identifier is not a known Sequence identifier.

1022 [Detail]

1023 `<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.

1024 4.4 Invalid Acknowledgement

1025 An example of when this fault is generated is when a message is Received by the RM Source containing
1026 a SequenceAcknowledgement covering messages that have not been sent.

1027 [Code] Sender

1028 [Subcode] wsrn:InvalidAcknowledgement

1029 [Reason] The SequenceAcknowledgement violates the cumulative Acknowledgement invariant.

1030 [Detail]

1031 `<wsrm:SequenceAcknowledgement ...> ... </wsrm:SequenceAcknowledgement>`

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

1032 4.5 Message Number Rollover

1033 If the condition listed below is reached, the RM Destination MUST generate this fault.

1034 Properties:

1035 [Code] Sender

1036 [Subcode] wsrn:MessageNumberRollover

1037 [Reason] The maximum value for wsrn:MessageNumber has been exceeded.

1038 [Detail]

```
1039 <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
1040 <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.

1041 4.6 Create Sequence Refused

1042 Properties:

1043 [Code] Sender

1044 [Subcode] wsrm:CreateSequenceRefused

1045 [Reason] The create Sequence request has been refused by the RM Destination.

1046 [Detail]

```
1047 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.

1048 4.7 Sequence Closed

1049 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.

1050 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
1051 is closed or when an RM Destination is asked to close a Sequence that is already closed.

1052 Properties:

1053 [Code] Sender

1054 [Subcode] wsrm:SequenceClosed

1055 [Reason] The Sequence is closed and can not accept new messages.

1056 [Detail]

1057 `<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.

1058 4.8 WSRM Required

1059 If an RM Destination requires the use of WS-RM, this fault is generated when it Receives an incoming
1060 message that did not use this protocol.

1061 Properties:

1062 [Code] Sender

1063 [Subcode] wsrm:WSRMRequired

1064 [Reason] The RM Destination requires the use of WSRM.

1065 [Detail]

1066 `xs:any`

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

1067 4.9 Unsupported Selection

1068 The QName of the unsupported element(s) are included in the detail.

1069 Properties:

1070 [Code] Receiver

1071 [Subcode] wsrm:UnsupportedSelection

1072 [Reason] The extension element used in the message selection is not supported by the RM Source

1073 [Detail]

1074 `<wsrm:UnsupportedElement> xs:QName </wsrm:UnsupportedElement>+`

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

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

1155 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
1156 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
1157 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
1158 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
1159 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
1160 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

1161 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
1162 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
1163 sequence peer it MUST be able to identify and authenticate the entity that sent the
1164 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
1165 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
1166 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
1167 creation time.

1168 5.2 Security Solutions and Technologies

1169 The security threats described in the previous sections are neither new nor unique. The solutions that
1170 have been developed to secure other SOAP-based protocols can be used to secure WS-RM as well. This
1171 section maps the facilities provided by common web services security solutions against countermeasures
1172 described in the previous sections.

1173 Before continuing this discussion, however, some examination of the underlying requirements of the
1174 previously described countermeasures is necessary. Specifically it should be noted that the technique
1175 described in Section 5.1.2.1 has two components. Firstly, the RM Destination identifies and authenticates
1176 the issuer of a `CreateSequence` message. Secondly, the RM Destination to performs an authorization
1177 check against this authenticated identity and determines if the RM Source is permitted to create
1178 Sequences with the RM Destination. Since the facilities for performing this authorization check (runtime
1179 infrastructure, policy frameworks, etc.) lie completely within the domain of individual implementations, any
1180 discussion of such facilities is considered to be beyond the scope of this specification.

1181 5.2.1 Transport Layer Security

1182 This section describes how the the facilities provided by SSL/TLS [RFC 4346] can be used to implement
1183 the countermeasures described in the previous sections. The use of SSL/TLS is subject to the constraints
1184 defined in Section 4 of the Basic Security Profile 1.0 [BSP 1.0].

1185 The description provided here is general in nature and is not intended to serve as a complete definition on
1186 the use of SSL/TLS to protect WS-RM. In order to interoperate implementations need to agree on the
1187 choice of features as well as the manner in which they will be used. The mechanisms described in the
1188 Web Services Security Policy Language [SecurityPolicy] MAY be used by services to describe the
1189 requirements and constraints of the use of SSL/TLS.

1190 5.2.1.1 Model

1191 The basic model for using SSL/TLS is as follows:

- 1192 1. The RM Source establishes an SSL/TLS session with the RM Destination.
- 1193 2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
1194 Destination.

- 1195 3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
1196 asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
1197 synchronous `CreateSequenceResponse` using the session established in (1).
- 1198 4. For the lifetime of the Sequence the RM Source uses the SSL/TLS session from (1) to Transmit
1199 any and all messages or faults that refer to that Sequence.
- 1200 5. For the lifetime of the Sequence the RM Destination either uses the SSL/TLS session established
1201 in (3) to Transmit any and all messages or faults that refer to that Sequence or, for synchronous
1202 exchanges, the RM Destination uses the SSL/TLS session established in (1).

1203 5.2.1.2 Countermeasure Implementation

1204 Used in its simplest fashion (without relying upon any authentication mechanisms), SSL/TLS provides the
1205 necessary integrity qualities to counter the threats described in Section 5.1.1. Note, however, that the
1206 nature of SSL/TLS limits the scope of this integrity protection to a single transport level session. If
1207 SSL/TLS is the only mechanism used to provide integrity, any intermediaries between the RM Source and
1208 the RM Destination MUST be trusted to preserve the integrity of the messages that flow through them.

1209 As noted, the technique described in Sections 5.1.2.1 involves the use of authentication. This specification
1210 advocates either of two mechanisms for authenticating entities using SSL/TLS. In both of these methods
1211 the SSL/TLS server (the party accepting the SSL/TLS connection) authenticates itself to the SSL/TLS
1212 client using an X.509 certificate that is exchanged during the SSL/TLS handshake.

- 1213 • **HTTP Basic Authentication:** This method of authentication presupposes that a SOAP/HTTP
1214 binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
1215 establishment of the the SSL/TLS session, the sending party authenticates itself to the receiving
1216 party using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might
1217 authenticate itself to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using
1218 BasicAuth. Similarly the RM Destination might authenticate itself to the RM Source (e.g. when
1219 sending an Acknowledgement) using BasicAuth.
- 1220 • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
1221 connection authenticates itself to the party accepting the connection using an X.509 certificate
1222 that is exchanged during the SSL/TLS handshake.

1223 To implement the countermeasures described in section 5.1.2.1 the RM Source must authenticate itself
1224 using one the above mechanisms. The authenticated identity can then be used to determine if the RM
1225 Source is authorized to create a Sequence with the RM Destination.

1226 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1227 an RM node's Sequence peer to be equivalent to their SSL/TLS session peer. This allows the
1228 authorization decisions described in section 5.1.3.2 to be based on SSL/TLS session identity rather than
1229 on authentication information. For example, an RM Destination can determine that a Sequence Traffic
1230 Message rightfully belongs to its referenced Sequence if that message arrived over the same SSL/TLS
1231 session that was used to carry the `CreateSequence` message for that Sequence. Note that requiring a
1232 one-to-one relationship between SSL/TLS session peer and Sequence peer constrains the lifetime of a
1233 SSL/TLS-protected Sequence to be less than or equal to the lifetime of the SSL/TLS session that is used
1234 to protect that Sequence.

1235 This specification does not preclude the use of other methods of using SSL/TLS to implement the
1236 countermeasures (such as associating specific authentication information with a Sequence) although such
1237 methods are not covered by this document.

1238 Issues specific to the life-cycle management of SSL/TLS sessions (such as the resumption of a SSL/TLS
1239 session) are outside the scope of this specification.

1240 **5.2.2 SOAP Message Security**

1241 The mechanisms described in WS-Security may be used in various ways to implement the
1242 countermeasures described in the previous sections. This specification advocates using the protocol
1243 described by WS-SecureConversation [[SecureConversation](#)] (optionally in conjunction with WS-Trust
1244 [[Trust](#)]) as a mechanism for protecting Sequences. The use of WS-Security (as an underlying component
1245 of WS-SecureConversation) is subject to the constraints defined in the Basic Security Profile 1.0.

1246 The description provided here is general in nature and is not intended to serve as a complete definition on
1247 the use of WS-SecureConversation/WS-Trust to protect WS-RM. In order to interoperate implementations
1248 need to agree on the choice of features as well as the manner in which they will be used. The
1249 mechanisms described in the Web Services Security Policy Language MAY be used by services to
1250 describe the requirements and constraints of the use of WS-SecureConversation.

1251 **5.2.2.1 Model**

1252 The basic model for using WS-SecureConversation is as follows:

- 1253 1. The RM Source and the RM Destination create a WS-SecureConversation security context. This
1254 may involve the participation of third parties such as a security token service. The tokens
1255 exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).
- 1256 2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security
1257 context that will be used to protect the Sequence. This is done so that, in cases where the
1258 `CreateSequence` message is signed by more than one security context, the RM Source can
1259 indicate which security context should be used to protect the newly created Sequence.
- 1260 3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s)
1261 associated with the security context to sign (as defined by WS-Security) at least the body and any
1262 relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

1263 **5.2.2.2 Countermeasure Implementation**

1264 Without relying upon any authentication information, the per-message signatures provide the necessary
1265 integrity qualities to counter the threats described in Section 5.1.1.

1266 To implement the countermeasures described in section 5.1.2.1 some mutually agreed upon form of
1267 authentication claims must be provided by the RM Source to the RM Destination during the establishment
1268 of the Security Context. These claims can then be used to determine if the RM Source is authorized to
1269 create a Sequence with the RM Destination.

1270 This specification advocates implementing the countermeasures described in section 5.1.3.2 by requiring
1271 an RM node's Sequence peer to be equivalent to their security context session peer. This allows the
1272 authorization decisions described in section 5.1.3.2 to be based on the identity of the message's security
1273 context rather than on any authentication claims that may have been established during security context
1274 initiation. Note that other methods of using WS-SecurityConversation to implement the countermeasures
1275 (such as associating specific authentication claims to a Sequence) are possible but not covered by this
1276 document.

1277 As with transport security, the requisite equivalence of a security context peer and with a Sequence peer
1278 limits the lifetime of a Sequence to the lifetime of the protecting security context. Unlike transport security,

1279 the association between a Sequence and its protecting security context cannot always be established
1280 implicitly at Sequence creation time. This is due to the fact that the `CreateSequence` and
1281 `CreateSequenceResponse` messages may be signed by more than one security context.

1282 Issues specific to the life-cycle management of WS-SecurityConversation security contexts (such as
1283 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>
```

`/wsrm:CreateSequence/wsse:SecurityTokenReference`

This element uses the extensibility mechanism defined for the `CreateSequence` element (defined in section 3.1) 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 with the requirements listed above. In order to achieve this, the RM Source SHOULD include the `UsesSequenceSTR` element as a SOAP header block within the `CreateSequence` message. This element MUST include a `soap:mustUnderstand` attribute with a value of 'true'. Thus the RM Source

1332 can be assured that a RM Destination that responds with a `CreateSequenceResponse` understands
1333 and conforms with the requirements listed above. Note that an RM Destination understanding this header
1334 does not mean that it has processed and understood any WS-Security headers, the fault behavior defined
1335 in WS-Security still applies.

1336 The following exemplar defines the `UsesSequenceSTR` syntax:

```
1337 <wsrm:UsesSequenceSTR ... />
```

1338 /wsrm:UsesSequenceSTR

1339 This element SHOULD be included as a SOAP header block in `CreateSequence` messages that use the
1340 extensibility mechanism described above in this section. The `soap:mustUnderstand` attribute value
1341 MUST be 'true'. The receiving RM Destination MUST understand and correctly implement the extension
1342 described above or else generate a `soap:MustUnderstand` fault, thus aborting the requested
1343 Sequence creation.

1344 The following is an example of a `CreateSequence` message using the

1345 `wsse:SecurityTokenReference` extension and the `UsesSequenceSTR` header block:

```
1346 <soap:Envelope ...>
1347   <soap:Header>
1348     ...
1349     <wsrm:UsesSequenceSTR soap:mustUnderstand='true' />
1350     ...
1351   </soap:Header>
1352   <soap:Body>
1353     <wsrm:CreateSequence>
1354       <wsrm:AcksTo>
1355         <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1356       </wsrm:AcksTo>
1357       <wsse:SecurityTokenReference>
1358         ...
1359       </wsse:SecurityTokenReference>
1360     </wsrm:CreateSequence>
1361   </soap:Body>
1362 </soap:Envelope>
```

1363 6.2 Securing Sequences Using SSL/TLS

1364 One mechanism for protecting a Sequence is to bind the Sequence to the underlying SSL/TLS session(s).
1365 The RM Source indicates to the RM Destination that a Sequence is to be bound to the underlying
1366 SSL/TLS session(s) via the `UsesSequenceSSL` header block. If the RM Source wishes to bind a
1367 Sequence to the underlying SSL/TLS sessions(s) it MUST include the `UsesSequenceSSL` element as a
1368 SOAP header block within the `CreateSequence` message.

1369 The following exemplar defines the `UsesSequenceSSL` syntax:

```
1370 <wsrm:UsesSequenceSSL soap:mustUnderstand="true" ... />
```

1371 /wsrm:UsesSequenceSSL

1372 The RM Source MAY include this element as a SOAP header block of a `CreateSequence` message to
1373 indicate to the RM Destination that the resulting Sequence is to be bound to the SSL/TLS session that was
1374 used to carry the `CreateSequence` message. If included, the RM Source MUST mark this header with a
1375 `soap:mustUnderstand` attribute with a value of 'true'. The receiving RM Destination MUST understand
1376 and correctly implement the functionality described in Section 5.2.1 or else generate a
1377 `soap:MustUnderstand` fault, thus aborting the requested Sequence creation.

1378 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
1379 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
1380 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
1381 `CreateSequenceResponse` message.

7 References

7.1 Normative

[KEYWORDS]

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

[SOAP 1.1]

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

[SOAP 1.2]

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

[URI]

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

[UUID]

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

[XML]

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

[XML-ns]

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

[XML-Schema Part1]

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

[XML-Schema Part2]

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

[XPath 1.0]

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

[WSDL 1.1]

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

[WS-Addressing]

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

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

7.2 Non-Normative

[BSP 1.0]

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

[RDDL 2.0]

1416 Johnathan Borden, Tim Bray, eds. "[Resource Directory Description Language \(RDDL\) 2.0](#)," January 2004

1417 **[RFC 2617]**

1418 J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Loutonen, L. Stewart, "[HTTP](#)

1419 [Authentication: Basic and Digest Access Authentication](#)," June 1999.

1420 **[RFC 4346]**

1421 T. Dierks, E. Rescorla, "[The Transport Layer Security \(TLS\) Protocol Version 1.1](#)," April 2006.

1422 **[WS-Policy]**

1423 W3C Member Submission, "[Web Services Policy Framework \(WS-Policy\)](#)," April 2006.

1424 **[WS-PolicyAttachment]**

1425 W3C Member Submission, "[Web Services Policy Attachment \(WS-PolicyAttachment\)](#)," April 2006.

1426 **[WS-Security]**

1427 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security:](#)

1428 [SOAP Message Security 1.0 \(WS-Security 2004\)](#)", OASIS Standard 200401, March 2004.

1429 Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, Ronald Monzillo, eds. "[OASIS Web Services Security:](#)

1430 [SOAP Message Security 1.1 \(WS-Security 2004\)](#)", OASIS Standard 200602, February 2006.

1431 **[RTTM]**

1432 V. Jacobson, R. Braden, D. Borman, "[TCP Extensions for High Performance](#)", RFC 1323, May

1433 1992.

1434 **[SecurityPolicy]**

1435 G. Della-Libra, et. al. "[Web Services Security Policy Language \(WS-SecurityPolicy\)](#)", July 2005

1436 **[SecureConversation]**

1437 S. Anderson, et al, "[Web Services Secure Conversation Language \(WS-SecureConversation\)](#)," February

1438 2005.

1439 **[Trust]**

1440 S. Anderson, et al, "[Web Services Trust Language \(WS-Trust\)](#)," February 2005.

Appendix A. Schema

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

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

The following copy is provided for reference.

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

```

1497     <xs:anyAttribute namespace="##other" processContents="lax"/>
1498 </xs:complexType>
1499 <xs:element name="Sequence" type="wsrm:SequenceType"/>
1500 <xs:element name="SequenceAcknowledgement">
1501   <xs:complexType>
1502     <xs:sequence>
1503       <xs:element ref="wsrm:Identifier"/>
1504       <xs:choice>
1505         <xs:sequence>
1506           <xs:choice>
1507             <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
1508               <xs:complexType>
1509                 <xs:sequence/>
1510                 <xs:attribute name="Upper" type="xs:unsignedLong"
1511 use="required"/>
1512                 <xs:attribute name="Lower" type="xs:unsignedLong"
1513 use="required"/>
1514               <xs:anyAttribute namespace="##other" processContents="lax"/>
1515             </xs:complexType>
1516           </xs:element>
1517           <xs:element name="None">
1518             <xs:complexType>
1519               <xs:sequence/>
1520             </xs:complexType>
1521           </xs:element>
1522         </xs:choice>
1523         <xs:element name="Final" minOccurs="0">
1524           <xs:complexType>
1525             <xs:sequence/>
1526           </xs:complexType>
1527         </xs:element>
1528       </xs:sequence>
1529       <xs:element name="Nack" type="xs:unsignedLong"
1530 maxOccurs="unbounded"/>
1531     </xs:choice>
1532     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1533 maxOccurs="unbounded"/>
1534   </xs:sequence>
1535   <xs:anyAttribute namespace="##other" processContents="lax"/>
1536 </xs:complexType>
1537 </xs:element>
1538 <xs:complexType name="AckRequestedType">
1539   <xs:sequence>
1540     <xs:element ref="wsrm:Identifier"/>
1541     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1542 maxOccurs="unbounded"/>
1543   </xs:sequence>
1544   <xs:anyAttribute namespace="##other" processContents="lax"/>
1545 </xs:complexType>
1546 <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
1547 <xs:complexType name="MessagePendingType">
1548   <xs:sequence>
1549     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1550 maxOccurs="unbounded"/>
1551   </xs:sequence>
1552   <xs:attribute name="pending" type="xs:boolean"/>
1553   <xs:anyAttribute namespace="##other" processContents="lax"/>
1554 </xs:complexType>
1555 <xs:element name="MessagePending" type="wsrm:MessagePendingType"/>
1556 <xs:element name="Identifier">
1557   <xs:complexType>
1558     <xs:annotation>
1559       <xs:documentation>

```

```

1560         This type is for elements whose [children] is an anyURI and can have
1561 arbitrary attributes.
1562     </xs:documentation>
1563 </xs:annotation>
1564 <xs:simpleContent>
1565     <xs:extension base="xs:anyURI">
1566         <xs:anyAttribute namespace="##other" processContents="lax"/>
1567     </xs:extension>
1568 </xs:simpleContent>
1569 </xs:complexType>
1570 </xs:element>
1571 <xs:element name="Address">
1572     <xs:complexType>
1573         <xs:simpleContent>
1574             <xs:extension base="xs:anyURI">
1575                 <xs:anyAttribute namespace="##other" processContents="lax"/>
1576             </xs:extension>
1577         </xs:simpleContent>
1578     </xs:complexType>
1579 </xs:element>
1580 <xs:complexType name="MakeConnectionType">
1581     <xs:sequence>
1582         <xs:element ref="wsrm:Identifier" minOccurs="0" maxOccurs="1"/>
1583         <xs:element ref="wsrm:Address" minOccurs="0" maxOccurs="1"/>
1584         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1585 maxOccurs="unbounded"/>
1586     </xs:sequence>
1587     <xs:anyAttribute namespace="##other" processContents="lax"/>
1588 </xs:complexType>
1589 <xs:element name="MakeConnection" type="wsrm:MakeConnectionType"/>
1590 <xs:simpleType name="MessageNumberType">
1591     <xs:restriction base="xs:unsignedLong">
1592         <xs:minInclusive value="1"/>
1593         <xs:maxInclusive value="9223372036854775807"/>
1594     </xs:restriction>
1595 </xs:simpleType>
1596 <!-- Fault Container and Codes -->
1597 <xs:simpleType name="FaultCodes">
1598     <xs:restriction base="xs:QName">
1599         <xs:enumeration value="wsrm:SequenceTerminated"/>
1600         <xs:enumeration value="wsrm:UnknownSequence"/>
1601         <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
1602         <xs:enumeration value="wsrm:MessageNumberRollover"/>
1603         <xs:enumeration value="wsrm:CreateSequenceRefused"/>
1604         <xs:enumeration value="wsrm:SequenceClosed"/>
1605         <xs:enumeration value="wsrm:WSRMRequired"/>
1606         <xs:enumeration value="wsrm:UnsupportedSelection"/>
1607     </xs:restriction>
1608 </xs:simpleType>
1609 <xs:complexType name="SequenceFaultType">
1610     <xs:sequence>
1611         <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
1612         <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
1613         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1614 maxOccurs="unbounded"/>
1615     </xs:sequence>
1616     <xs:anyAttribute namespace="##other" processContents="lax"/>
1617 </xs:complexType>
1618 <xs:complexType name="DetailType">
1619     <xs:sequence>
1620         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1621 maxOccurs="unbounded"/>
1622     </xs:sequence>

```

```

1623     <xs:anyAttribute namespace="##other" processContents="lax"/>
1624 </xs:complexType>
1625 <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
1626 <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
1627 <xs:element name="CreateSequenceResponse"
1628 type="wsrm:CreateSequenceResponseType"/>
1629 <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
1630 <xs:element name="CloseSequenceResponse"
1631 type="wsrm:CloseSequenceResponseType"/>
1632 <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
1633 <xs:element name="TerminateSequenceResponse"
1634 type="wsrm:TerminateSequenceResponseType"/>
1635 <xs:complexType name="CreateSequenceType">
1636 <xs:sequence>
1637 <xs:element ref="wsrm:AcksTo"/>
1638 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1639 <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
1640 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1641 maxOccurs="unbounded">
1642 <xs:annotation>
1643 <xs:documentation>
1644 It is the authors intent that this extensibility be used to
1645 transfer a Security Token Reference as defined in WS-Security.
1646 </xs:documentation>
1647 </xs:annotation>
1648 </xs:any>
1649 </xs:sequence>
1650 <xs:anyAttribute namespace="##other" processContents="lax"/>
1651 </xs:complexType>
1652 <xs:complexType name="CreateSequenceResponseType">
1653 <xs:sequence>
1654 <xs:element ref="wsrm:Identifier"/>
1655 <xs:element ref="wsrm:Expires" minOccurs="0"/>
1656 <xs:element name="IncompleteSequenceBehavior"
1657 type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
1658 <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
1659 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1660 maxOccurs="unbounded"/>
1661 </xs:sequence>
1662 <xs:anyAttribute namespace="##other" processContents="lax"/>
1663 </xs:complexType>
1664 <xs:complexType name="CloseSequenceType">
1665 <xs:sequence>
1666 <xs:element ref="wsrm:Identifier"/>
1667 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1668 maxOccurs="unbounded"/>
1669 </xs:sequence>
1670 <xs:anyAttribute namespace="##other" processContents="lax"/>
1671 </xs:complexType>
1672 <xs:complexType name="CloseSequenceResponseType">
1673 <xs:sequence>
1674 <xs:element ref="wsrm:Identifier"/>
1675 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1676 maxOccurs="unbounded"/>
1677 </xs:sequence>
1678 <xs:anyAttribute namespace="##other" processContents="lax"/>
1679 </xs:complexType>
1680 <xs:complexType name="TerminateSequenceType">
1681 <xs:sequence>
1682 <xs:element ref="wsrm:Identifier"/>
1683 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1684 maxOccurs="unbounded"/>
1685 </xs:sequence>

```

```

1686     <xs:anyAttribute namespace="##other" processContents="lax"/>
1687 </xs:complexType>
1688 <xs:complexType name="TerminateSequenceResponseType">
1689   <xs:sequence>
1690     <xs:element ref="wsrm:Identifier"/>
1691     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1692 maxOccurs="unbounded"/>
1693   </xs:sequence>
1694   <xs:anyAttribute namespace="##other" processContents="lax"/>
1695 </xs:complexType>
1696 <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
1697 <xs:complexType name="OfferType">
1698   <xs:sequence>
1699     <xs:element ref="wsrm:Identifier"/>
1700     <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
1701     <xs:element ref="wsrm:Expires" minOccurs="0"/>
1702     <xs:element name="IncompleteSequenceBehavior"
1703 type="wsrm:IncompleteSequenceBehaviorType" 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="AcceptType">
1710   <xs:sequence>
1711     <xs:element ref="wsrm:AcksTo"/>
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="Expires">
1718   <xs:complexType>
1719     <xs:simpleContent>
1720       <xs:extension base="xs:duration">
1721         <xs:anyAttribute namespace="##other" processContents="lax"/>
1722       </xs:extension>
1723     </xs:simpleContent>
1724   </xs:complexType>
1725 </xs:element>
1726 <xs:simpleType name="IncompleteSequenceBehaviorType">
1727   <xs:restriction base="xs:string">
1728     <xs:enumeration value="DiscardEntireSequence"/>
1729     <xs:enumeration value="DiscardFollowingFirstGap"/>
1730     <xs:enumeration value="NoDiscard"/>
1731   </xs:restriction>
1732 </xs:simpleType>
1733 <xs:element name="UsesSequenceSTR">
1734   <xs:sequence/>
1735   <xs:anyAttribute namespace="##other" processContents="lax"/>
1736 </xs:element>
1737 <xs:element name="UsesSequenceSSL">
1738   <xs:sequence/>
1739   <xs:anyAttribute namespace="##other" processContents="lax"/>
1740 </xs:element>
1741 <xs:element name="UnsupportedElement">
1742   <xs:simpleType>
1743     <xs:restriction base="xs:QName"/>
1744   </xs:simpleType>
1745 </xs:element>
1746 </xs:schema>

```

Appendix B. WSDL

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

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

The following non-normative copy is provided for reference.

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

  <wsdl:types>
    <xs:schema>
      <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrn/200608"
schemaLocation="http://docs.oasis-open.org/ws-rx/wsrn/200608/wsrn-1.1-schema-
200608.xsd"/>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="CreateSequence">
    <wsdl:part name="create" element="rm:CreateSequence"/>
  </wsdl:message>
</wsdl:definitions>
```

```

1802     </wsdl:message>
1803     <wsdl:message name="CreateSequenceResponse">
1804         <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
1805     </wsdl:message>
1806     <wsdl:message name="CloseSequence">
1807         <wsdl:part name="close" element="rm:CloseSequence"/>
1808     </wsdl:message>
1809     <wsdl:message name="CloseSequenceResponse">
1810         <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
1811     </wsdl:message>
1812     <wsdl:message name="TerminateSequence">
1813         <wsdl:part name="terminate" element="rm:TerminateSequence"/>
1814     </wsdl:message>
1815     <wsdl:message name="TerminateSequenceResponse">
1816         <wsdl:part name="terminateResponse"
1817 element="rm:TerminateSequenceResponse"/>
1818     </wsdl:message>
1819     <wsdl:message name="MakeConnection">
1820     <wsdl:part name="makeConnection" element="rm:MakeConnection"/>
1821     </wsdl:message>
1822
1823     <wsdl:portType name="SequenceAbstractPortType">
1824         <wsdl:operation name="CreateSequence">
1825             <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
1826 open.org/ws-rx/wsrn/200608/CreateSequence"/>
1827             <wsdl:output message="tns:CreateSequenceResponse"
1828 wsaw:Action="http://docs.oasis-open.org/ws-
1829 rx/wsrn/200608/CreateSequenceResponse"/>
1830         </wsdl:operation>
1831         <wsdl:operation name="CloseSequence">
1832             <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
1833 open.org/ws-rx/wsrn/200608/CloseSequence"/>
1834             <wsdl:output message="tns:CloseSequenceResponse"
1835 wsaw:Action="http://docs.oasis-open.org/ws-
1836 rx/wsrn/200608/CloseSequenceResponse"/>
1837         </wsdl:operation>
1838         <wsdl:operation name="TerminateSequence">
1839             <wsdl:input message="tns:TerminateSequence"
1840 wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrn/200608/TerminateSequence"/>
1841             <wsdl:output message="tns:TerminateSequenceResponse"
1842 wsaw:Action="http://docs.oasis-open.org/ws-
1843 rx/wsrn/200608/TerminateSequenceResponse"/>
1844         </wsdl:operation>
1845         <wsdl:operation name="MakeConnection">
1846         <wsdl:input message="tns:MakeConnection" wsaw:Action="http://docs.oasis-
1847         open.org/ws-rx/wsrn/200608/MakeConnection"/>
1848         </wsdl:operation>
1849         <!-- The MakeConnection operation would normally go here but since it
1850         doesn't follow the traditional in/out MEP it is excluded. See
1851         section 3.10 for more information. -->
1852     </wsdl:portType>
1853
1854 </wsdl:definitions>

```

Appendix C. Message Examples

Appendix C.1 Create Sequence

Create Sequence

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

Create Sequence Response

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

Appendix C.2 Initial Transmission

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

1903 Message 1

```
1904 <?xml version="1.0" encoding="UTF-8"?>
1905 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1906 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1907 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1908   <S:Header>
1909     <wsa:MessageID>
1910       http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfcbc9e
1911     </wsa:MessageID>
1912     <wsa:To>http://example.com/serviceB/123</wsa:To>
1913     <wsa:From>
1914       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1915     </wsa:From>
1916     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1917     <wsmr:Sequence>
1918       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1919       <wsmr:MessageNumber>1</wsmr:MessageNumber>
1920     </wsmr:Sequence>
1921   </S:Header>
1922   <S:Body>
1923     <!-- Some Application Data -->
1924   </S:Body>
1925 </S:Envelope>
```

1926 Message 2

```
1927 <?xml version="1.0" encoding="UTF-8"?>
1928 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1929 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1930 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1931   <S:Header>
1932     <wsa:MessageID>
1933       http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
1934     </wsa:MessageID>
1935     <wsa:To>http://example.com/serviceB/123</wsa:To>
1936     <wsa:From>
1937       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
1938     </wsa:From>
1939     <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1940     <wsmr:Sequence>
1941       <wsmr:Identifier>http://Business456.com/RM/ABC</wsmr:Identifier>
1942       <wsmr:MessageNumber>2</wsmr:MessageNumber>
1943     </wsmr:Sequence>
1944   </S:Header>
1945   <S:Body>
1946     <!-- Some Application Data -->
1947   </S:Body>
1948 </S:Envelope>
```

1949 Message 3

```
1950 <?xml version="1.0" encoding="UTF-8"?>
1951 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1952 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
1953 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1954   <S:Header>
1955     <wsa:MessageID>
1956       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546819
1957     </wsa:MessageID>
1958     <wsa:To>http://example.com/serviceB/123</wsa:To>
1959     <wsa:From>
1960       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
```

```

1961 </wsa:From>
1962 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
1963 <wsrm:Sequence>
1964 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1965 <wsrm:MessageNumber>3</wsrm:MessageNumber>
1966 </wsrm:Sequence>
1967 <wsrm:AckRequested>
1968 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1969 </wsrm:AckRequested>
1970 </S:Header>
1971 <S:Body>
1972 <!-- Some Application Data -->
1973 </S:Body>
1974 </S:Envelope>

```

1975 Appendix C.3 First Acknowledgement

1976 Message number 2 has not been accepted by the RM Destination due to some transmission error so it
1977 responds with an Acknowledgement for messages 1 and 3:

```

1978 <?xml version="1.0" encoding="UTF-8"?>
1979 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
1980 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
1981 xmlns:wsa="http://www.w3.org/2005/08/addressing">
1982 <S:Header>
1983 <wsa:MessageID>
1984 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
1985 </wsa:MessageID>
1986 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
1987 <wsa:From>
1988 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
1989 </wsa:From>
1990 <wsa:Action>
1991 http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
1992 </wsa:Action>
1993 <wsrm:SequenceAcknowledgement>
1994 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
1995 <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
1996 <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
1997 </wsrm:SequenceAcknowledgement>
1998 </S:Header>
1999 <S:Body/>
2000 </S:Envelope>

```

2001 Appendix C.4 Retransmission

2002 The RM Sourcediscovers that message number 2 was not accepted so it resends the message and
2003 requests an Acknowledgement:

```

2004 <?xml version="1.0" encoding="UTF-8"?>
2005 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2006 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
2007 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2008 <S:Header>
2009 <wsa:MessageID>
2010 http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
2011 </wsa:MessageID>
2012 <wsa:To>http://example.com/serviceB/123</wsa:To>
2013 <wsa:From>
2014 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2015 </wsa:From>

```

```

2016 <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
2017 <wsrm:Sequence>
2018 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2019 <wsrm:MessageNumber>2</wsrm:MessageNumber>
2020 </wsrm:Sequence>
2021 <wsrm:AckRequested>
2022 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2023 </wsrm:AckRequested>
2024 </S:Header>
2025 <S:Body>
2026 <!-- Some Application Data -->
2027 </S:Body>
2028 </S:Envelope>

```

2029 Appendix C.5 Termination

2030 The RM Destination now responds with an Acknowledgement for the complete Sequence which can then
2031 be terminated:

```

2032 <?xml version="1.0" encoding="UTF-8"?>
2033 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2034 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
2035 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2036 <S:Header>
2037 <wsa:MessageID>
2038 http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
2039 </wsa:MessageID>
2040 <wsa:To>http://Business456.com/serviceA/789</wsa:To>
2041 <wsa:From>
2042 <wsa:Address>http://example.com/serviceB/123</wsa:Address>
2043 </wsa:From>
2044 <wsa:Action>
2045 http://docs.oasis-open.org/ws-rx/wsr/200608/SequenceAcknowledgement
2046 </wsa:Action>
2047 <wsrm:SequenceAcknowledgement>
2048 <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2049 <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
2050 </wsrm:SequenceAcknowledgement>
2051 </S:Header>
2052 <S:Body/>
2053 </S:Envelope>

```

2054 Terminate Sequence

```

2055 <?xml version="1.0" encoding="UTF-8"?>
2056 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2057 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsr/200608"
2058 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2059 <S:Header>
2060 <wsa:MessageID>
2061 http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2062 </wsa:MessageID>
2063 <wsa:To>http://example.com/serviceB/123</wsa:To>
2064 <wsa:Action>
2065 http://docs.oasis-open.org/ws-rx/wsr/200608/TerminateSequence
2066 </wsa:Action>
2067 <wsa:From>
2068 <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2069 </wsa:From>
2070 </S:Header>
2071 <S:Body>
2072 <wsrm:TerminateSequence>

```

```

2073     <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2074   </wsrm:TerminateSequence>
2075 </S:Body>
2076 </S:Envelope>

```

2077 Terminate Sequence Response

```

2078 <?xml version="1.0" encoding="UTF-8"?>
2079 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2080   xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrmp/200608"
2081   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2082   <S:Header>
2083     <wsa:MessageID>
2084       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
2085     </wsa:MessageID>
2086     <wsa:To>http://example.com/serviceA/789</wsa:To>
2087     <wsa:Action>
2088       http://docs.oasis-open.org/ws-rx/wsrmp/200608/TerminateSequenceResponse
2089     </wsa:Action>
2090     <wsa:RelatesTo>
2091       http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
2092     </wsa:RelatesTo>
2093     <wsa:From>
2094       <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
2095     </wsa:From>
2096   </S:Header>
2097   <S:Body>
2098     <wsrm:TerminateSequenceResponse>
2099       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
2100     </wsrm:TerminateSequenceResponse>
2101   </S:Body>
2102 </S:Envelope>

```

2103 Appendix C.6 MakeConnection

2104 To illustrate how a `MakeConnection` message exchange can be used to deliver messages to an
 2105 Endpoint that is not addressable, consider the case of a pub/sub scenario in which the Endpoint to which
 2106 notifications are to be delivered (the "event consumer") is not addressable by the notification sending
 2107 Endpoint (the "event producer"). In this scenario the event consumer must initiate the connections in order
 2108 for the notifications to be delivered. One possible set of message exchanges (using HTTP) that
 2109 demonstrate how this can be achieved using `MakeConnection` is shown below.

2110 **Step 1** – During a "subscribe" operation, the event consumer's EPR specifies the RM anonymous URI
 2111 and the RM Policy Assertion to indicate whether or not RM is required:

```

2112 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2113   xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrmp/200608"
2114   xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200608"
2115   xmlns:wsa="http://www.w3.org/2005/08/addressing">
2116   <S:Header>
2117     <wsa:To> http://example.org/subscriptionService </wsa:To>
2118     <wsa:MessageID> http://client456.org/id-a6d8-a7c2eb546813</wsa:MessageID>
2119     <wsa:ReplyTo>
2120       <wsa:To> http://client456.org/response </wsa:To>
2121     </wsa:ReplyTo>
2122   </S:Header>
2123   <S:Body>
2124     <sub:Subscribe xmlns:sub="http://example.org/subscriptionService">
2125       <!-- subscription service specific data -->
2126       <targetEPR>

```

```

2127     <wsa:Address>http://docs.oasis-open.org/ws-
2128 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:Address>
2129     <wsa:Metadata>
2130         <wsp:Policy wsu:Id="MyPolicy">
2131             <wsrmp:RMAssertion/>
2132         </wsp:Policy>
2133     </wsa:Metadata>
2134 </targetEPR>
2135 </sub:Subscribe>
2136 </S:Body>
2137 </S:Envelope>

```

2138 In this example the `subscribe` and `targetEPR` elements are simply examples of what a subscription
2139 request message might contain. Note: the `wsa:Address` element contains the RM anonymous URI
2140 indicating that the notification producer needs to queue the messages until they are requested using the
2141 `MakeConnection` message exchange. The EPR also contains the RM Policy Assertion indicating the RM
2142 must be used when notifications related to this subscription are sent.

2143 **Step 2** – Once the subscription is established, the event consumer checks for a pending message:

```

2144 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2145 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2146 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2147     <S:Header>
2148         <wsa:Action>http://docs.oasis-open.org/ws-
2149 rx/wsrn/200608/MakeConnection</wsa:Action>
2150         <wsa:To> http://example.org/subscriptionService </wsa:To>
2151     </S:Header>
2152     <S:Body>
2153         <wsrm:MakeConnection>
2154             <wsrm:Address>http://docs.oasis-open.org/ws-
2155 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-
2156 446655440000</wsrm:Address>
2157         </wsrm:MakeConnection>
2158     </S:Body>
2159 </S:Envelope>

```

2160 **Step 3** – If there are messages waiting to be delivered then a message will be returned back to the event
2161 consumer. However, because WS-RM is being used to deliver the messages, the first message returned
2162 is a `CreateSequence`:

```

2163 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2164 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrn/200608"
2165 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2166     <S:Header>
2167         <wsa:Action>http://docs.oasis-open.org/ws-
2168 rx/wsrn/200608/CreateSequence</wsa:Action>
2169         <wsa:To>http://docs.oasis-open.org/ws-
2170 rx/wsrn/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
2171         <wsa:ReplyTo> http://example.org/subscriptionService </wsa:ReplyTo>
2172         <wsa:MessageID> http://example.org/id-123-456 </wsa:MessageID>
2173     </S:Header>
2174     <S:Body>
2175         <wsrm:CreateSequence>
2176             <wsrm:AcksTo>
2177                 <wsa:Address> http://example.org/subscriptionService </wsa:Address>
2178             </wsrm:AcksTo>
2179         </wsrm:CreateSequence>
2180     </S:Body>

```

2181 </S:Envelope>

2182 Notice from the perspective of how the RM Source on the event producer interacts with the RM
2183 Destination of those messages, nothing new is introduced by the use of the `MakeConnection`, the use
2184 of RM protocol is the same as the case where the event consumer is addressable.

2185 **Step 4** – The event consumer will respond with a `CreateSequenceResponse` message per normal WS-
2186 Addressing rules:

```
2187 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2188 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2189 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2190   <S:Header>
2191     <wsa:Action>http://docs.oasis-open.org/ws-
2192 rx/wsmr/200608/CreateSequenceResponse</wsa:Action>
2193     <wsa:To> http://example.org/subscriptionService </wsa:To>
2194     <wsa:RelatesTo> http://example.org/id-123-456 </wsa:RelatesTo>
2195   </S:Header>
2196   <S:Body>
2197     <wsmr:CreateSequenceResponse>
2198       <wsmr:Identifier> http://example.org/rmid-456 </wsmr:Identifier>
2199     </wsmr:CreateSequenceResponse>
2200   </S:Body>
2201 </S:Envelope>
```

2202 Note, this message is carried on an HTTP request directed to the `wsa:ReplyTo` EPR, and the HTTP
2203 response will be an HTTP 202.

2204 **Step 5** – The event consumer checks for another message pending:

```
2205 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2206 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2207 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2208   <S:Header>
2209     <wsa:Action>http://docs.oasis-open.org/ws-
2210 rx/wsmr/200608/MakeConnection</wsa:Action>
2211     <wsa:To> http://example.org/subscriptionService </wsa:To>
2212   </S:Header>
2213   <S:Body>
2214     <wsmr:MakeConnection>
2215       <wsmr:Address>http://docs.oasis-open.org/ws-
2216 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-
2217 446655440000</wsmr:Address>
2218     </wsmr:MakeConnection>
2219   </S:Body>
2220 </S:Envelope>
```

2221 Notice this is the same message as the one sent in step 2.

2222 **Step 6** – If there is a message pending for this destination then it is returned on the HTTP response:

```
2223 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2224 xmlns:wsmr="http://docs.oasis-open.org/ws-rx/wsmr/200608"
2225 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2226   <S:Header>
2227     <wsa:Action> http://example.org/eventType1 </wsa:Action>
2228     <wsa:To>http://docs.oasis-open.org/ws-
2229 rx/wsmr/200608/anonymous?id=550e8400-e29b-11d4-a716-446655440000</wsa:To>
```

```

2230     <wsrm:Sequence>
2231         <wsrm:Identifier> http://example.org/rmid-456 </wsrm:Identifier>
2232     </wsrm:Sequence>
2233     <wsrm:MessagePending pending="true"/>
2234 </S:Header>
2235 <S:Body>
2236     <!-- event specific data -->
2237 </S:Body>
2238 </S:Envelope>

```

2239 As noted in step 3, the use of the RM protocol does not change when using `MakeConnection`. The
 2240 format of the messages, the order of the messages sent and the timing of when to send it remains the
 2241 same.

2242 **Step 7** – At some later interval, or immediately due to the `MessagePending` header's "pending"
 2243 attribute being set to "true", the event consumer will poll again:

```

2244 <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
2245 xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
2246 xmlns:wsa="http://www.w3.org/2005/08/addressing">
2247     <S:Header>
2248         <wsa:Action>http://docs.oasis-open.org/ws-
2249 rx/wsrm/200608/MakeConnection</wsa:Action>
2250         <wsa:To> http://example.org/subscriptionService </wsa:To>
2251     </S:Header>
2252     <S:Body>
2253         <wsrm:MakeConnection>
2254             <wsrm:Address>http://docs.oasis-open.org/ws-
2255 rx/wsrm/200608/anonymous?id=550e8400-e29b-11d4-a716-
2256 446655440000</wsrm:Address>
2257         </wsrm:MakeConnection>
2258     </S:Body>
2259 </S:Envelope>

```

2260 Notice this is the same message as the one sent in steps 2 and 5. As in steps 3 and 6, the response to
 2261 the `MakeConnection` can be any message destined to the specified Endpoint. This allows the event
 2262 producer to send not only application messages but RM protocol messages (e.g. `CloseSequence`,
 2263 `TerminateSequence` or even additional `CreateSequences`) as needed.

2264 **Step 8** – If at any point in time there are no messages pending, in response to a `MakeConnection` the
 2265 event producer returns an HTTP 202 back to the event consumer. The process then repeats (back to step
 2266 7) until the subscription ends.

Appendix D. State Tables

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

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

Legend:

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

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

Where:

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

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}

Where:

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

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

2294 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
Create Sequence [unspec] {3.1}	Xmit Create Sequence [Creating] {3.1}	N/A	N/A	N/A	N/A	N/A
Create Sequence Response [msg] {3.1}		Process Create Sequence Response [Created] {3.1}				
Create Sequence Refused Fault [msg] {3.1}		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] {2.1}	N/A	N/A	Xmit message [Same] {2.4}	Xmit message [Same] {2.4}	N/A	N/A
SeqAck (non-final) [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}	Process Ack ranges [Same] {3.6}
Nack [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	<Xmit message(s)> [Same] {3.6}	<Xmit message(s)> [Same] {3.6}	No action [Same]	No action [Same]
Message Number Rollover Fault [msg] {4.3}	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]
<Close Sequence> [int] {3.2}	N/A		Xmit Close Sequence [Closing] {3.2}	N/A	N/A	N/A
Close Sequence Response [msg] {3.2}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}		No action [Closed] {3.2}	No action [Same] {3.2}	No action [Same] {3.2}
SeqAck (final) [msg] {3.6}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Closed] {3.6}	Process Ack ranges [Closed] {3.6}	Process Ack ranges [Same]	Process Ack ranges [Same]
Sequence Closed Fault [msg] {4.7}	Generate Unknown Sequence Fault	Generate Unknown Sequence Fault	No action [Closed] {4.7}	No action [Closed] {4.7}	No action [Same]	No action [Same]

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
{4.7}	[Same] {4.3}	[Same] {4.3}				
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}
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] {3.3}
Expires exceeded [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
Invalid Acknowledgement [msg] {4.4}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}	Generate Invalid Acknowledgement Fault [Same] {4.4}

2295 Table 2 RM Destination Sequence State Transition Table

Events	Sequence States		
	None	Created	Closed
CreateSequence (successful) [msg/int] {3.1}	Xmit Create Sequence Response [Created] {3.1}	N/A	N/A
CreateSequence (unsuccessful) [msg/int] {3.1}	Generate Create Sequence Refused Fault [None] {3.1}	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] {3.2}
Message (with message number outside of range) [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Message Number Rollover Fault [Same] {3.4}{4.5}	Generate Sequence Closed Fault (with SeqAck+Final) [Same] {3.2}
<AckRequested> [msg] {3.5}	Generate Unknown Seq Fault [Same] {4.3}	Xmit SeqAck [Same] {3.5}	Xmit SeqAck+Final [Same] {3.6}

Events	Sequence States		
	None	Created	Closed
CloseSequence [msg] {3.2}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response with SeqAck+Final [Closed] {3.2}	Generate Sequence Closed Fault [Same] {4.7}
<CloseSequence autonomously> [int]	N/A	No Action [Closed]	N/A
TerminateSequence [msg] {3.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit Terminate Sequence Response [None] {3.3}	Xmit Terminate Sequence Response [None] {3.3}
UnknownSequence Fault [msg] {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}
Invalid Acknowledgement Fault [msg] {4.4}	N/A		
Expires exceeded [int]	N/A	Terminate Sequence [None] {3.4}	Terminate Sequence [None] {3.4}
<Seq Acknowledgement autonomously> [int] {3.6}	N/A	Xmit SeqAck [Same] {3.6}	Xmit SeqAck+Final [Same] {3.6}
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}

2296 The following two tables apply only if the `MakeConnection` mechanism is utilized.

2297 Table 3 Sending Endpoint Message Transfer Engine

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

2298 Table 4 Receiving Endpoint Message Transfer Engine

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

Appendix E. Acknowledgments

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

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

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

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

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

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

Appendix F. Revision History

Rev	Date	By Whom	What
wd-01	2005-07-07	Christopher Ferris	Initial version created based on submission by the authors.
ws-02	2005-07-21	Doug Davis	i011 (PT0S) added
wd-02	2005-08-16	Anish Karmarkar	Trivial editorial changes
ws-03	2005-09-15	Doug Davis	i019 and i028 (CloseSeq) added
wd-05	2005-09-26	Gilbert Pilz	i005 (Source resend of nacks messages when ack already received) added.
wd-05	2005-09-27	Doug Davis	i027 (InOrder delivery assurance spanning multiple sequences) added
wd-05	2005-09-27	Doug Davis	i020 (Semantics of "At most once" Delivery Assurance) added
wd-05	2005-09-27	Doug Davis	i034 (Fault while processing a piggy-backed RM header) added
wd-05	2005-09-27	Doug Davis	i033 (Processing model of NACKs) added
wd-05	2005-09-27	Doug Davis	i031 (AckRequested schema inconsistency) added
wd-05	2005-09-27	Doug Davis	i025 (SeqAck/None) added
wd-05	2005-09-27	Doug Davis	i029 (Remove dependency on WS-Security) added
wd-05	2005-09-27	Doug Davis	i039 (What does 'have a mU attribute' mean) added
wd-05	2005-09-27	Doug Davis	i040 (Change 'optiona'/'required' to 'OPTIONAL'/'REQUIRED') added
wd-05	2005-09-30	Anish Karmarkar	i017 (Change NS to 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
wd-07	2005-11-28	Doug Davis	i044

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

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

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

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

Rev	Date	By Whom	What
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.

Appendix G. Notices

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

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

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

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

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

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