Web Services Reliable Messaging (WS-ReliableMessaging)

Working Draft 16, January 29, 2007

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 Yalçinalp, 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

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:

    o "?" (0 or 1)

    o "*" (0 or more)

    o "+" (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 wsrm: 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 wsrm: 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/wsrm/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/wsrm/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 Conformance

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

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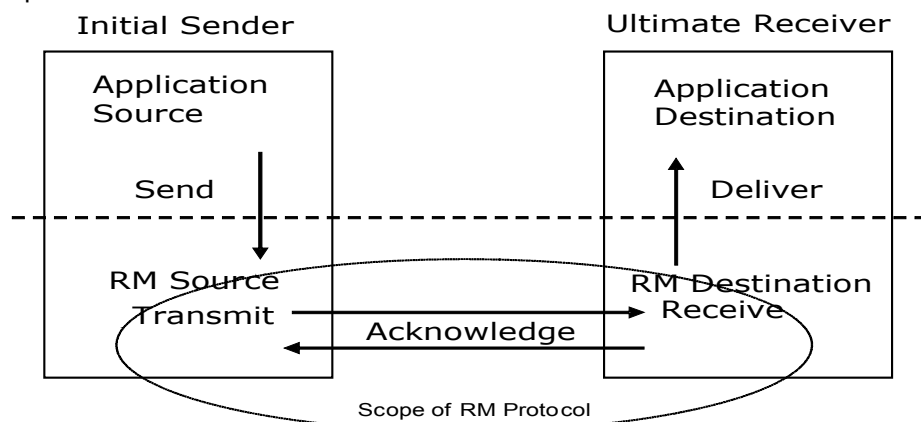
2 Reliable Messaging Model

161

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

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

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

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

187 Figure 1: Reliable Messaging Model

2.1 Glossary

189 The following definitions are used throughout this specification:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

218 2.2 Protocol Preconditions

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

221 • For any single message exchange the RM Source MUST have an endpoint reference that uniquely
222   identifies the RM Destination Endpoint.

223 • The RM Source MUST have successfully created a Sequence with the RM Destination.

224 • The RM Source MUST be capable of formulating messages that adhere to the RM Destination's
225   policies.

226 • If a secure exchange of messages is REQUIRED, then the RM Source and RM Destination MUST
227   have a security context.

228 2.3 Protocol Invariants

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

230 • The RM Source MUST assign each message within a Sequence a message number (defined
231 below) beginning at 1 and increasing by exactly 1 for each subsequent message. These numbers
232 MUST be assigned in the same order in which messages are sent by the Application Source.

233 • Within every Acknowledgement Message it issues, the RM Destination MUST include one or more
234 `AcknowledgementRange` child elements that contain, in their collective ranges, the message
235 number of every message accepted by the RM Destination. The RM Destination MUST exclude, in
236 the `AcknowledgementRange` elements, the message numbers of any messages it has not
237 accepted. If no messages have been received the RM Destination MUST return `None` instead of an
238 `AcknowledgementRange(s)`. The RM Destination MAY transmit a `Nack` for a specific message
239 or messages in stead of an `AcknowledgementRange(s)`.

240 • While the Sequence is not closed or terminated, the RM Source SHOULD retransmit
241 unacknowledged messages.

242 2.4 Example Message Exchange

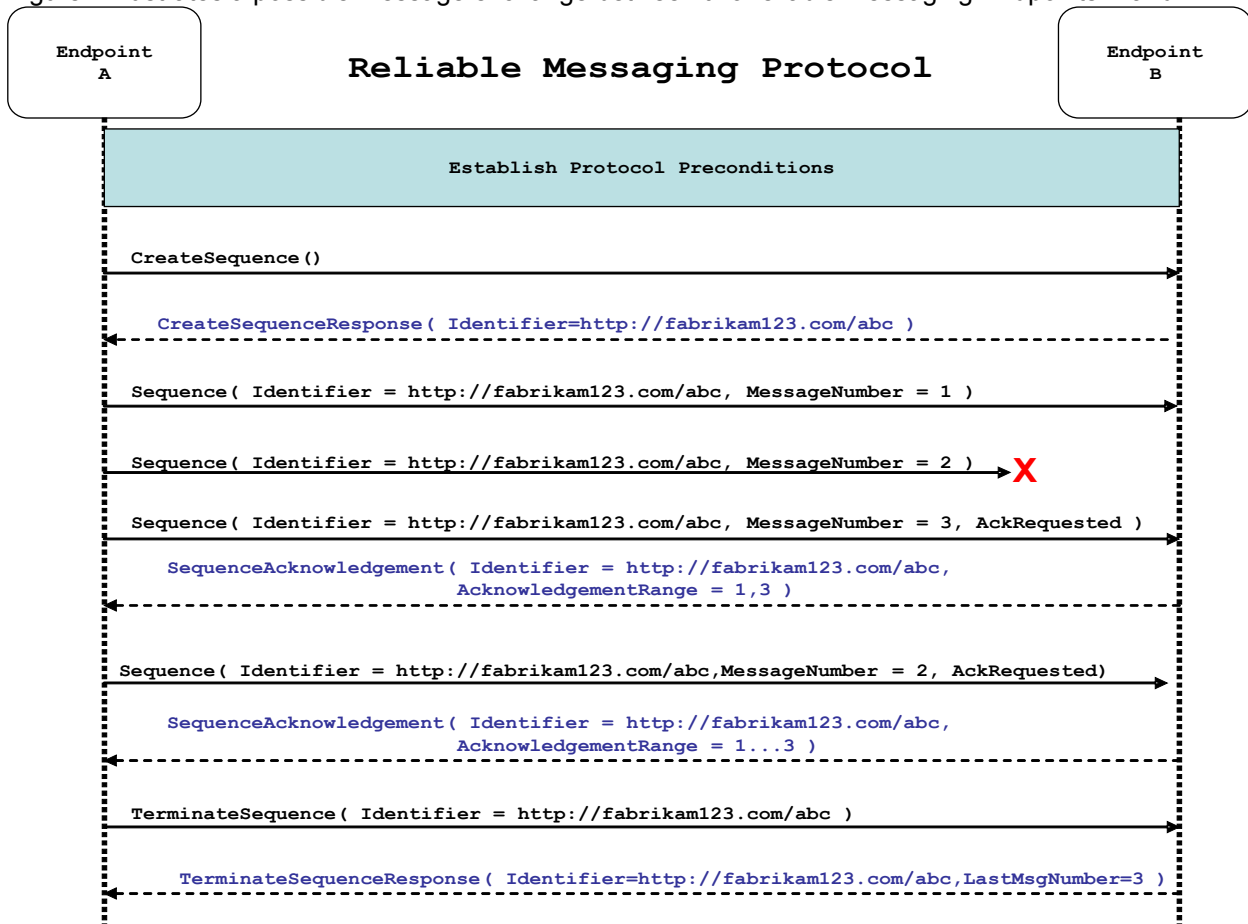243 Figure 2 illustrates a possible message exchange between two reliable messaging Endpoints A and B.

Figure 2: The WS-ReliableMessaging Protocol

244   1. The protocol preconditions are established. These include policy exchange, endpoint resolution,
245   and establishing trust.

246   2. The RM Source requests creation of a new Sequence.

247   3. The RM Destination creates a new Sequence and returns its unique identifier.

248   4. The RM Source begins Transmitting messages in the Sequence beginning with MessageNumber 1.
249   In the figure above, the RM Source sends 3 messages in the Sequence.

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

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

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

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

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

263   10. The RM Source Receives this Acknowledgement and sends a TerminateSequence message to the
264   RM Destination indicating that the Sequence is completed. The TerminateSequence message
265   indicates that message number 3 was the last message in the Sequence. The RM Destination then
266   reclaims any resources associated with the Sequence.

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

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

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

3 RM Protocol Elements

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

3.1 Considerations on the Use of Extensibility Points

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

3.2 Considerations on the Use of "Piggy-Backing"

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

3.3 Composition with WS-Addressing

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

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

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

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

The following exemplar defines the `CreateSequence` syntax:

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

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

/wsrm:CreateSequence

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

/wsrm:CreateSequence/wsrm:AcksTo

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

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

/wsrm:CreateSequence/wsrm:Expires

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

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

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

367 /wsrm:CreateSequence/wsrm:Offer

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

370 /wsrm:CreateSequence/wsrm:Offer/wsrm:Identifier

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

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

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

376 /wsrm:CreateSequence/wsrm:Offer/wsrm:Endpoint

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

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

385 The Offer of an Endpoint containing the "http://www.w3.org/2005/08/addressing/anonymous" IRI as its
386 address is problematic due to the inability of a source to connect to this address and retry
387 unacknowledged messages (as described in Section 2.3). When offering a Sequence with such an IRI, an
388 endpoint MUST ensure that it will provide the protocol back-channel opportunities necessary to carry
389 Sequence Traffic Messages for the offered Sequence, as well as any Sequence Lifecycle Messages
390 and/or Acknowledgement Requests that the anonymous RM Destination expects to receive for this
391 Sequence. In the absence of sufficient assurance that the endpoint offering the sequence will provide
392 these opportunities, an RM Destination MUST NOT accept (via the
393 /wsrm:CreateSequenceResponse/wsrm:Accept element described below) an Offer that contains the
394 "http://www.w3.org/2005/08/addressing/anonymous" IRI as its address.

395 /wsrm:CreateSequence/wsrm:Offer/wsrm:Expires

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

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

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

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

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

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

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

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

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

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

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

/wsrm:CreateSequence/{any}

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

/wsrm:CreateSequence/@{any}

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

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

The following exemplar defines the `CreateSequenceResponse` syntax:

```
<wsrm:CreateSequenceResponse ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
    <wsrm:IncompleteSequenceBehavior>
        wsrm:IncompleteSequenceBehaviorType
    </wsrm:IncompleteSequenceBehavior> ?
    <wsrm:Accept ...>
        <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
        ...
    </wsrm:Accept> ?
    ...
</wsrm:CreateSequenceResponse>
```

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

/wsrm:CreateSequenceResponse

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

/wsrm:CreateSequenceResponse/wsrm:Identifier

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

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

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

/wsrm:CreateSequenceResponse/wsrm:Expires

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

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

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

/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior

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

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

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

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

/wsrm:CreateSequenceResponse/wsrm:Accept

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

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

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

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

Implementations MUST NOT use an endpoint reference in the AcksTo element that would prevent the
sending of Sequence Acknowledgements back to the RM Source. For example, using the WS-Addressing

409 "http://www.w3.org/2005/08/addressing/none" IRI would make it impossible for the RM Destination to ever
410 send Sequence Acknowledgements.

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

409 /wsrm:CreateSequenceResponse/wsrm:Accept/@{any}
409 This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the
410 element.

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

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

409 ## 3.5 Closing A Sequence

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

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

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

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

409 While the RM Destination MUST NOT accept any new messages for the specified Sequence it MUST still
410 process Sequence Lifecyle Messages and Acknowledgement Requests. For example, it MUST respond to

409 AckRequested, TerminateSequence as well as CloseSequence messages. Note, subsequent
410 CloseSequence messages have no effect on the state of the Sequence.

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

409 The following exemplar defines the CloseSequence syntax:

```
<wsrm:CloseSequence ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:LastMsgNumber> wsrm:MessageNumberType </wsrm:LastMsgNumber> ?
    ...
</wsrm:CloseSequence>
```

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

409 /wsrm:CloseSequence

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

409 /wsrm:CloseSequence/wsrm:Identifier

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

409 /wsrm:CloseSequence/wsrm:LastMessageNumber

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

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

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

409 /wsrm:CloseSequence/{any}

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

409 /wsrm:CloseSequence@{any}

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

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

409 The following exemplar defines the `CloseSequenceResponse` syntax:

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

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

409 /wsrm:CloseSequenceResponse

409 This element is sent in the body of a message in response to receipt of a `CloseSequence` request

410 message. It indicates that the responder has closed the Sequence.

409 /wsrm:CloseSequenceResponse/wsrm:Identifier

409 The responder (RM Source or RM Destination) MUST include this element in any

410 `CloseSequenceResponse` message it sends. The responder MUST set the value of this element to the

411 absolute URI (conformant with RFC3986) of the closing Sequence.

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

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

410 element.

409 /wsrm:CloseSequenceResponse/{any}

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

410 to be passed.

409 /wsrm:CloseSequenceResponse@{any}

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

410 element.

409 3.6 Sequence Termination

409 When the RM Source has completed its use of the Sequence it sends a `TerminateSequence` element,

410 in the body of a message, to the RM Destination to indicate that the Sequence is complete and that it will

411 not be sending any further messages related to the Sequence. The RM Destination can safely reclaim any

412 resources associated with the Sequence upon receipt of the `TerminateSequence` message. Under

413 normal usage the RM Source will complete its use of the Sequence when all of the messages in the

414 Sequence have been acknowledged. However, the RM Source is free to Terminate or Close a Sequence

415 at any time regardless of the acknowledgement state of the messages.

409 To allow the RM Destination to determine if it has received all of the messages in a Sequence, the RM

410 Source SHOULD include the `LastMsgNumber` element in any TerminateSequence messages it sends.

411 The RM Destination can use this information, for example, to implement the behavior indicated by

412 `/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior`. The value of the

413 `LastMsgNumber` element in the TerminateSequence message MUST be equal to the value of the

414 `LastMsgNumber` element in any CloseSequence message(s) sent by the RM Source for the same

415 Sequence.

409 If the RM Destination decides to terminate a Sequence of its own volition, it MAY inform the RM Source of

410 this event by sending a `TerminateSequence` element, in the body of a message, to the AcksTo EPR for

411 that Sequence. The RM Destination MUST include a final `SequenceAcknowledgement` (within which

412 the RM Destination MUST include the `Final` element) header block in this message.

409 The following exemplar defines the TerminateSequence syntax:

```
<wsrm:TerminateSequence ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:LastMsgNumber> wsrm:MessageNumberType </wsrm:LastMsgNumber> ?
    ...
</wsrm:TerminateSequence>
```

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

/wsrm:TerminateSequence

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

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

/wsrm:TerminateSequence/wsrm:Identifier

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

/wsrm:TerminateSequence/wsrm:LastMsgNumber

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

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

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

/wsrm:TerminateSequence/{any}

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

/wsrm:TerminateSequence/@{any}

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

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

The following exemplar defines the `TerminateSequenceResponse` syntax:

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

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

/wsrm:TerminateSequenceResponse

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

/wsrm:TerminateSequenceResponse/wsrm:Identifier

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

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

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

/wsrm:TerminateSequenceResponse/{any}

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

/wsrm:TerminateSequenceResponse/@{any}

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

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

3.7 Sequences

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

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

A following exemplar defines its syntax:

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

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

/wsrm:Sequence

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

/wsrm:Sequence/wsrm:Identifier

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

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

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

409 /wsrm:Sequence/wsrm:MessageNumber

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

409 /wsrm:Sequence/{any}

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

409 /wsrm:Sequence/@{any}

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

409 The following example illustrates a Sequence header block.

```
409  <wsrm:Sequence>
409      <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
409      <wsrm:MessageNumber>10</wsrm:MessageNumber>
409  </wsrm:Sequence>
```

409 3.8 Request Acknowledgement

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

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

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

409 The following exemplar defines its syntax:

```
409  <wsrm:AckRequested ...>
409      <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
409      ...
409  </wsrm:AckRequested>
```

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

409 /wsrm:AckRequested

409 This element requests an Acknowledgement for the identified Sequence.

409 /wsrm:AckRequested/wsrm:Identifier

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

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

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

409 /wsrm:AckRequested/{any}

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

409 /wsrm:AckRequested/@{any}

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

409 3.9 Sequence Acknowledgement

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

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

409 During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the
410 address of the `AcksTo` EPR for that Sequence. When the RM Source specifies the WS-Addressing
411 anonymous IRI as the address of the `AcksTo` EPR, the RM Destination MUST Transmit any
412 `SequenceAcknowledgement` headers for the created Sequence in a SOAP envelope to be Transmitted
413 on the protocol binding-specific back-channel. Such a channel is provided by the context of a Received
414 message containing a SOAP envelope that contains a `Sequence` header block and/or an `AckRequested`
415 header block for that same Sequence identifier. When the RM Destination receives an `AckRequested`
416 header, and the `AckTo` EPR for that sequence is the WS-Addressing anonymous IRI, the RM Destination
417 SHOULD respond on the protocol binding-specific back-channel provided by the Received message
418 containing the `AckRequested` header block.

409 The following exemplar defines its syntax:

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

    ...
```

```
409    </wsrm:SequenceAcknowledgement>
```

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

409 /wsrm:SequenceAcknowledgement

409 This element contains the Sequence Acknowledgement information.

409 /wsrm:SequenceAcknowledgement/wsrm:Identifier

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

409 /wsrm:SequenceAcknowledgement/wsrm:Identifier/@{any}

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

409 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange

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

409 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper

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

409 /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower

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

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

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

409 /wsrm:SequenceAcknowledgement/wsrm:None

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

409 /wsrm:SequenceAcknowledgement/wsrm:Final

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

409 /wsrm:SequenceAcknowledgement/wsrm:Nack

409 The RM Destination MAY include this element within a `SequenceAcknowledgement` header block. If
410 used, the RM Destination MUST set the value of this element to a `MessageNumberType` representing
411 the `MessageNumber` of an unreceived message in a Sequence. The RM Destination MUST NOT include
412 a `Nack` element if a sibling `AcknowledgementRange` or `None` element is also present as a child of
413 `SequenceAcknowledgement`. Upon the receipt of a `Nack`, an RM Source SHOULD retransmit the
414 message identified by the `Nack`. The RM Destination MUST NOT issue a `SequenceAcknowledgement`
415 containing a `Nack` for a message that it has previously acknowledged within a
416 `AcknowledgementRange`. The RM Source SHOULD ignore a `SequenceAcknowledgement` containing
417 a `Nack` for a message that has previously been acknowledged within a `AcknowledgementRange`.

409 /wsrm:SequenceAcknowledgement/{any}

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

409 /wsrm:SequenceAcknowledgement/@{any}

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

409 The following examples illustrate `SequenceAcknowledgement` elements:

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

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

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

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

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

```
409    <wsrm:SequenceAcknowledgement>
409        <wsrm:Identifier>http://example.com/abc</wsrm:Identifier>
409        <wsrm:Nack>3</wsrm:Nack>
409    </wsrm:SequenceAcknowledgement>
```

4 Faults

Faults for the `CreateSequence` message exchange are treated as defined in WS-Addressing. Create Sequence Refused is a possible fault reply for this operation. Unknown Sequence is a fault generated by Endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences are detected. 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. Destinations that generate faults related to known sequences SHOULD transmit those faults. If transmitted, such faults MUST be transmitted to the same [destination] as Acknowledgement messages.

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

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

```
409          [Detail]
409           ...
409        </S:Detail>
409       </S:Fault>
409     </S:Body>
409    </S:Envelope>
```

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

```
409    <S11:Envelope>
409     <S11:Header>
409       <wsrm:SequenceFault>
409         <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
409         <wsrm:Detail> [Detail] </wsrm:Detail>
409         ...
409       </wsrm:SequenceFault>
409       <!-- Headers elided for brevity.  -->
409     </S11:Header>
409     <S11:Body>
409      <S11:Fault>
409       <faultcode> [Code] </faultcode>
409       <faultstring> [Reason] </faultstring>
409      </S11:Fault>
409     </S11:Body>
409    </S11:Envelope>
```

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

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

409 4.1 SequenceFault Element

409 The purpose of the SequenceFault element is to carry the specific details of a fault generated during
410 the reliable messaging specific processing of a message belonging to a Sequence. WS-
411 ReliableMessaging nodes MUST use the SequenceFault container only in conjunction with the SOAP
412 1.1 fault mechanism. WS-ReliableMessaging nodes MUST NOT use the SequenceFault container in
413 conjunction with the SOAP 1.2 binding.

409 The following exemplar defines its syntax:

```
409    <wsrm:SequenceFault ...>
409      <wsrm:FaultCode> wsrm:FaultCodes </wsrm:FaultCode>
409      <wsrm:Detail> ... </wsrm:Detail> ?
409      ...
409    </wsrm:SequenceFault>
```

409 The following describes the content model of the SequenceFault element.

409 /wsrm:SequenceFault

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

409 /wsrm:SequenceFault/wsrm:FaultCode

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

409 /wsrm:SequenceFault/wsrm:Detail

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

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

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

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

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

409 /wsrm:SequenceFault/{any}

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

409 /wsrm:SequenceFault/@{any}

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

409 4.2 Sequence Terminated

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

409 Properties:

409 [Code] Sender or Receiver

409 [Subcode] wsrm:SequenceTerminated

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

409 [Detail]

409   `<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.

409 4.3 Unknown Sequence

409 Properties:

409 [Code] Sender

409 [Subcode] wsrm:UnknownSequence

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

409 [Detail]

409 `<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.

409 4.4 Invalid Acknowledgement

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

409 [Code] Sender

409 [Subcode] wsrm:InvalidAcknowledgement

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

409 [Detail]

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

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

409 4.5 Message Number Rollover

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

409 Properties:

409 [Code] Sender

409 [Subcode] wsrm:MessageNumberRollover

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

409 [Detail]

```
409     <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
410     <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.

409 4.6 Create Sequence Refused

409 Properties:

409 [Code] Sender or Receiver

409 [Subcode] wsrm:CreateSequenceRefused

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

409 [Detail]

```
409     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.

409 4.7 Sequence Closed

409 This fault is generated by an RM Destination to indicate that the specified Sequence has been closed.
410 This fault MUST be generated when an RM Destination is asked to accept a message for a Sequence that
411 is closed.

409 Properties:

409 [Code] Sender

409 [Subcode] wsrm:SequenceClosed

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

409 [Detail]

```
409   <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.

409 4.8 WSRM Required

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

409 Properties:

409 [Code] Sender

409 [Subcode] wsrm:WSRMRequired

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

409 [Detail]

```
409   xs:any
```

5 Security Threats and Countermeasures

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

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

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

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

5.1 Threats and Countermeasures

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

5.1.1 Integrity Threats

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

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

5.1.1.1 Countermeasures

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

5.1.2 Resource Consumption Threats

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

5.1.2.1 Countermeasures

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

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

5.1.3 Sequence Spoofing Threats

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

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

5.1.3.1 Sequence Hijacking

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

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

5.1.3.2 Countermeasures

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

409 Source that initiated its creation (i.e. that sent the `CreateSequence` message) and the RM Destination that
410 serves as its terminus (i.e. that sent the `CreateSequenceResponse` message). To counter sequence
411 spoofing attempts the RM Destination SHOULD ensure that every message or fault that it Receives that
412 refers to a particular Sequence originated from the RM Source that jointly owns the referenced Sequence.
413 For its part the RM Source SHOULD ensure that every message or fault that it Receives that refers to a
414 particular Sequence originated from the RM Destination that jointly owns the referenced Sequence.

409 For the RM Destination to be able to identify its sequence peer it MUST be able to identify and
410 authenticate the entity that sent the `CreateSequence` message. Similarly for the RM Source to identify its
411 sequence peer it MUST be able to identify and authenticate the entity that sent the
412 `CreateSequenceResponse` message. For either the RM Destination or the RM Source to determine if a
413 message was sent by its sequence peer it MUST be able to identify and authenticate the initiator of that
414 message and, if necessary, correlate this identity with the sequence peer identity established at sequence
415 creation time.

5.2 Security Solutions and Technologies

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

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

5.2.1 Transport Layer Security

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

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

5.2.1.1 Model

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

409   1. The RM Source establishes an SSL/TLS session with the RM Destination.

409   2. The RM Source uses this SSL/TLS session to send a `CreateSequence` message to the RM
410   Destination.

409   3. The RM Destination establishes an SSL/TLS session with the RM Source and sends an
410   asynchronous `CreateSequenceResponse` using this session. Alternately it may respond with a
411   synchronous `CreateSequenceResponse` using the session established in (1).

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

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

5.2.1.2 Countermeasure Implementation

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

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

409   • **HTTP Basic Authentication**: This method of authentication presupposes that a SOAP/HTTP
410   binding is being used as part of the protocol stack beneath WS-RM. Subsequent to the
411   establishment of the SSL/TLS session, the sending party authenticates itself to the receiving party
412   using HTTP Basic Authentication [RFC 2617]. For example, a RM Source might authenticate itself
413   to a RM Destination (e.g. when transmitting a Sequence Traffic Message) using BasicAuth.
414   Similarly the RM Destination might authenticate itself to the RM Source (e.g. when sending an
415   Acknowledgement) using BasicAuth.

409   • **SSL/TLS Client Authentication:** In this method of authentication, the party initiating the
410   connection authenticates itself to the party accepting the connection using an X.509 certificate
411   that is exchanged during the SSL/TLS handshake.

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

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

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

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

5.2.2 SOAP Message Security

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

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

5.2.2.1 Model

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

1. The RM Source and the RM Destination create a WS-SecureConversation security context. This may involve the participation of third parties such as a security token service. The tokens exchanged may contain authentication claims (e.g. X.509 certificates or Kerberos service tickets).

2. During the `CreateSequence` exchange, the RM Source SHOULD explicitly identify the security context that will be used to protect the Sequence. This is done so that, in cases where the `CreateSequence` message is signed by more than one security context, the RM Source can indicate which security context should be used to protect the newly created Sequence.

3. For the lifetime of the Sequence the RM Source and the RM Destination use the session key(s) associated with the security context to sign (as defined by WS-Security) at least the body and any relevant WS-RM-defined headers of any and all messages or faults that refer to that Sequence.

5.2.2.2 Countermeasure Implementation

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

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

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

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

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

409 Issues specific to the life-cycle management of WS-SecureConversation security contexts (such as
410 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>
            wsrm:IncompleteSequenceBehaviorType
        </wsrm:IncompleteSequenceBehavior> ?
        ...
    </wsrm:Offer> ?
    ...
    <wsse:SecurityTokenReference>
        ...
    </wsse:SecurityTokenReference> ?
    ...
</wsrm:CreateSequence>
```

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

/wsrm:CreateSequence/wsse:SecurityTokenReference

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

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

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

The following exemplar defines the `UsesSequenceSTR` syntax:

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

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

/wsrm:UsesSequenceSTR

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

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

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

6.2 Securing Sequences Using SSL/TLS

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

The following exemplar defines the `UsesSequenceSSL` syntax:

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

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

/wsrm:UsesSequenceSSL

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

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

409 Note that the use inclusion of the above header by the RM Source implies that all Sequence-related
410 information (Sequence Lifecycle or Acknowledgment messages or Sequence-related faults) flowing from
411 the RM Destination to the RM Source will be bound to the SSL/TLS session that is used to carry the
412 `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

http://www.ietf.org/rfc/rfc2119.txt

[WS-RM Policy]

OASIS WS-RX Technical Committee Draft, "Web Services ReliableMessaging Policy Assertion(WS-RM Policy)" October 2006

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

[SOAP 1.1]

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

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

[SOAP 1.2]

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

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

[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.

http://ietf.org/rfc/rfc3986

[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

http://www.ietf.org/rfc/rfc4122.txt

[XML]

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

http://www.w3.org/TR/REC-xml/

[XML-ns]

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

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

[XML-Schema Part1]

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

http://www.w3.org/TR/xmlschema-1/

[XML-Schema Part2]

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

http://www.w3.org/TR/xmlschema-2/

[XPATH 1.0]

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

http://www.w3.org/TR/xpath

[WSDL 1.1]

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

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

[WS-Addressing]

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

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

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

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

7.2 Non-Normative

[BSP 1.0]

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

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

[RDDL 2.0]

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

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

[RFC 2617]

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

http://www.ietf.org/rfc/rfc2617.txt

[RFC 4346]

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

http://www.ietf.org/rfc/rfc4346.txt

[WS-Policy]

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

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

[WS-PolicyAttachment]

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

http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/

[WS-Security]

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

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

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

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

[RTTM]

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

http://www.rfc-editor.org/rfc/rfc1323.txt

[SecurityPolicy]

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

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

[SecureConversation]

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

http://schemas.xmlsoap.org/ws/2004/04/sc/

[Trust]

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

http://schemas.xmlsoap.org/ws/2005/02/trust

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/wsrm/200608/wsrm-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/wsrm/200608"
targetNamespace="http://docs.oasis-open.org/ws-rx/wsrm/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>
```

```
409        <xs:anyAttribute namespace="##other" processContents="lax"/>
410      </xs:complexType>
411      <xs:element name="Sequence" type="wsrm:SequenceType"/>
412      <xs:element name="SequenceAcknowledgement">
413        <xs:complexType>
414          <xs:sequence>
415            <xs:element ref="wsrm:Identifier"/>
416            <xs:choice>
417              <xs:sequence>
418                <xs:choice>
419                  <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
420                    <xs:complexType>
421                      <xs:sequence/>
422                      <xs:attribute name="Upper" type="xs:unsignedLong"
423  use="required"/>
424                      <xs:attribute name="Lower" type="xs:unsignedLong"
425  use="required"/>
426                      <xs:anyAttribute namespace="##other" processContents="lax"/>
427                    </xs:complexType>
428                  </xs:element>
429                  <xs:element name="None">
430                    <xs:complexType>
431                      <xs:sequence/>
432                    </xs:complexType>
433                  </xs:element>
434                </xs:choice>
435                <xs:element name="Final" minOccurs="0">
436                  <xs:complexType>
437                    <xs:sequence/>
438                  </xs:complexType>
439                </xs:element>
440              </xs:sequence>
441              <xs:element name="Nack" type="xs:unsignedLong"
442  maxOccurs="unbounded"/>
443            </xs:choice>
444            <xs:any namespace="##other" processContents="lax" minOccurs="0"
445  maxOccurs="unbounded"/>
446          </xs:sequence>
447          <xs:anyAttribute namespace="##other" processContents="lax"/>
448        </xs:complexType>
449      </xs:element>
450      <xs:complexType name="AckRequestedType">
451        <xs:sequence>
452          <xs:element ref="wsrm:Identifier"/>
453          <xs:any namespace="##other" processContents="lax" minOccurs="0"
454  maxOccurs="unbounded"/>
455        </xs:sequence>
456        <xs:anyAttribute namespace="##other" processContents="lax"/>
457      </xs:complexType>
458      <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
459      <xs:element name="Identifier">
460        <xs:complexType>
461          <xs:annotation>
462            <xs:documentation>
463              This type is for elements whose [children] is an anyURI and can have
464  arbitrary attributes.
465            </xs:documentation>
466          </xs:annotation>
467          <xs:simpleContent>
468            <xs:extension base="xs:anyURI">
469              <xs:anyAttribute namespace="##other" processContents="lax"/>
470            </xs:extension>
471          </xs:simpleContent>
```

```
409        </xs:complexType>
410      </xs:element>
411      <xs:element name="Address">
412        <xs:complexType>
413          <xs:simpleContent>
414            <xs:extension base="xs:anyURI">
415              <xs:anyAttribute namespace="##other" processContents="lax"/>
416            </xs:extension>
417          </xs:simpleContent>
418        </xs:complexType>
419      </xs:element>
420      <xs:simpleType name="MessageNumberType">
421        <xs:restriction base="xs:unsignedLong">
422          <xs:minInclusive value="1"/>
423          <xs:maxInclusive value="9223372036854775807"/>
424        </xs:restriction>
425      </xs:simpleType>
426      <!-- Fault Container and Codes -->
427      <xs:simpleType name="FaultCodes">
428        <xs:restriction base="xs:QName">
429          <xs:enumeration value="wsrm:SequenceTerminated"/>
430          <xs:enumeration value="wsrm:UnknownSequence"/>
431          <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
432          <xs:enumeration value="wsrm:MessageNumberRollover"/>
433          <xs:enumeration value="wsrm:CreateSequenceRefused"/>
434          <xs:enumeration value="wsrm:SequenceClosed"/>
435          <xs:enumeration value="wsrm:WSRMRequired"/>
436          <xs:enumeration value="wsrm:UnsupportedSelection"/>
437        </xs:restriction>
438      </xs:simpleType>
439      <xs:complexType name="SequenceFaultType">
440        <xs:sequence>
441          <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
442          <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
443          <xs:any namespace="##other" processContents="lax" minOccurs="0"
444  maxOccurs="unbounded"/>
445        </xs:sequence>
446        <xs:anyAttribute namespace="##other" processContents="lax"/>
447      </xs:complexType>
448      <xs:complexType name="DetailType">
449        <xs:sequence>
450          <xs:any namespace="##other" processContents="lax" minOccurs="0"
451  maxOccurs="unbounded"/>
452        </xs:sequence>
453        <xs:anyAttribute namespace="##other" processContents="lax"/>
454      </xs:complexType>
455      <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
456      <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
457      <xs:element name="CreateSequenceResponse"
458  type="wsrm:CreateSequenceResponseType"/>
459      <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
460      <xs:element name="CloseSequenceResponse"
461  type="wsrm:CloseSequenceResponseType"/>
462      <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
463      <xs:element name="TerminateSequenceResponse"
464  type="wsrm:TerminateSequenceResponseType"/>
465      <xs:complexType name="CreateSequenceType">
466        <xs:sequence>
467          <xs:element ref="wsrm:AcksTo"/>
468          <xs:element ref="wsrm:Expires" minOccurs="0"/>
469          <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
470          <xs:any namespace="##other" processContents="lax" minOccurs="0"
471  maxOccurs="unbounded">
```

```
409          <xs:annotation>
410            <xs:documentation>
411              It is the authors intent that this extensibility be used to
412  transfer a Security Token Reference as defined in WS-Security.
413            </xs:documentation>
414          </xs:annotation>
415        </xs:any>
416      </xs:sequence>
417      <xs:anyAttribute namespace="##other" processContents="lax"/>
418    </xs:complexType>
419    <xs:complexType name="CreateSequenceResponseType">
420      <xs:sequence>
421        <xs:element ref="wsrm:Identifier"/>
422        <xs:element ref="wsrm:Expires" minOccurs="0"/>
423        <xs:element name="IncompleteSequenceBehavior"
424  type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
425        <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
426        <xs:any namespace="##other" processContents="lax" minOccurs="0"
427  maxOccurs="unbounded"/>
428      </xs:sequence>
429      <xs:anyAttribute namespace="##other" processContents="lax"/>
430    </xs:complexType>
431    <xs:complexType name="CloseSequenceType">
432      <xs:sequence>
433        <xs:element ref="wsrm:Identifier"/>
409        <xs:element ref="wsrm:MessageNumberType"/>
410        <xs:any namespace="##other" processContents="lax" minOccurs="0"
411  maxOccurs="unbounded"/>
412      </xs:sequence>
413      <xs:anyAttribute namespace="##other" processContents="lax"/>
414    </xs:complexType>
415    <xs:complexType name="CloseSequenceResponseType">
416      <xs:sequence>
417        <xs:element ref="wsrm:Identifier"/>
418        <xs:any namespace="##other" processContents="lax" minOccurs="0"
419  maxOccurs="unbounded"/>
420      </xs:sequence>
421      <xs:anyAttribute namespace="##other" processContents="lax"/>
422    </xs:complexType>
423    <xs:complexType name="TerminateSequenceType">
424      <xs:sequence>
425        <xs:element ref="wsrm:Identifier"/>
409        <xs:element ref="wsrm:MessageNumberType" minOccurs="0"/>
410        <xs:any namespace="##other" processContents="lax" minOccurs="0"
411  maxOccurs="unbounded"/>
412      </xs:sequence>
413      <xs:anyAttribute namespace="##other" processContents="lax"/>
414    </xs:complexType>
415    <xs:complexType name="TerminateSequenceResponseType">
416      <xs:sequence>
417        <xs:element ref="wsrm:Identifier"/>
418        <xs:any namespace="##other" processContents="lax" minOccurs="0"
419  maxOccurs="unbounded"/>
420      </xs:sequence>
421      <xs:anyAttribute namespace="##other" processContents="lax"/>
422    </xs:complexType>
423    <xs:element name="AcksTo" type="wsa:EndpointReferenceType"/>
424    <xs:complexType name="OfferType">
425      <xs:sequence>
426        <xs:element ref="wsrm:Identifier"/>
427        <xs:element name="Endpoint" type="wsa:EndpointReferenceType"/>
428        <xs:element ref="wsrm:Expires" minOccurs="0"/>
429        <xs:element name="IncompleteSequenceBehavior"
```

```
409    type="wsrm:IncompleteSequenceBehaviorType" minOccurs="0"/>
410            <xs:any namespace="##other" processContents="lax" minOccurs="0"
411    maxOccurs="unbounded"/>
412        </xs:sequence>
413        <xs:anyAttribute namespace="##other" processContents="lax"/>
414      </xs:complexType>
415      <xs:complexType name="AcceptType">
416        <xs:sequence>
417          <xs:element ref="wsrm:AcksTo"/>
418          <xs:any namespace="##other" processContents="lax" minOccurs="0"
419    maxOccurs="unbounded"/>
420        </xs:sequence>
421        <xs:anyAttribute namespace="##other" processContents="lax"/>
422      </xs:complexType>
423      <xs:element name="Expires">
424        <xs:complexType>
425          <xs:simpleContent>
426            <xs:extension base="xs:duration">
427              <xs:anyAttribute namespace="##other" processContents="lax"/>
428            </xs:extension>
429          </xs:simpleContent>
430        </xs:complexType>
431      </xs:element>
432      <xs:simpleType name="IncompleteSequenceBehaviorType">
433        <xs:restriction base="xs:string">
434          <xs:enumeration value="DiscardEntireSequence"/>
435          <xs:enumeration value="DiscardFollowingFirstGap"/>
436          <xs:enumeration value="NoDiscard"/>
437        </xs:restriction>
438      </xs:simpleType>
409      <xs:element name="UsesSequenceSTR">
409        <xs:complexType>
409          <xs:sequence/>
409          <xs:anyAttribute namespace="##other" processContents="lax"/>
409        </xs:complexType>
409      </xs:element>
409      <xs:element name="UsesSequenceSSL">
409        <xs:complexType>
409          <xs:sequence/>
409          <xs:anyAttribute namespace="##other" processContents="lax"/>
409        </xs:complexType.
409      </xs:element>
410      <xs:element name="UnsupportedElement">
411        <xs:simpleType>
412          <xs:restriction base="xs:QName"/>
413        </xs:simpleType>
414      </xs:element>
415    </xs:schema>
```

Appendix B. WSDL

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

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

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

http://docs.oasis-open.org/ws-rx/wsrm/200608/wsdl/wsrm-1.1-wsdl-200608.wsdl

The following non-normative copy is provided for reference.

```xml
<?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/wsrm/200608" xmlns:tns="http://docs.oasis-open.org/ws-
rx/wsrm/200608/wsdl" targetNamespace="http://docs.oasis-open.org/ws-
rx/wsrm/200608/wsdl">

    <wsdl:types>
```

```
409        <xs:schema>
410          <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrm/200608"
411    schemaLocation="http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-schema-
412    200608.xsd"/>
413        </xs:schema>
414      </wsdl:types>

415      <wsdl:message name="CreateSequence">
416        <wsdl:part name="create" element="rm:CreateSequence"/>
417      </wsdl:message>
418      <wsdl:message name="CreateSequenceResponse">
419        <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
420      </wsdl:message>
421      <wsdl:message name="CloseSequence">
422        <wsdl:part name="close" element="rm:CloseSequence"/>
423      </wsdl:message>
424      <wsdl:message name="CloseSequenceResponse">
425        <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
426      </wsdl:message>
427      <wsdl:message name="TerminateSequence">
428        <wsdl:part name="terminate" element="rm:TerminateSequence"/>
429      </wsdl:message>
430      <wsdl:message name="TerminateSequenceResponse">
431        <wsdl:part name="terminateResponse"
432    element="rm:TerminateSequenceResponse"/>
433      </wsdl:message>

434      <wsdl:portType name="SequenceAbstractPortType">
435        <wsdl:operation name="CreateSequence">
436          <wsdl:input message="tns:CreateSequence" wsaw:Action="http://docs.oasis-
437    open.org/ws-rx/wsrm/200608/CreateSequence"/>
438          <wsdl:output message="tns:CreateSequenceResponse"
439    wsaw:Action="http://docs.oasis-open.org/ws-
440    rx/wsrm/200608/CreateSequenceResponse"/>
441        </wsdl:operation>
442        <wsdl:operation name="CloseSequence">
443          <wsdl:input message="tns:CloseSequence" wsaw:Action="http://docs.oasis-
444    open.org/ws-rx/wsrm/200608/CloseSequence"/>
445          <wsdl:output message="tns:CloseSequenceResponse"
446    wsaw:Action="http://docs.oasis-open.org/ws-
447    rx/wsrm/200608/CloseSequenceResponse"/>
448        </wsdl:operation>
449        <wsdl:operation name="TerminateSequence">
450          <wsdl:input message="tns:TerminateSequence"
451    wsaw:Action="http://docs.oasis-open.org/ws-rx/wsrm/200608/TerminateSequence"/>
452          <wsdl:output message="tns:TerminateSequenceResponse"
453    wsaw:Action="http://docs.oasis-open.org/ws-
454    rx/wsrm/200608/TerminateSequenceResponse"/>
455        </wsdl:operation>
456      </wsdl:portType>

457    </wsdl:definitions>
```

Appendix C. Message Examples

Appendix C.1   Create Sequence

Create Sequence

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/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/wsrm/200608/CreateSequence</wsa:Action>
  <wsa:ReplyTo>
   <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:ReplyTo>
 </S:Header>
 <S:Body>
  <wsrm:CreateSequence>
    <wsrm:AcksTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsrm:AcksTo>
  </wsrm:CreateSequence>
 </S:Body>
</S:Envelope>
```

Create Sequence Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/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/wsrm/200608/CreateSequenceResponse
     </wsa:Action>
   </S:Header>
   <S:Body>
     <wsrm:CreateSequenceResponse>
       <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
     </wsrm: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:

Message 1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200608"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfcbc9e
    </wsa:MessageID>
    <wsa:To>http://example.com/serviceB/123</wsa:To>
    <wsa:From>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:From>
    <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
    <wsrm:Sequence>
      <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
      <wsrm:MessageNumber>1</wsrm:MessageNumber>
    </wsrm:Sequence>
  </S:Header>
  <S:Body>
    <!--  Some  Application  Data  -->
  </S:Body>
</S:Envelope>
```

Message 2

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

Message 3

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

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

Appendix C.3   First Acknowledgement

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

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

Appendix C.4   Retransmission

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

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

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

Appendix C.5   Termination

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

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

Terminate Sequence

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

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

409 **Terminate Sequence Response**

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

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
Event name [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
Action to take [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.

409 Table 1 RM Source Sequence State Transition Table

Events	Sequence States					
	None	Creating	Created	Closing	Closed	Terminating
Create Sequence [unspec] {3.4}	Xmit Create Sequence [Creating] {3.4}	N/A	N/A	N/A	N/A	N/A
Create Sequence Response [msg] {3.4}		Process Create Sequence Response [Created] {3.4}				
Create Sequence Refused Fault [msg] {3.4}		No action [None] {4.6}				
Send message [app] {2.1}	N/A	N/A	Xmit message [Same] {2}	No action [Same] {2}	N/A	N/A
Retransmit of un-ack'd message [int]	N/A	N/A	Xmit message [Same] {2.4}	Xmit message [Same] {2.4}	N/A	N/A
SeqAck (non-final) [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}	Process Ack ranges [Same] {3.9}
Nack [msg] {3.9}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	<Xmit message(s)> [Same] {3.9}	<Xmit message(s)> [Same] {3.9}	No action [Same]	No action [Same]
Message Number Rollover Fault [msg]	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	No action [Rollover]	No action [Same]	No action [Same]	No action [Same]
CloseSequence [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Xmit CloseSequence Response [Closed] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}
<Close Sequence> [int] {3.5}	N/A		Xmit Close Sequence [Closing] {3.5}	N/A	N/A	N/A
Close Sequence Response [msg] {3.5}	Generate Unknown Sequence Fault [Same] {4.3}	Generate Unknown Sequence Fault [Same] {4.3}		No action [Closed] {3.5}	No action [Same] {3.5}	No action [Same] {3.5}

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

409   Table 2 RM Destination Sequence State Transition Table

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

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

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

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(BTplc), 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 Raepple(SAP), Eric Rajkovic(Oracle), Stefan Rossmanith(SAP), Tom Rutt(Fujitsu), Rich Salz(IBM), Shivajee Samdarshi(Tibco), Vladimir Videlov(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/wsrm/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 wsrm: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 formating 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 editoral 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 RDDL doc; added non-normative reference to RDDL 2.0
wd-10	2006-02-17	Anish Karmarkar	Namespace changed to 200602 for both WSDL and XSD docs.
wd-10	2006-02-17	Anish Karmarkar	Issue i087 as it applies to WSRM spec.
wd-10	2006-02-17	Anish Karmarkar	Added titles and minor text for state table (issue i058).
wd-11	2006-02-22	Doug Davis	Accept all changes for new WD

Minor typos fixed |
| wd-11 | 2006-02-23 | Doug Davis | s/'close'/close/g – per Marc Goodner

Added first ref to [URI] – per Marc G again |
wd-11	2006-02-27	Doug Davis	Issue i061 applied
wd-11	2006-02-28	Doug Davis	Fixed typo around the use of "above" and "below"
wd-11	2006-03-01	Doug Davis	Minor typos found by Marc Goodner
wd-11	2006-03-02	Doug Davis	Minor typos found by Matt Lovett
wd-11	2006-03-08	Doug Davis	Issue 091 applied
wd-11	2006-03-08	Doug Davis	Issue 092 applied
wd-11	2006-03-08	Doug Davis	Issue 100 applied

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

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

Rev	Date	By Whom	What
cd-04	2006-08-11	Doug Davis	Issue 158 applied
cd-04	2006-08-11	Doug Davis	Issue 153 applied
cd-04	2006-08-11	Doug Davis	Issue 156 applied
cd-04	2006-08-15	Gilbert Pilz	More formatting changes for better HTML rendering.
wd-16	2006-10-25	Doug Davis	Accept all changes, update to wd16
wd-16	2006-10-26	Doug Davis	PR002 applied
wd-16	2006-10-26	Doug Davis	PR003 applied
wd-16	2006-10-26	Doug Davis	PR004 applied
wd-16	2006-10-27	Doug Davis	PR005 applied
wd-16	2006-10-27	Doug Davis	PR006 applied
wd-16	2006-10-27	Doug Davis	PR024 applied
wd-16	2006-11-13	Doug Davis	PR010 applied
wd-16	2006-11-13	Doug Davis	PR011 applied (technically as part of PR004)
wd-16	2006-11-13	Doug Davis	PR016 applied
wd-16	2006-11-13	Doug Davis	PR032 applied
wd-16	2006-11-20	Doug Davis	PR025 applied
wd-16	2006-11-20	Doug Davis	PR023 applied
wd-16	2006-12-03	Doug Davis	PR036 applied
wd-16	2006-12-03	Doug Davis	PR017 applied
wd-16	2006-12-11	Doug Davis	PR012 applied (and PR013)
wd-16	2006-12-14	Doug Davis	PR033 applied – changed a 'return' to 'generate' when talking about a fault
wd-16	2007-01-04	Doug Davis	PR018 applied
wd-16	2007-01-05	Doug Davis	Moved MakeConnection to new spec
wd-16	2007-01-17	Doug Davis	PR026 applied
wd-16	2007-01-18	Doug Davis	PR021 applied
wd-16	2007-01-18	Doug Davis	PR022 applied
wd-16	2007-01-18	Doug Davis	Fixed a few typos (Doug,Gil)
wd-16	2007-01-18	Gilbert Pilz	PR007 applied
wd-16	2007-01-25	Doug Davis	PR039 applied

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.