# Web Services Reliable Messaging (WS-ReliableMessaging)

## Working Draft 14, June 8, 2006

**Document identifier:**

wsrm-1.1-spec-wd-14

**Location:**

**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:**

TBD

**Abstract:**

This specification (WS-ReliableMessaging) describes a protocol that allows messages to be delivered reliably between distributed applications 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 protocols related to the operation of distributed Web services.

**Status:**

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

# 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 delivery of messages. It defines a messaging protocol to identify, track, and manage the reliable delivery of messages between a source and a destination. It also defines a SOAP binding that is required for interoperability. Additional bindings may be defined.

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

## 1.1 Notational Conventions

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

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

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

## 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/200604
```

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/200604 |
| wsa | http://www.w3.org/2005/08/addressing |
| xs | http://www.w3.org/2001/XMLSchema |

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

http://docs.oasis-open.org/ws-rx/wsrm/200604/wsrm-1.1-schema-200604.xsd

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

## 1.3 Compliance

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

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

# 2 Reliable Messaging Model

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

The WS-ReliableMessaging specification defines an interoperable protocol that requires a Reliable Messaging (RM) Source and Reliable Messaging Destination to ensure that each message transmitted by the RM Source is successfully received by an RM Destination, or barring successful receipt, that an RM Source can, except in the most extreme circumstances, accurately determine the disposition of each message transmitted as perceived by the RM Destination, so as to resolve any in-doubt status. Note that this specification makes no restriction on the scope of the RM Source or RM Destination entities. For example, either may span multiple WSDL Ports or endpoints.

The protocol supports reliability features that enable ordered delivery, duplicate elimination, and guaranteed receipt for the RM Destination. It is expected that the Application Destination and RM Destination will implement as many of these or as few of these characteristics as necessary. Regardless of which of the reliability features are employed, the wire protocol does not change.

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

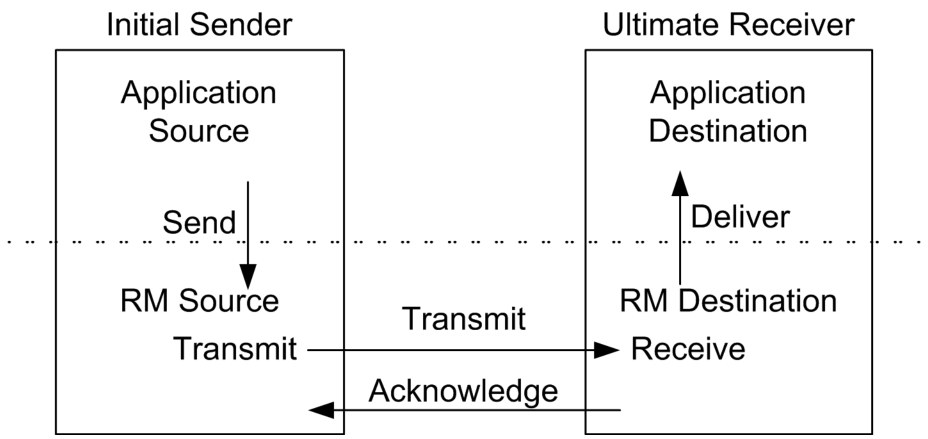

Figure 1: Reliable Messaging Model

## 2.1 Glossary

The following definitions are used throughout this specification:

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

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

**Application Source:** The endpoint that sends a message.

**Deliver:** The act of transferring a message from the RM Destination to the Application Destination. The reliability guarantee is fulfilled at this point.

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

**Receive:** The act of reading a message from a network connection and qualifying it as relevant to RM Destination functions.

**RM Destination:** For any one reliably sent message the endpoint that receives the message.

**RM Source:** The endpoint that transmits the message.

**Send:** The act of submitting a message to the RM Source for reliable delivery. The reliability guarantee begins at this point.

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

## 2.2 Protocol Preconditions

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

- For any single message exchange the RM Source MUST have an endpoint reference that uniquely identifies the RM Destination endpoint.
- The RM Source MUST have knowledge of the destination's policies, if any, and the RM Source MUST be capable of formulating messages that adhere to this policy.

If a secure exchange of messages is required, then the RM Source and RM Destination MUST have a security context.

## 2.3 Protocol Invariants

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

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

## 2.4 Example Message Exchange

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

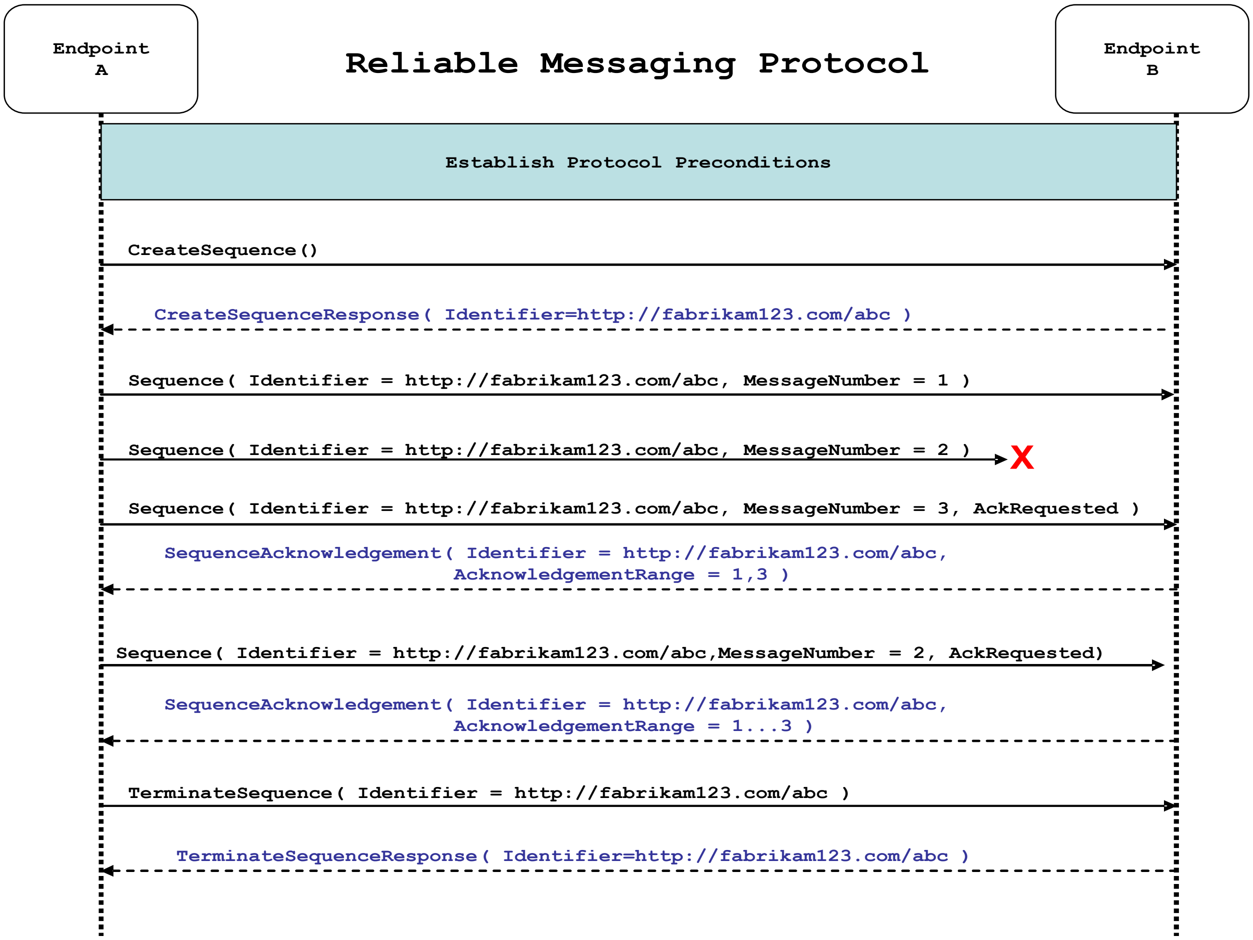
Endpoint A
Reliable Messaging Protocol
Endpoint B
Establish Protocol Preconditions
CreateSequence()
CreateSequenceResponse( Identifier=http://fabrikam123.com/abc )
Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 1 )
Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 2 )
X
Sequence( Identifier = http://fabrikam123.com/abc, MessageNumber = 3, AckRequested )
SequenceAcknowledgement( Identifier = http://fabrikam123.com/abc, AcknowledgementRange = 1,3 )
Sequence( Identifier = http://fabrikam123.com/abc,MessageNumber = 2, AckRequested)
SequenceAcknowledgement( Identifier = http://fabrikam123.com/abc, AcknowledgementRange = 1...3 )
TerminateSequence( Identifier = http://fabrikam123.com/abc )
TerminateSequenceResponse( Identifier=http://fabrikam123.com/abc )

*Figure 2: The WS-ReliableMessaging Protocol*

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

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

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

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

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

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

# 3 RM Protocol Elements

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.

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

If action IRIs are used by either the RM Source or the RM Destination, and one is not already defined as per the rules of the WS-Addressing specification, then the RM Source or the RM Destination MUST use an action IRI that consists of the WS-RM namespace URI concatenated with a '/', followed by the message element name. For example:

```
http://docs.oasis-open.org/ws-rx/wsrm/200604/SequenceAcknowledgement
```

## 3.1 Sequence Creation

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

### /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 `<wsrm: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 `<wsrm:SequenceAcknowledgement>` header blocks and faults related to the created Sequence are to be sent, unless otherwise noted in this specification (for example, see Section 3.2).

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 "none" IRI would make it impossible for the RM Destination to ever send Sequence Acknowledgements.

/wsrm:CreateSequence/wsrm:Expires

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

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

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

/wsrm:CreateSequence/wsrm:Offer

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

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

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

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

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

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

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

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

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

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

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

/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 different (extensible) types of information, based on a schema, to be passed.

### /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 `<wsrm:CreateSequenceResponse>` is sent in the body of a response message by an RM Destination in response to receipt of a `<wsrm:CreateSequence>` request message. It carries the `<wsrm:Identifier>` of the created Sequence and indicates that the RM Source may begin sending messages in the context of the identified Sequence.

The following exemplar defines the `<wsrm:CreateSequenceResponse>` syntax:

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

### /wsrm:CreateSequenceResponse

This element is sent in the body of the response message in response to a `<wsrm: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 [URI]) of 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. 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 `<wsrm: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:AcknowledgementInterval

This element, if present, specifies the duration after which the RM Destination will transmit an acknowledgement. If omitted, there is no implied value.

/wsrm:CreateSequenceResponse/wsrm:AcknowledgementInterval/@Milliseconds

The acknowledgement interval, specified in milliseconds.

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

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

/wsrm:CreateSequenceResponse/wsrm:IncompleteSequenceBehavior

This OPTIONAL element, if present, specifies the behavior that the RM Destination will exhibit upon the closure of an incomplete sequence.

A value of “DiscardEntireSequence” indicates that the entire sequence will be discarded by the RM Destination if the sequence is closed 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 will be discarded by the RM Destination 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 by the RM Destination.

/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 `<wsrm:CreateSequenceResponse>` is returned without a child `<wsrm:Accept>` in response to a `<wsrm:CreateSequence>` that did contain a child `<wsrm: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). The RM Source SHOULD send messages with `<wsrm:SequenceAcknowledgement>` header blocks related to the accepted Sequence to the referenced endpoint.

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

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

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

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

/wsrm:CreateSequenceResponse/{any}

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

/wsrm:CreateSequenceResponse/@{any}

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

## 3.2 Closing A Sequence

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

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

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

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

The following exemplar defines the CloseSequence syntax:

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

### /wsrm:CloseSequence

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

### /wsrm:CloseSequence/wsrm:Identifier

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

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

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

### /wsrm:CloseSequence/{any}

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

### /wsrm:CloseSequence@{any}

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

A `<wsrm:CloseSequenceResponse>` is sent in the body of a response message by an RM Destination in response to receipt of a `<wsrm:CloseSequence>` request message. It indicates that the RM Destination has closed the Sequence.

The following exemplar defines the `<wsrm:CloseSequenceResponse>` syntax:

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

### /wsrm:CloseSequenceResponse

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

### /wsrm:CloseSequenceResponse/wsrm:Identifier

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

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

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

### /wsrm:CloseSequenceResponse/{any}

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

### /wsrm:CloseSequenceResponse@{any}

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

## 3.3 Sequence Termination

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

The following exemplar defines the TerminateSequence syntax:

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

/wsrm:TerminateSequence

This element is 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.

/wsrm:TerminateSequence/wsrm:Identifier

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

/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 `<wsrm:TerminateSequenceResponse>` is sent in the body of a response message by an RM Destination in response to receipt of a `<wsrm:TerminateSequence>` request message. It indicates that the RM Destination has terminated the Sequence.

The following exemplar defines the `<wsrm:TerminateSequenceResponse>` syntax:

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

/wsrm:TerminateSequenceResponse

This element is sent in the body of a response message by an RM Destination in response to receipt of a `<wsrm:TerminateSequence>` request message. It indicates that the RM Destination has terminated the sequence. The RM Destination MUST NOT send this element as a header block.

/wsrm:TerminateSequenceResponse/wsrm:Identifier

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

/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 <wsrm:TerminateSequence> message an RM Destination MUST respond with a corresponding <wsrm:TerminateSequenceResponse> message or generate a fault.

## 3.4 Sequences

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

The RM Source MUST NOT include more than one <wsrm: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 <wsrm: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 <wsrm:Sequence> SOAP header block is bound) to the <wsrm:Sequence> header block element.

/wsrm:Sequence/wsrm:Identifier

An RM Source that includes a <wsrm: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 [URI]) that uniquely identifies the Sequence.

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

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

/wsrm:Sequence/wsrm:MessageNumber

The RM Source MUST include this element within any Sequence headers it creates. This element is of type wsrm:MessageNumberType. It represents the ordinal position of the message within a Sequence. Sequence message numbers start at 1 and monotonically increase by 1 throughout the Sequence. If the message number exceeds the internal limitations of an RM Source or RM Destination or reaches the maximum value of 9,223,372,036,854,775,807 the RM Source or Destination MUST generate a MessageNumberRollover fault.

/wsrm:Sequence/{any}

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

/wsrm:Sequence/@{any}

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

The following example illustrates a Sequence header block.

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

## 3.5 Request Acknowledgement

The purpose of the `<wsrm:AckRequested>` header block is to signal to the RM Destination that the RM Source is requesting that a `<wsrm:SequenceAcknowledgement>` be sent.

The RM Source MAY request an acknowledgement message from the RM Destination at any time by including an `<wsrm:AckRequested>` header block in any message targeted to the RM Destination. An RM Destination that receives a message that contains an `<wsrm:AckRequested>` header block MUST send a message containing a `<wsrm:SequenceAcknowledgement>` header block to the wsrm:AcksTo endpoint reference (see Section 3.1). If a non-mustUnderstand fault occurs when processing an RM header that was piggy-backed on another message, a fault MUST be generated, but the processing of the original message MUST NOT be affected. It is RECOMMENDED that the RM Destination return a <wsrm:AcknowledgementRange> or <wsrm:None> element instead of a <wsrm:Nack> element (see Section 3.6).

The following exemplar defines its syntax:

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

    ...
</wsrm:AckRequested>
```

/wsrm:AckRequested

This element requests an acknowledgement for the identified Sequence.

/wsrm:AckRequested/wsrm:Identifier

An RM Source that includes a `<wsrm:AckRequested>` 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 [URI]), that uniquely identifies the Sequence to which the request applies.

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

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

/wsrm:AckRequested/{any}

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

/wsrm:AckRequested/@{any}

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

## 3.6 Sequence Acknowledgement

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

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

During creation of a Sequence the RM Source MAY specify the WS-Addressing anonymous IRI as the address of the `<wsrm:AcksTo>` EPR for that Sequence. When the RM Source specifies the WS-Addressing anonymous IRI as the address of the `<wsrm:AcksTo>` EPR, the RM Destination MUST transmit any `<wsrm:SequenceAcknowledgement>` headers for the created Sequence in a SOAP envelope to be transmitted on the protocol binding-specific channel. Such a channel is provided by the context of a received message containing a SOAP envelope that contains a `<wsrm:Sequence>` header block and/or a `<wsrm:AckRequested>` header block for that same Sequence identifier.

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> + ]

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

The following describes the content model of the `<wsrm:SequenceAcknowledgement>` header block.

/wsrm:SequenceAcknowledgement

This element contains the Sequence acknowledgement information.

/wsrm:SequenceAcknowledgement/wsrm:Identifier

An RM Destination that includes a `<wsrm:SequenceAcknowledgement>` header block in a SOAP envelope MUST include this element in that header block. The RM Destination MUST set the value of this element to the absolute URI (conformant with RFC3986 [URI]) that uniquely identifies the Sequence. The

RM Destination MUST NOT include multiple `<wsrm:SequenceAcknowledgement>` header blocks that share the same value for `<wsrm:Identifier>` within the same SOAP envelope.

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

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

### /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange

The RM Destination MAY include one or more instances of this element within a `<wsrm:SequenceAcknowledgement>` header block. It contains a range of Sequence MessageNumbers successfully received by the RM Destination. The ranges SHOULD NOT overlap. The RM Destination MUST NOT include this element if a sibling `<wsrm:Nack>` or `<wsrm:None>` element is also present as a child of `<wsrm:SequenceAcknowledgement>`.

### /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper

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

### /wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower

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

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

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

### /wsrm:SequenceAcknowledgement/wsrm:Final

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

### /wsrm:SequenceAcknowledgement/wsrm:Nack

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

### /wsrm:SequenceAcknowledgement/wsrm:None

The RM Destination MUST include this element within a <wsrm:SequenceAcknowledgement> header block if the RM Destination has not received any messages for the specified Sequence. The RM Destination MUST NOT include this element if a sibling <wsrm:AcknowledgementRange> or <wsrm:Nack> element is also present as a child of the <wsrm:SequenceAcknowledgement>.

#### /wsrm:SequenceAcknowledgement/{any}

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

#### /wsrm:SequenceAcknowledgement/@{any}

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

The following examples illustrate <wsrm:SequenceAcknowledgement> elements:

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

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

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

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

- Message number 3 in a Sequence has not been received by the RM Destination.

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

# 4 Faults

The faults defined in this section fall into one of two categories; those faults that are the result of messages or operations within a specific Sequence and those faults that are not. By their nature the CreateSequenceRefused, UnknownSequence, and WSRMRequired faults cannot be correlated with a Sequence. All other faults defined in this section relate to the processing of WS-RM protocol messages or messages containing WS-RM header blocks targeted at a specific Sequence and are collectively referred to as "Sequence faults".

Faults for the CreateSequence message exchange are treated as defined in WS-Addressing. CreateSequenceRefused is a possible fault reply for this operation. UnknownSequence is a fault generated by endpoints when messages carrying RM header blocks targeted at unrecognized or terminated Sequences are detected. All other faults in this section relate to the processing of RM header blocks targeted at known Sequences and are collectively referred to as Sequence faults. Entities that generate Sequence faults SHOULD send those faults to the same [destination] as `<wsrm:SequenceAcknowledgement>` messages. These faults are correlated using the Sequence identifier carried in the detail.

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/200604/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/200604/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </S:Header>
 <S:Body>
  <S:Fault>
   <S:Code>
     <S:Value> [Code] </S:Value>
     <S:Subcode>
      <S:Value> [Subcode] </S:Value>
     </S:Subcode>
   </S:Code>
   <S:Reason>
     <S:Text xml:lang="en"> [Reason] </S:Text>
   </S:Reason>
   <S:Detail>
     [Detail]
           ...
   </S:Detail>
  </S:Fault>
 </S:Body>
</S:Envelope>
```

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

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

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

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

## 4.1 SequenceFault Element

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

The following exemplar defines its syntax:

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

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

/wsrm:SequenceFault

This is the element containing Sequence information for WS-ReliableMessaging

/wsrm:SequenceFault/wsrm:FaultCode

WS-ReliableMessaging nodes that generate a <wsrm:SequenceFault> MUST set the value of this element to a qualified name from the set of fault [Subcodes] defined below.

/wsrm:SequenceFault/wsrm:Detail

This OPTIONAL element is intended for carrying application specific error information related to the fault being described.

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

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

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

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

/wsrm:SequenceFault/{any}

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

/wsrm:SequenceFault/@{any}

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

## 4.2 Sequence Terminated

This fault is generated by either the RM Source or the RM Destination to indicate that it has either encountered an unrecoverable condition, or has detected a violation of the protocol and as a consequence, has chosen to terminate the Sequence. The endpoint that generates this fault should make every reasonable effort to notify the corresponding endpoint of this decision.

Receipt of SequenceTerminated by either the RM Destination or the RM Source shall terminate the Sequence if it is not otherwise terminated.

Properties:

[Code] Sender or Receiver

[Subcode] wsrm:SequenceTerminated

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

[Detail]

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

## 4.3 Unknown Sequence

This fault is generated by either the RM Source or the RM Destination in response to a message containing an unknown or terminated Sequence identifier. Receipt of UnknownSequence by either the RM Destination or the RM Source shall terminate the Sequence if it is not otherwise terminated.

Properties:

[Code] Sender

[Subcode] wsrm:UnknownSequence

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

[Detail]

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

## 4.4 Invalid Acknowledgement

This fault is generated by the RM Source in response to a `<wsrm:SequenceAcknowledgement>` that violates the cumulative acknowledgement invariant. An example of such a violation would be a SequenceAcknowledgement covering messages that have not been sent.

[Code] Sender

[Subcode] wsrm:InvalidAcknowledgement

[Reason] The SequenceAcknowledgement violates the cumulative acknowledgement invariant.

[Detail]

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

## 4.5 Message Number Rollover

This fault is generated to indicate that message numbers for a Sequence have been exhausted.

Properties:

[Code] Sender

[Subcode] wsrm:MessageNumberRollover

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

[Detail]

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

## 4.6 Create Sequence Refused

This fault is generated in response to a create Sequence request that cannot be satisfied.

Properties:

[Code] Sender

[Subcode] wsrm:CreateSequenceRefused

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

[Detail]

```
xs:any
```

## 4.7 Sequence Closed

This fault is generated by an RM Destination to indicate that the specified Sequence has been closed. This fault MUST be generated when an RM Destination is asked to receive a message for a Sequence that is closed or when an RM Destination is asked to close a Sequence that is already closed.

Properties:

[Code] Sender

[Subcode] wsrm:SequenceClosed

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

[Detail]

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

## 4.8 WSRM Required

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

Properties:

[Code] Sender

[Subcode] wsrm:WSRMRequired

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

[Detail]

```
xs:any
```

# 5 Security Considerations

It is strongly recommended that the communication between services be secured using the mechanisms described in WS-Security. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the `<wsrm:Sequence>` header needs to be signed with the body in order to "bind" the two together. The `<wsrm:SequenceAcknowledgement>` header may be signed independently because a reply independent of the message is not a security concern.

Because Sequences are expected to exchange a number of messages, it is recommended that a security context be established using the mechanisms described in WS-Trust[Trust] and WS-SecureConversation [SecureConversation]. If a Sequence is bound to a specific destination, then the security context needs to be established or shared with the destination servicing the Sequence. While the context can be established at any time, it is critical that the messages establishing the Sequence be secured even if they precede security context establishment. However, it is recommended that the security context be established first. Security contexts are independent of reliable messaging Sequences. Consequently, security contexts can come and go independent of the lifetime of the Sequence. In fact, it is recommended that the lifetime of a security context be less than the lifetime of the Sequence unless the Sequence is very short-lived.

It is common for message Sequences to exchange a number of messages (or a large amount of data). As a result, the usage profile of a Sequence is such that it is susceptible to key attacks. For this reason it is strongly recommended that the keys be changed frequently. This "re-keying" can be effected a number of ways. The following list outlines four common techniques:

- Closing and re-establishing a security context
- Exchanging new secrets between the parties
- Using a derived key sequence and switch "generations"
- Attaching a nonce to each message and using it in a derived key function with the shared secret

The security context may be re-established using the mechanisms described in WS-Trust and WS-SecureConversation. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret should not be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

There is a core tension between security and reliable messaging that can be problematic if not considered in implementations. That is, one aspect of security is to prevent message replay and the core tenet of reliable messaging is to replay messages until they are acknowledged. Consequently, if the security sub-system processes a message but a failure occurs before the reliable messaging sub-system records the message (or the message is considered "processed"), 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 rare condition.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

- **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security.
- **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.

- **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy and WS-SecurityPolicy).
- **Authentication** – Authentication is established using the mechanisms described in WS-Security and WS-Trust. Each message is authenticated using the mechanisms described in WS-Security.
- **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- **Availability** – All reliable messaging services are subject to a variety of availability attacks. Replay detection is a common attack and it is recommended that this be addressed by the mechanisms described in WS-Security. (Note that because of legitimate message replays, detection should include a differentiator besides message id such as a timestamp). Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal state is saved prior to any authenticating Sequences.

# 6 References

## 6.1 Normative

**[KEYWORDS]**

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

**[SOAP 1.1]**

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

**[SOAP 1.2]**

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

**[URI]**

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

**[XML]**

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

**[XML-ns]**

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

**[XML-Schema Part1]**

W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001.

**[XML-Schema Part2]**

W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001.

**[WSDL 1.1]**

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

**[WS-Addressing]**

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

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

## 6.2 Non-Normative

**[RDDL 2.0]**

Johnathan Borden, Tim Bray, eds. “Resource Directory Description Language (RDDL) 2.0,” January 2004

**[WS-Policy]**

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

**[WS-PolicyAttachment]**

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

**[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.

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.

**[RTTM]**

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

**[SecurityPolicy]**

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

**[SecureConversation]**

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

**[Trust]**

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

# 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/200604/wsrm-1.1-schema-200604.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 â€œAS
ISâ€� 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/200604"
targetNamespace="http://docs.oasis-open.org/ws-rx/wsrm/200604"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2005/08/addressing"
schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
  <!-- Protocol Elements -->
  <xs:complexType name="SequenceType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:element name="MessageNumber" type="wsrm:MessageNumberType"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="Sequence" type="wsrm:SequenceType"/>
  <xs:element name="SequenceAcknowledgement">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="wsrm:Identifier"/>
        <xs:choice>
          <xs:sequence>
            <xs:choice>
              <xs:element name="AcknowledgementRange" maxOccurs="unbounded">
                <xs:complexType>
```

```
                  <xs:sequence/>
                  <xs:attribute name="Upper" type="xs:unsignedLong"
use="required"/>
                  <xs:attribute name="Lower" type="xs:unsignedLong"
use="required"/>
                  <xs:anyAttribute namespace="##other" processContents="lax"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="None" minOccurs="0">
                <xs:complexType>
                  <xs:sequence/>
                </xs:complexType>
              </xs:element>
            </xs:choice>
            <xs:element name="Final" minOccurs="0">
              <xs:complexType>
                <xs:sequence/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:element name="Nack" type="xs:unsignedLong"
maxOccurs="unbounded"/>
        </xs:choice>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="AckRequestedType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="AckRequested" type="wsrm:AckRequestedType"/>
  <xs:element name="Identifier">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          This type is for elements whose [children] is an anyURI and can have
arbitrary attributes.
        </xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
        <xs:extension base="xs:anyURI">
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="MessageNumberType">
    <xs:restriction base="xs:unsignedLong">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="9223372036854775807"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- Fault Container and Codes -->
  <xs:simpleType name="FaultCodes">
    <xs:restriction base="xs:QName">
      <xs:enumeration value="wsrm:SequenceTerminated"/>
      <xs:enumeration value="wsrm:UnknownSequence"/>
      <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
      <xs:enumeration value="wsrm:MessageNumberRollover"/>
      <xs:enumeration value="wsrm:CreateSequenceRefused"/>
      <xs:enumeration value="wsrm:SequenceClosed"/>
      <xs:enumeration value="wsrm:WSRMRequired"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="SequenceFaultType">
    <xs:sequence>
      <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
      <xs:element name="Detail" type="wsrm:DetailType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="DetailType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
  <xs:element name="CreateSequence" type="wsrm:CreateSequenceType"/>
  <xs:element name="CreateSequenceResponse"
type="wsrm:CreateSequenceResponseType"/>
  <xs:element name="CloseSequence" type="wsrm:CloseSequenceType"/>
  <xs:element name="CloseSequenceResponse"
type="wsrm:CloseSequenceResponseType"/>
  <xs:element name="TerminateSequence" type="wsrm:TerminateSequenceType"/>
  <xs:element name="TerminateSequenceResponse"
type="wsrm:TerminateSequenceResponseType"/>
  <xs:complexType name="CreateSequenceType">
    <xs:sequence>
      <xs:element ref="wsrm:AcksTo"/>
      <xs:element ref="wsrm:Expires" minOccurs="0"/>
      <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            It is the authors intent that this extensibility be used to
transfer a Security Token Reference as defined in WS-Security.
          </xs:documentation>
        </xs:annotation>
      </xs:any>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="CreateSequenceResponseType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:element ref="wsrm:Expires" minOccurs="0"/>
      <xs:element ref="wsrm:AcknowledgementInterval" minOccurs="0"/>
      <xs:element name=”IncompleteSequenceBehaviour”
type=”wsrm:IncompleteSequenceBehaviorType” minOccurs=”0”/>
      <xs:element name="Accept" type="wsrm:AcceptType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="CloseSequenceType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="CloseSequenceResponseType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="TerminateSequenceType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="TerminateSequenceResponseType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="AcksTo"
```

```
 type="wsa:EndpointReferenceType"/>
  <xs:complexType name="OfferType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:element ref="wsrm:Expires" minOccurs="0"/>
      <xs:element name="EndpointReference" type="wsa:EndpointReferenceType"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="AcceptType">
    <xs:sequence>
      <xs:element ref="wsrm:AcksTo"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:element name="Expires">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:duration">
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="AcknowledgementInterval">
    <xs:complexType>
      <xs:sequence/>
      <xs:attribute name="Milliseconds" type="xs:unsignedLong"
use="required"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="IncompleteSequenceBehaviorType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="DiscardEntireSequence"/>
      <xs:enumeration value="DiscardFollowingFirstGap"/>
      <xs:enumeration value="NoDiscard"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

# B. Message Examples

## B.1 Create Sequence

### Create Sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsrm="http://docs.oasis-open.org/ws-rx/wsrm/200604"
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/200604/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 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/200604"
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/200604/CreateSequenceResponse
    </wsa:Action>
  </S:Header>
  <S:Body>
    <wsrm:CreateSequenceResponse>
      <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
    </wsrm:CreateSequenceResponse>
  </S:Body>
</S:Envelope>
```

## B.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 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/200604"
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 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/200604"
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 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/200604"
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>
 </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>3</wsrm:MessageNumber>
 </wsrm:Sequence>
 <wsrm:AckRequested>
  <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
 </wsrm:AckRequested>
</S:Header>
<S:Body>
 <!-- Some Application Data -->
</S:Body>
</S:Envelope>
```

## B.3 First Acknowledgement

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

```
<?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/200604"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <S:Header>
  <wsa:MessageID>
   http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546810
  </wsa:MessageID>
  <wsa:To>http://Business456.com/serviceA/789</wsa:To>
  <wsa:From>
   <wsa:Address>http://example.com/serviceB/123</wsa:Address>
  </wsa:From>
  <wsa:Action>
    http://docs.oasis-open.org/ws-rx/wsrm/200604/SequenceAcknowledgement
  </wsa:Action>
  <wsrm:SequenceAcknowledgement>
   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
   <wsrm:AcknowledgementRange Upper="1" Lower="1"/>
   <wsrm:AcknowledgementRange Upper="3" Lower="3"/>
  </wsrm:SequenceAcknowledgement>
 </S:Header>
 <S:Body/>
</S:Envelope>
```

## B.4 Retransmission

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

```
<?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/200604"
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>
  <wsrm:AckRequested>
   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
  </wsrm:AckRequested>
 </S:Header>
 <S:Body>
  <!-- Some Application Data -->
 </S:Body>
</S:Envelope>
```

## B.5 Termination

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

```
<?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/200604"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <S:Header>
  <wsa:MessageID>
   http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546811
  </wsa:MessageID>
  <wsa:To>http://Business456.com/serviceA/789</wsa:To>
  <wsa:From>
   <wsa:Address>http://example.com/serviceB/123</wsa:Address>
  </wsa:From>
  <wsa:Action>
    http://docs.oasis-open.org/ws-rx/wsrm/200604/SequenceAcknowledgement
  </wsa:Action>
  <wsrm:SequenceAcknowledgement>
   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
   <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
  </wsrm:SequenceAcknowledgement>
 </S:Header>
 <S:Body/>
</S:Envelope>
```

**Terminate Sequence**

```
<?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/200604"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
  </wsa:MessageID>
  <wsa:To>http://example.com/serviceB/123</wsa:To>
  <wsa:Action>
    http://docs.oasis-open.org/ws-rx/wsrm/200604/TerminateSequence
  </wsa:Action>
  <wsa:From>
   <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:From>
 </S:Header>
 <S:Body>
  <wsrm:TerminateSequence>
```

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

### Terminate Sequence Response

```
<?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/200604"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546813
  </wsa:MessageID>
  <wsa:To>http://example.com/serviceA/789</wsa:To>
  <wsa:Action>
    http://docs.oasis-open.org/ws-rx/wsrm/200604/TerminateSequenceResponse
  </wsa:Action>
  <wsa:RelatesTo>
    http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546812
  </wsa:RelatesTo>
  <wsa:From>
   <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:From>
 </S:Header>
 <S:Body>
  <wsrm:TerminateSequenceResponse>
   <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
  </wsrm:TerminateSequenceResponse>
 </S:Body>
</S:Envelope>
```

# C. WSDL

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

http://docs.oasis-open.org/ws-rx/wsrm/200604/wsdl/wsrm-1.1-wsdl-200604.wsdl

The following non-normative copy is provided for reference.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
OASIS takes no position regarding the validity or scope of any intellectual
property or other rights that might be claimed to pertain to the
implementation or use of the technology described in this document or the
extent to which any license under such rights might or might not be available;
neither does it represent that it has made any effort to identify any such
rights. Information on OASIS's procedures with respect to rights in OASIS
specifications can be found at the OASIS website. Copies of claims of rights
made available for publication and any assurances of licenses to be made
available, or the result of an attempt made to obtain a general license or
permission for the use of such proprietary rights by implementors or users of
this specification, can be obtained from the OASIS Executive Director.
OASIS invites any interested party to bring to its attention any copyrights,
patents or patent applications, or other proprietary rights which may cover
technology that may be required to implement this specification. Please
address the information to the OASIS Executive Director.
Copyright (c) OASIS Open 2002-2006. All Rights Reserved.
This document and translations of it may be copied and furnished to others,
and derivative works that comment on or otherwise explain it or assist in its
implementation may be prepared, copied, published and distributed, in whole or
in part, without restriction of any kind, provided that the above copyright
notice and this paragraph are included on all such copies and derivative
works. However, this document itself does not be modified in any way, such as
by removing the copyright notice or references to OASIS, except as needed for
the purpose of developing OASIS specifications, in which case the procedures
for copyrights defined in the OASIS Intellectual Property Rights document must
be followed, or as required to translate it into languages other than English.
The limited permissions granted above are perpetual and will not be revoked by
OASIS or its successors or assigns.
This document and the information contained herein is provided on an â€œAS
ISâ€� 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/200604" xmlns:tns="http://docs.oasis-open.org/ws-
rx/wsrm/200604/wsdl" targetNamespace="http://docs.oasis-open.org/ws-
rx/wsrm/200604/wsdl">

  <wsdl:types>
    <xs:schema>
      <xs:import namespace="http://docs.oasis-open.org/ws-rx/wsrm/200604"
schemaLocation="http://docs.oasis-open.org/ws-rx/wsrm/200604/wsrm-1.1-schema-
200604.xsd"/>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="CreateSequence">
    <wsdl:part name="create" element="rm:CreateSequence"/>
  </wsdl:message>
  <wsdl:message name="CreateSequenceResponse">
    <wsdl:part name="createResponse" element="rm:CreateSequenceResponse"/>
  </wsdl:message>
  <wsdl:message name="CloseSequence">
    <wsdl:part name="close" element="rm:CloseSequence"/>
  </wsdl:message>
  <wsdl:message name="CloseSequenceResponse">
    <wsdl:part name="closeResponse" element="rm:CloseSequenceResponse"/>
  </wsdl:message>
  <wsdl:message name="TerminateSequence">
    <wsdl:part name="terminate" element="rm:TerminateSequence"/>
  </wsdl:message>
  <wsdl:message name="TerminateSequenceResponse">
    <wsdl:part name="terminateResponse"
element="rm:TerminateSequenceResponse"/>
  </wsdl:message>

  <wsdl:portType name="SequenceAbstractPortType">
    <wsdl:operation name="CreateSequence">
      <wsdl:input message="tns:CreateSequence" wsa:Action="http://docs.oasis-
open.org/ws-rx/wsrm/200604/CreateSequence"/>
      <wsdl:output message="tns:CreateSequenceResponse"
wsa:Action="http://docs.oasis-open.org/ws-
rx/wsrm/200604/CreateSequenceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="CloseSequence">
      <wsdl:input message="tns:CloseSequence" wsa:Action="http://docs.oasis-
open.org/ws-rx/wsrm/200604/CloseSequence"/>
      <wsdl:output message="tns:CloseSequenceResponse"
wsa:Action="http://docs.oasis-open.org/ws-
rx/wsrm/200604/CloseSequenceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="TerminateSequence">
      <wsdl:input message="tns:TerminateSequence"
wsa:Action="http://docs.oasis-open.org/ws-rx/wsrm/200604/TerminateSequence"/>
      <wsdl:output message="tns:TerminateSequenceResponse"
wsa:Action="http://docs.oasis-open.org/ws-
rx/wsrm/200604/TerminateSequenceResponse"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

# D. State Tables

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

Each cell in the tables in this appendix uses the following convention:

| **Legend** |
|---|
| *action to take*<br>next state |

Table 2 RM Source State Transition Table

| **Events** | **States** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **None** | **Connecting** | **Connected** | **Rollover** | **Closing** | **Closed** | **Terminating** | **Terminated** |
| **Create Sequence** | *Transmit Create Sequence*<br>Connecting | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Create Sequence Response** | N/A | No action<br>Connected | N/A | N/A | N/A | N/A | N/A | N/A |
| **Create Sequence Refused Fault** | N/A | No action<br>Terminated | N/A | N/A | N/A | N/A | N/A | N/A |
| **New Message** | N/A | N/A | *Transmit message*<br>Connected | *no action*<br>Rollover | *No action*<br>Closing | N/A | N/A | N/A |
| **Retransmit of unack message** | N/A | N/A | *Transmit message*<br>Connected | *Transmit message*<br>Rollover | *Trasmit message?*<br>Closing | *No action*<br>Closed | N/A | N/A |
| **SeqAck (non-final)** | N/A | N/A | *Process Ack ranges*<br>Connected | *Process Ack ranges*<br>Rollover | *Process Ack ranges*<br>Closing | *Process Ack ranages*<br>Closed | *Process Ack ranages*<br>Terminating | *Transmit Unknown Sequence Fault*<br>Terminated |
| **Nack** | N/A | N/A | *Transmit message(s)*<br>Connected | *Transmit message(s)*<br>Rollover | *Transmit message(s)*<br>Closing | *No action*<br>Closed | *No action*<br>Terminating | *Transmit Unknown Sequence fault*<br>Terminated |
| **Reached max msg number** | N/A | N/A | *No action*<br>Rollover | *No action*<br>Rollover | N/A | N/A | N/A | N/A |

| **Events** | **States** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **None** | **Connecting** | **Connected** | **Rollover** | **Closing** | **Closed** | **Terminating** | **Terminated** |
| **Message Number Rollover Fault** | N/A | N/A | *No action*<br>Rollover | *No action*<br>Rollover | *No action*<br>Closing | *No action*<br>Closed | *No action*<br>Terminating | *Transmit Unknowne Sequence Fault*<br>Terminated |
| **Close Sequence** | N/A | N/A | *Transmit Close Sequence*<br>Closing | *Transmit Close Sequence*<br>Closing | *Transmit Close Sequence*<br>Closing | *No action*<br>Closed | *No action*<br>Terminating | N/A |
| **Close Sequence Response** | N/A | N/A | N/A | N/A | *No action*<br>Closed | *No action*<br>Closed | *No action*<br>Terminating | *Transmit Unknown Sequence Fault*<br>Terminated |
| **SeqAck (final)** | N/A | N/A | *Process Ack/Nack ranges*<br>Closed | *Process Ack/Nack ranges*<br>Closed | *Process Ack/Nack ranges*<br>Closed | *Process Ack/Nack ranges*<br>Closed | *Process Ack/Nack ranges*<br>Terminating | *Transmit Unknown Sequence fault*<br>Terminated |
| **Sequence Closed Fault** | N/A | N/A | *No action*<br>Closed | *No action*<br>Closed | *No action*<br>Closed | *No action*<br>Closed | *No action*<br>Terminating | *Transmit Unknown Sequence Fault*<br>Terminated |
| **Unknown Sequence Fault** | N/A | N/A | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated |
| **Sequence Terminated Fault** | N/A | N/A | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No action*<br>Terminated | *No Action*<br>Terminated |
| **Terminate Sequence** | N/A | N/A | *Transmit Terminate Sequence*<br>Terminating | *Transmit Terminate Sequence*<br>Terminating | *Transmit Terminate Sequence*<br>Terminating | *Transmit Terminate Sequence*<br>Terminating | *Transmit Terminate Sequence*<br>Terminating | N/A |
| **Terminate Sequence Response** | N/A | N/A | N/A | N/A | N/A | N/A | *No action*<br>Terminated | *No action*<br>Terminated |
| **Elapse Expires duration** | N/A | N/A | *Send SequenceTerminated Fault*<br>Terminated | *Send SequenceTerminated Fault*<br>Terminated | *Send SequenceTerminated Fault*<br>Terminated | *Send SequenceTerminated Fault*<br>Terminated | *Send SequenceTerminated Fault*<br>Terminated | N/A |

In Table 2 above, the rows consists of events that occur at the RM Source throughout the lifetime of an RM Sequence and the columns consists of various RM Source states. Each cell in the table above lists the action that the RM Source takes on occurrence of a particular event and the next state that it transitions.

Table 3 RM Destination State Transition Table

| **Events** | **States** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **None** | **Connecting** | **Connected** | | | **Closed** | **Terminated** |
| **Creation request not satisfied** | N/A | *Send Create Sequence Refused Fault*<br>Terminated | N/A | | | N/A | |
| | | | | | | | |
| **Message (with message number within range)** | N/A | N/A | *No action*<br>Connected | | | *Send Sequence Closed Fault (with SeqAck+Final )*<br>Closed | *Send Unknown Seq Fault*<br>Terminated |
| | | | | | | | |
| **Ack requested** | N/A | N/A | *Send SequenceAck*<br>Connected | | | *Send SeqAck+Final*<br>Closed | *Send Unknown Seq Fault*<br>Terminated |
| **Message (with message number outside of range)** | N/A | N/A | *Send Message Number Rollover Fault*<br>Connected | | | N/A | N/A |
| | | | | | | | |
| **Close Sequence** | N/A | N/A | *Send CloseSequenceResponse with SequenceAck (Final)*<br>Closed | | | *Send Close Sequence Response with SeqAck+Final*<br>Closed | *Send Unknown Sequence Fault*<br>Terminated |
| **Close Sequence itself** | N/A | N/A | Closed | | | *Send Sequence Closed Fault*<br>Closed | N/A |
| **Terminate Sequence** | N/A | N/A | *Send Terminate Sequence Response*<br>Terminated | | | *Send Terminate Sequence Response*<br>Terminated | *Send Unknown Sequence Fault*<br>Terminated |

| Events | States | | | | | | |
|---|---|---|---|---|---|---|---|
| | **None** | **Connecting** | **Connected** | | | **Closed** | **Terminated** |
| **Unknown Sequence Fault** | N/A | N/A | *No action*<br>Terminated | | | *No action*<br>Terminated | *No action*<br>Terminated |
| **Sequence Terminated Fault** | N/A | N/A | *No action*<br>Terminated | | | *No action*<br>Terminated | *No action*<br>Terminated |
| | | | | | | | |
| **Elapse Expires duration** | N/A | N/A | *Send Sequence Terminated Fault*<br>Terminated | | | *Send Sequence Terminated Fault*<br>Terminated | N/A |

In Table 3 above, the rows consists of events that occur at the RM Destination throughout the lifetime of an RM Sequence and the columns consists of various RM Destination states. Each cell in the table above lists the action that the RM Destination takes on occurrence of a particular event and the next state that it transitions.

# 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, IBM (Editor), Tom Freund, IBM, Mary Ann Hondo, IBM, John Ibbotson, IBM, Lei Jin, BEA, Chris Kaler, Microsoft, David Langworthy, Microsoft (Editor), 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, 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, 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:

*TBD*

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

# 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.