Web Services Reliable Messaging (WS-Reliable Messaging)

Working Draft 05, October 20th 2005

Document identifier:

WS-ReliableMessaging-1.1-draft-03.doc

Location:

TBD

Editors:

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

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

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.

Composable Architecture:

By using the SOAP [SOAP] and WSDL [WSDL] 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:

TBD

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 message delivery. It defines a messaging protocol to identify, track, and manage the reliable delivery of messages between exactly two parties, 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-Policy, and other Web services specifications. Combined, these allow for a broad range of reliable, secure messaging options.

1.1 Goals and Requirements

1.1.1 Requirements

1.2 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. Additional children elements and/or attributes MAY be added at the indicated extension points but 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 Namespace) are used to indicate the namespace of the element being defined.
-

1.3 Namespace

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

```
http://docs.oasis-open.org/wsrm/200510/
```

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

The following namespaces are used in this document:

Table 1

Prefix	Namespace
S	http://www.w3.org/2003/05/soap-envelope
S11	http://schemas.xmlsoap.org/soap/envelope/
wsrm	http://docs.oasis-open.org/wsrm/200510/
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

The normative schema for WS-Reliable Messaging can be found at:

```
http://docs.oasis-open.org/wsrm/200510/wsrm.xsd
```

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

If an action URI is used, and one is not already defined per the rules of the WS-Addressing specification [WS-Addressing], then the action URI MUST consist of the reliable messaging namespace URI concatenated with the element name. For example:

```
http://docs.oasis-open.org/wsrm/200510/SequenceAcknowledgement
```

1.4 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 SectionNamespace) 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 (RM) 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 extreem circumstances, accurately determine the disposition of each message transmitted as perceived by the RM Destination, so as to resolve any in-doubt status.

In addition, The protocol allows the RM Source and RM Destination to provide their respective Application Source and Application Destination a guarantee that a message that is sent by an Application Source will be delivered to the Application Destination.

This guarantee is specified as a delivery assurance. It is the responsibility of the RM Source and RM Destination to fulfill the delivery assurances on behalf of their respective Application counterparts, or raise an error. The protocol defined here allows endpoints to meet this guarantee for the delivery assurances defined below. However, the means by which these delivery assurances are manifested by either the RM Source or RM Destination roles is an implementation concern, and is out of scope of this specification.

Note that the underlying protocol defined in this specification remains the same regardless of the delivery assurance.

Persistence considerations related to an endpoint's ability to satisfy the delivery assurances defined below are the responsibility of the implementation and do not affect the wire protocol. As such, they are out of scope of this specification.

There are four basic delivery assurances that endpoints can provide:

AtMostOnce Messages will be delivered at most once without duplication or an error will be raised on at least one endpoint. It is possible that some messages in a sequence may not be delivered.

AtLeastOnce Every message sent will be delivered or an error will be raised on at least one endpoint. Some messages may be delivered more than once.

ExactlyOnce Every message sent will be delivered without duplication or an error will be raised on at least one endpoint. This delivery assurance is the logical "and" of the two prior delivery assurances.

InOrder Messages will be delivered in the order that they were sent. This delivery assurance may be combined with any of the above delivery assurances. It requires that the messages within a Sequence will be delivered in an order so that the message numbers are monotonically increasing. Note that this assurance says nothing about duplications or omissions. Note also that it is only applicable to messages in the same Sequence. Cross Sequence ordering of messages is not in the scope of this specification.

Figure 1 below illustrates the entities and events in a simple reliable message exchange. First, the Application Source Sends a message for reliable delivery. The Reliable Messaging (RM) 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.

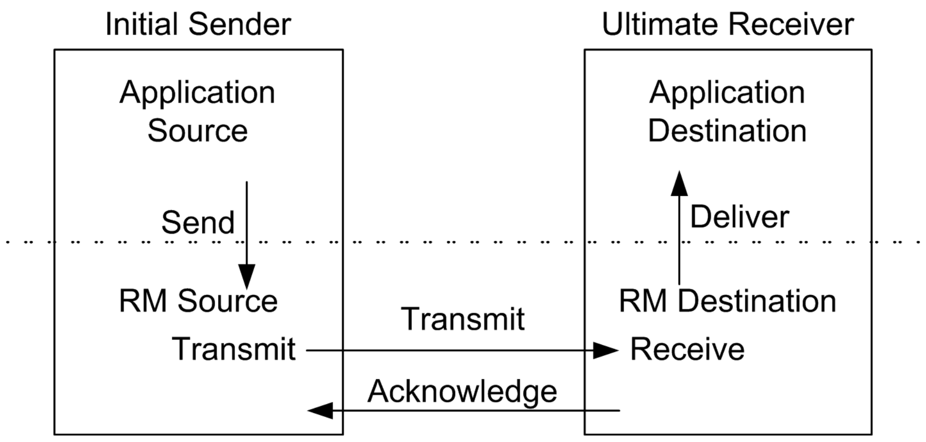


Figure 1: Reliable Messaging Model

2.1 Glossary

The following definitions are used throughout this specification:

Endpoint: A referencable entity, processor, or resource where Web service messages are originated or targeted.

Application Source: The endpoint that Sends a message.

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

Delivery Assurance: The guarantee that the messaging infrastructure provides on the delivery of a message.

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

RM Source: The endpoint that transmits the message.

RM Destination: The endpoint that receives the message.

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

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

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

Receive: The act of reading a message from a network connection.

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

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:

- The RM Source MUST have an endpoint reference that uniquely identifies the RM Destination endpoint; correlations across messages addressed to the unique endpoint MUST be meaningful.
- 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.

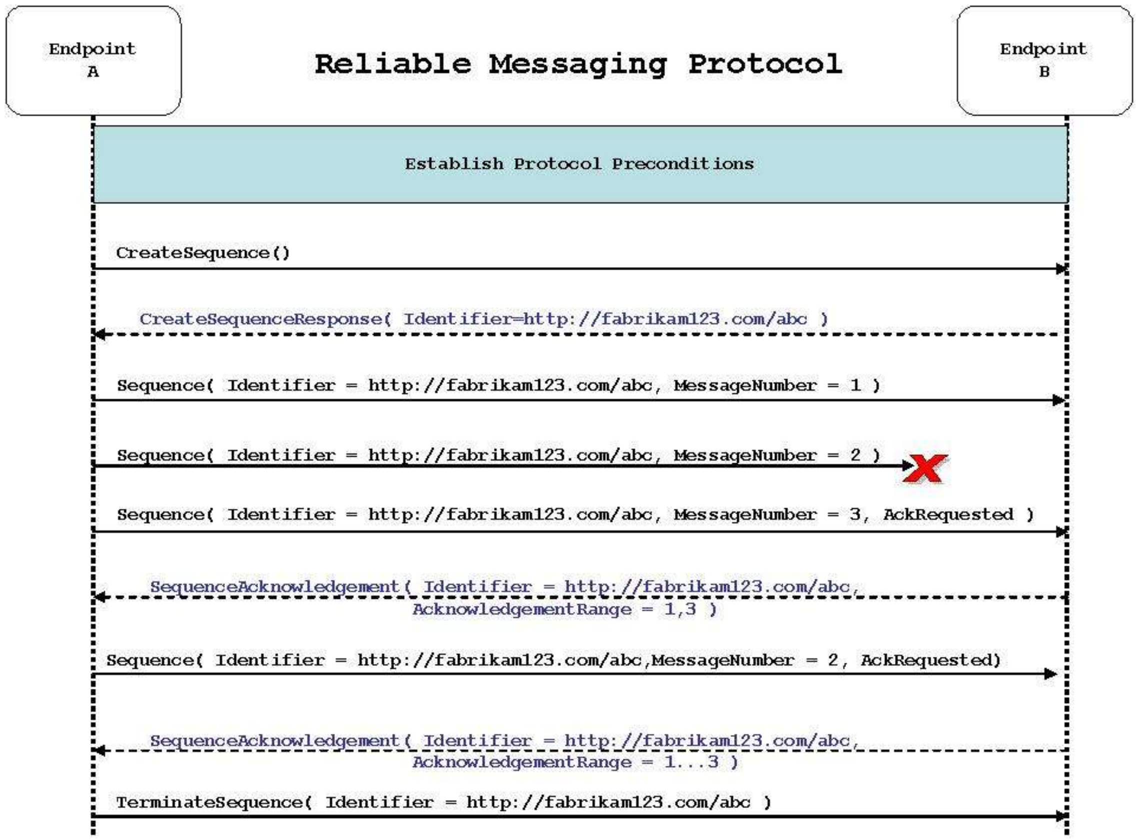
If a secure exchange of messages is required, then the RM Source and RM Destination MUST have a security context.

2.3 Protocol Invariants

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

- The RM Source MUST assign each reliable message a sequence number (defined below) beginning at 1 and increasing by exactly 1 for each subsequent reliable message.

Every acknowledgement issued by the RM Destination MUST include within an acknowledgement range or ranges the sequence number of every message successfully received by the RM Destination and MUST exclude sequence numbers of any messages not yet received.


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)
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)

226 2.4 Example Message Exchange

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

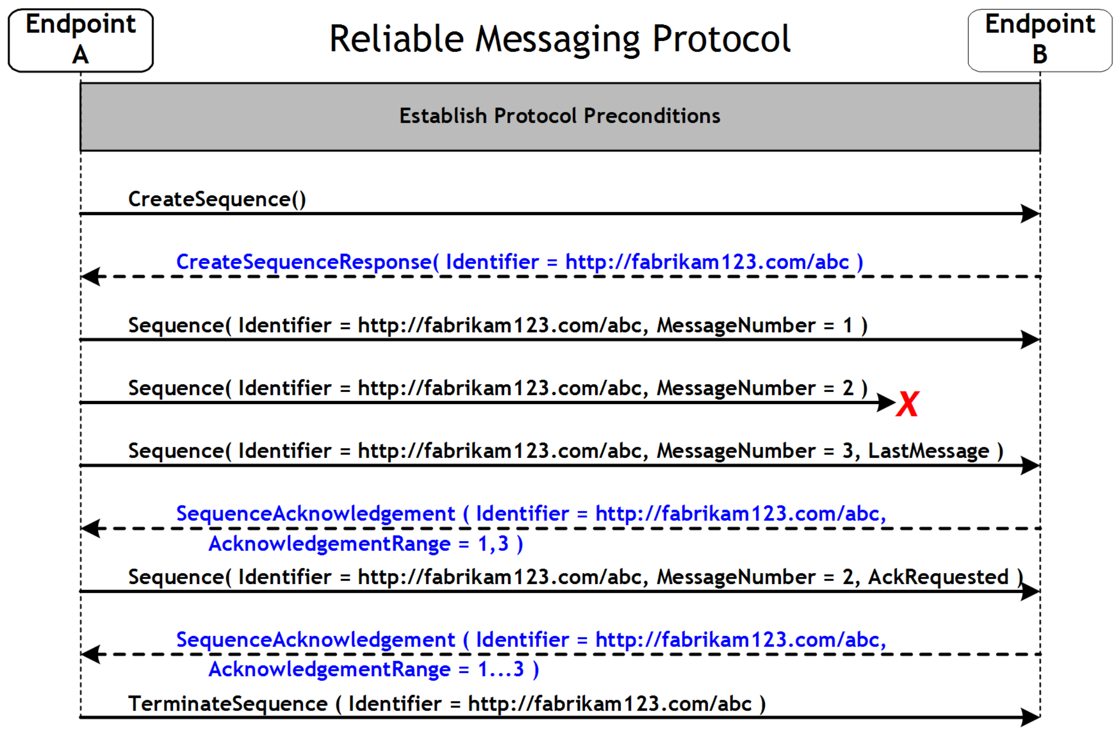

Endpoint A
Reliable Messaging Protocol
Endpoint B
Establish Protocol Preconditions
CreateSequence()
CreateSequenceResponse(Identifier = http://fabrikam123.com/abc)
Sequence(Identifier = http://fabrikam123.com/abc, MessageNumber = 1)
Sequence(Identifier = http://fabrikam123.com/abc, MessageNumber = 2)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

Figure 2: The WS-ReliableMessaging Protocol

1. The protocol preconditions are established. These include policy exchange, endpoint resolution, establishing trust.
2. The RM Source requests creation of a new Sequence.
3. The RM Destination creates a Sequence by returning a globally unique identifier.
4. The RM Source begins sending messages beginning with MessageNumber 1. In the figure the RM Source sends 3 messages.
5. Since the 3rd message is the last in this exchange, the RM Source includes a <wsrm:AckRequested~~LastMessage~~> ~~token~~Header.
6. The 2nd message is lost in transit.
7. The RM Destination acknowledges receipt of message numbers 1 and 3 in response to the RM Source's `<wsrm:~~LastMessage~~AckRequested>` ~~token~~Header.
8. The RM Source retransmits the 2nd message. This is a new message on the underlying transport, but since it has the same sequence identifier and message

number so the RM Destination can recognize it as equivalent to the earlier message, in case both are received.

9. The RM Source includes an `<wsrm:AckRequested>` element so the RM Destination will expedite an acknowledgement.

10. The RM Destination receives the second transmission of the message with MessageNumber 2 and acknowledges receipt of message numbers 1, 2, and 3 which carried the `<wsrm:`~~LastMessage~~AckRequested`>` ~~token~~Header.

11. 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.

12. The RM Destination receives the TerminateSequence message indicating that the RM Source will not be sending any more messages, and reclaims any resources associated with the Sequence.

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

3.1 Sequences

The RM protocol uses a `<wsrm:Sequence>` header block to track and manage the reliable delivery of messages. Messages for which the delivery assurance applies MUST contain a `<wsrm:Sequence>` header block. Each Sequence MUST have a unique `<wsrm:Identifier>` element and each message within a Sequence MUST have 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. ~~In addition to mandatory <wsrm:Identifier> and <wsrm:MessageNumber> elements, the header MAY include a <wsrm:LastMessage> element.~~

There MUST be no more than one `<wsrm:Sequence>` header block in any message.

~~The purpose of the <wsrm:LastMessage> element is to signal to the RM Destination that the message represents the last message in the Sequence.~~

A following exemplar defines its syntax:

```
<wsrm:Sequence ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:MessageNumber> xs:unsignedLong </wsrm:MessageNumber>
    ~~<wsrm:LastMessage/>?~~
    ...
</wsrm:Sequence>
```

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

/wsrm:Sequence

This is the element containing Sequence information for WS-ReliableMessaging. The <wsrm:Sequence> element MUST be understood by the RM Destination. The <wsrm:Sequence> element MUST have 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.

/wsrm:Sequence/wsrm:Identifier

This REQUIRED element MUST contain an absolute URI conformant with RFC2396 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

This REQUIRED element MUST contain an xs:unsignedLong representing the ordinal position of the message within a Sequence. Sequence MessageNumbers start at 1 and monotonically increase throughout the Sequence. If the message number exceeds the internal limitations of an RM Source or RM Destination or reaches the maximum value of an xs:unsignedLong (18,446,744,073,709,551,615), the RM Source or Destination MUST issue a MessageNumberRollover fault.

~~/wsrm:Sequence/wsrm:LastMessage~~

~~This element MAY be included by the RM Source endpoint. The `<wsrm:LastMessage>` element has no content.~~

/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.

~~A RM Source endpoint MUST include a `<wsrm:LastMessage>` element in the `<wsrm:Sequence>` element for the last message in a Sequence. An RM Destination endpoint MUST respond with a `<wsrm:SequenceAcknowledgement>` upon receipt of a `<wsrm:LastMessage>` element. A Sequence MUST NOT use a `<wsrm:MessageNumber>` value greater than that which accompanies a `<wsrm:LastMessage>` element. An RM Destination MUST generate a LastMessageNumberExceeded (See Section 4.6) fault upon receipt of such a message. In the event that an RM Source needs to close a Sequence and there is no application message, the RM Source MAY send a message with an empty body containing `<wsrm:Sequence>` header with the `<wsrm:LastMessage>` element. In this usage, the action URI MUST be:~~

```
~~http://docs.oasis-open.org/wsrm/200510/LastMessage~~
```

~~in preference to the pattern defined in Section 1.2.~~

The following example illustrates a Sequence header block.

```
<wsrm:Sequence>
```

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

3.2 Sequence Acknowledgement

The RM Destination informs the RM Source of successful message receipt using a `<wsrm:SequenceAcknowledgement>` header block. The `<wsrm:SequenceAcknowledgement>` header block MAY be transmitted independently or included on return messages. The RM Destination MAY send a `<wsrm:SequenceAcknowledgement>` header block at any point during which the sequence is valid. The timing of acknowledgements can be advertised using policy and acknowledgements can be explicitly requested using the `<wsrm:AckRequested>` directive (see Section 3.3). 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.

The following exemplar defines its syntax:

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

This REQUIRED element MUST contain an absolute URI conformant with RFC2396 that uniquely identifies the Sequence.

/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

This OPTIONAL element, if present, can occur 1 or more times. It contains a range of message Sequence MessageNumbers successfully received by the receiving endpoint manager. The ranges SHOULD NOT overlap. This element MUST NOT be present if either the `<wsrm:Nack>` or `<wsrm:None>` elements are also present as a child of `<wsrm:SequenceAcknowledgement>`.

/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Upper

This REQUIRED attribute contains an xs:unsignedLong representing the `<wsrm:MessageNumber>` of the highest contiguous message in a Sequence range received by the RM Destination.

/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange/@Lower

This REQUIRED attribute contains an xs:unsignedLong representing the `<wsrm:MessageNumber>` 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

This OPTIONAL element, if present, 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. This element MUST be present when the Sequence is no longer receiving new message for the specified sequence. Note: this element MUST NOT be used when sending a Nack, it can only be used when sending AcknowledgementRanges.

/wsrm:SequenceAcknowledgement/wsrm:Nack

This OPTIONAL element, if present, MUST contain an `xs:unsignedLong` representing the `<wsrm:MessageNumber>` of an unreceived message in a Sequence. This element permits the gap analysis of the `<wsrm:AcknowledgementRange>` elements to be performed at the RM Destination rather than at the RM Source which may yield performance benefits in certain environments. The `<wsrm:Nack>` element MUST NOT be present if either the `<wsrm:AcknowledgementRange>` or `<wsrm:None>` elements are 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

This OPTIONAL element, if present, MUST be used when the RM Destination has not received any messages for the specified sequence. The `<wsrm:None>` element MUST NOT be present if either the `<wsrm:AcknowledgementRange>` or `<wsrm:Nack>` elements are 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>
```

3.3 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 returned.

At any time, the RM Source may request an acknowledgement message from the RM Destination endpoint using an `<wsrm:AckRequested>` header block.

The RM Source endpoint requests this acknowledgement by including an `<wsrm:AckRequested>` header block in the message. An RM Destination that receives a message that contains an `<wsrm:AckRequested>` header block MUST respond with a message containing a `<wsrm:SequenceAcknowledgement>` header block. 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.

The following exemplar defines its syntax:

```
<wsrm:AckRequested ...>
    <wsrm:Identifier ...> xs:anyURI </wsrm:Identifier>
    <wsrm:MessageNumber> xs:unsignedLong </wsrm:MessageNumber> ?
    ...
</wsrm:AckRequested>
```

/wsrm:AckRequested

This element requests an acknowledgement for the identified sequence.

/wsrm:AckRequested/wsrm:Identifier

This REQUIRED element MUST contain an absolute URI, conformant with RFC2396, 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/wsrm:MessageNumber

This OPTIONAL element, if present, MUST contain an xs:unsignedLong representing the highest `<wsrm:MessageNumber>` sent by the RM Source within the Sequence. If present, it MAY be treated as a hint to the RM Destination as an optimization to the process of preparing to transmit a `<wsrm:SequenceAcknowledgement>`.

/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.

3.4 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 `<wsrm:CreateSequenceResponse>` or a `CreateSequenceRefused` fault in the body of the response message. `<wsrm:CreateSequence>` MAY carry an offer to create an inbound sequence which is either accepted or rejected in the `<wsrm:CreateSequenceResponse>`.

The RM Destination of the outbound sequence is the WS-Addressing EndpointReference [WS-Addressing] to which `<wsrm:CreateSequence>` is sent. The RM Destination of the inbound sequence is the WS-Addressing `<wsa:ReplyTo>` of the `<wsrm:CreateSequence>`.

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: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. This element MUST NOT be sent as a header block. The RM Destination MUST respond either with a `<wsrm:CreateSequenceResponse>` response message or a `CreateSequenceRefused` fault.

/wsrm:CreateSequence/wsrm:AcksTo

This REQUIRED element, of type wsa:EndpointReferenceType as specified by WS-Addressing [WS-Addressing] specifies the endpoint reference to which `<wsrm:SequenceAcknowledgement>` messages and faults related to the created Sequence are to be sent.

/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

This REQUIRED element MUST contain an absolute URI conformant with RFC2396 that uniquely identifies 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:Expires

This element, if present, of type `xs:duration` specifies the 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'.

/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.

OPTIONAL/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: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. This element MUST NOT be sent as a header block.

/wsrm:CreateSequenceResponse/wsrm:Identifier

This REQUIRED element MUST contain an absolute URI conformant with RFC2396 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'. This value MUST be equal or lesser 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: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. This element MUST be present if the corresponding `<wsrm:CreateSequence>` message contained an `<wsrm:Offer>` element.

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

This REQUIRED element, of type wsa:EndpointReferenceType as specified by WS-Addressing [WS-Addressing], specifies the endpoint reference to which `<wsrm:SequenceAcknowledgement>` messages related to the accepted Sequence are to be sent.

/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.

3.5 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. Note, 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, i.e. it MUST NOT send any additional message to the RM Destination referencing this sequence. It indicates that the RM Destination can safely reclaim any resources related to the identified Sequence. This element MUST NOT be sent as a header block.

/wsrm:TerminateSequence/wsrm:Identifier

This REQUIRED element MUST contain an absolute URI conformant with RFC2396 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.

3.6 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 even if some of the messages have not been successfully delivered to the RM Destination.

In the case where the RM Source wishes to discontinue use of a sequence, while it can send a TerminateSequence to the RM Destination, since this is a one-way message and due to the possibility of late arriving (or lost) messages and Acknowledgements, this would leave the RM Source unsure of the final ranges of messages that were successfully delivered to the RM Destination.

To alleviate this, the RM Source can send a <wsrm:CloseSequence> element, in the body of a message, to the RM Destination to indicate that 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 RMD. Upon receipt of this message the RM Destination MUST send a

SequenceAcknowledgement to the RM Source. Note, this SequenceAcknowledgement MUST include the <wsrm:Final> element.

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 may 'close' the sequence itself. Please see wsrm:Final above and the SequenceClosed fault below. Note, 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: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 closed.

/wsrm:CloseSequence@Identifier

This REQUIRED attribute contains an absolute URI conformant with RFC2396 that uniquely identifies the sequence.

/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: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/{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.

4 Faults

The fault definitions defined in this section reference certain abstract properties, such as [fault endpoint], that are defined in section 3 of the WS-Addressing [WS-Addressing] specification. Endpoints compliant with this specification MUST include required Message Addressing Properties on all fault messages.

Sequence creation uses a CreateSequence, CreateSequenceResponse request-response pattern. Faults for this operation 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 sequences are detected, these faults are also treated as defined in WS-Addressing. 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. Sequence faults SHOULD be sent to the same [destination] as `<wsrm:SequenceAcknowledgement>` messages. These faults are correlated using the Sequence identifier carried in the detail.

WS-ReliableMessaging faults MUST include as the [action] property the default fault action URI defined in the version of WS-Addressing used in the message. The value from the current version is below for informational purposes:

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/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 4 of WS-Addressing.

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. If absent, no detail element is defined for the fault.

The [Code] property MUST be 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://schemas.xmlsoap.org/ws/2004/08/addressing/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: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 xml:lang="en"> [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. The `<wsrm:SequenceFault>` container MUST only be used in conjunction with the SOAP1.1 fault mechanism. It MUST NOT be used in conjunction with the SOAP1.2 binding.

The following exemplar defines its syntax:

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

This element, if present, MUST contain a qualified name from the set of fault codes defined below.

/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 sent by either the RM Source or the RM Destination to indicate that the endpoint that generated the fault 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.

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 sent by either the RM Source or the RM Destination in response to a message containing an unknown sequence identifier.

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 sent 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 sent 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>
```

4.6 ~~Last Message Number Exceeded~~

~~This fault is sent by an RM Destination to indicate that it has received a message that has a `<wsrm:MessageNumber>` within a Sequence that exceeds the value of the `<wsrm:MessageNumber>` element that accompanied a `<wsrm:LastMessage>` element for the Sequence.~~

~~Properties:~~

~~[Code] Sender~~

~~[Subcode] wsrm:LastMessageNumberExceeded~~

~~[Reason] The value for wsrm:MessageNumber exceeds the value of the MessageNumber accompanying a LastMessage element in this Sequence.~~

~~[Detail]~~

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

4.7 Create Sequence Refused

This fault is sent 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] empty

4.8 Sequence Closed

This fault is sent 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.

Properties:

[Code] Sender

[Subcode] wsrm:SequenceClosed

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

[Detail] <wsrm:Identifier...> xs:anyURI </wsrm:Identifier>

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 and WS-SecureConversation [SecureConversation]. If a Sequence is bound to a specific endpoint, then the security context needs to be established or shared with the endpoint 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]

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

[URI]

T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[XML-ns]

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

[XML-Schema1]

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

[XML-Schema2]

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

[WSSecurity]

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

[Tanenbaum]

"Computer Networks," Andrew S. Tanenbaum, Prentice Hall PTR, 2003.

[WSDL]

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

[WS-Addressing]

D. Box, et al, "Web Services Addressing (WS-Addressing)," August 2004.

6.2 Non-Normative

[WS-Policy]

D. Box, et al, "Web Services Policy Framework (WS-Policy)," September 2004.

[WS-PolicyAttachment]

D. Box, et al, "Web Services Policy Attachment (WS-PolicyAttachment)," September 2004.

[SecurityPolicy]

G. Della-Libra, "Web Services Security Policy Language (WS-SecurityPolicy)," December 2002.

[SecureConversation]

S. Anderson, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," May 2004.

Appendix A.Schema

The normative schema for WS-ReliableMessaging is located at:

```
http://docs.oasis-open.org/wsrm/200510/wsrm.xsd
```

The following copy is provided for reference.

```
<xs:schema targetNamespace="http://docs.oasis-open.org/wsrm/200510/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsrm="http://docs.oasis-open.org/wsrm/200510/"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import
namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
  <!-- Protocol Elements -->
  <xs:complexType name="SequenceType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:element name="MessageNumber" type="xs:unsignedLong"/>
~~      <xs:element name="LastMessage" minOccurs="0">~~
~~        <xs:complexType>~~
~~          <xs:sequence/>~~
~~        </xs:complexType>~~
~~      </xs:element>~~
      <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>
          <ws:sequence>
            <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>
            <ws:element name="Final" minOccurs="0">
              <xs:complexType>
                <xs:sequence/>
              </xs:complexType>
            </ws:element>
          </ws:sequence>
          <xs:element name="Nack" type="xs:unsignedLong"
maxOccurs="unbounded"/>
          <xs:element name="None" minOccurs="0">
            <xs:complexType>
              <xs:sequence/>
            </xs:complexType>
          </xs:element>
        </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:element name="MessageNumber" type="xs:unsignedLong"
minOccurs="0"/>
      <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>
  <!-- Fault Container and Codes -->
  <xs:simpleType name="FaultCodes">
    <xs:restriction base="xs:QName">
      <xs:enumeration value="wsrm:UnknownSequence"/>
      <xs:enumeration value="wsrm:SequenceTerminated"/>
      <xs:enumeration value="wsrm:InvalidAcknowledgement"/>
      <xs:enumeration value="wsrm:MessageNumberRollover"/>
      <xs:enumeration value="wsrm:CreateSequenceRefused"/>
~~      <xs:enumeration value="wsrm:LastMessageNumberExceeded"/>~~
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="SequenceFaultType">
    <xs:sequence>
      <xs:element name="FaultCode" type="wsrm:FaultCodes"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" 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:complexType name="CreateSequenceType">
    <xs:sequence>
```

```
      <xs:element ref="wsrm:AcksTo"/>
      <xs:element ref="wsrm:Expires" minOccurs="0"/>
      <xs:element name="Offer" type="wsrm:OfferType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
It is the authors intent that this extensibility be used to transfer a
Security Token Reference as defined in WS-Security.
</xs:documentation>
        </xs:annotation>
      </xs:any>
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="CreateSequenceResponseType">
    <xs:sequence>
      <xs:element ref="wsrm:Identifier"/>
      <xs:element ref="wsrm:Expires" minOccurs="0"/>
      <xs:element name="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:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Identifier" type="xs:anyURI" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="CloseSequenceResponseType">
    <xs:sequence>
      <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: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: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:schema>
```

1154 Appendix 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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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:LastMessage/>~~
  </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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/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 sending endpoint discovers 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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/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/wsrm/200510/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/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/wsrm/200510/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>
```

Appendix C.WSDL

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

```
http://docs.oasis-open.org/wsrm/200510/wsdl/wsrm.wsdl
```

The following non-normative copy is provided for reference.

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:rm="http://docs.oasis-open.org/wsrm/200510/"
xmlns:tns="http://docs.oasis-open.org/wsrm/200510/wsdl"
targetNamespace="http://docs.oasis-open.org/wsrm/200510/wsdl">
<wsdl:types>
    <xs:schema>
      <xs:import namespace="http://docs.oasis-open.org/wsrm/200510/"
schemaLocation="http://docs.oasis-open.org/wsrm/200510/wsrm.xsd"/>
      <xs:import
namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
    </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:portType name="SequenceAbsractPortType">
    <wsdl:operation name="CreateSequence">
```

```
      <wsdl:input message="tns:CreateSequence"
wsa:Action="http://docs.oasis-open.org/wsrm/200510/CreateSequence"/>
      <wsdl:output message="tns:CreateSequenceResponse"
wsa:Action="http://docs.oasis-
open.org/wsrm/200510/CreateSequenceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="CloseSequence">
      <wsdl:input name="tns:CloseSequence"
wsa:Action="http://docs.oasis-open.org/wsrm/200510/CloseSequence"/>
      <wsdl:output name="tns:CloseSequenceResponse"
wsa:Action="http://docs.oasis-
open.org/wsrm/200510/CloseSequenceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="TerminateSequence">
      <wsdl:input message="tns:TerminateSequence"
wsa:Action="http://docs.oasis-
open.org/wsrm/200510/CreateSequenceResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Appendix D.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

1456 Appendix E.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)

Appendix F.Notices