Web Services Reliable Messaging Protocol (WS-ReliableMessaging)

March 13, 2003

Authors

Ruslan Bilorusets, BEA
Adam Bosworth, BEA
Don Box, Microsoft
Felipe Cabrera, Microsoft
Derek Collison, TIBCO Software
Jon Dart, TIBCO Software
Donald Ferguson, IBM
Christopher Ferris, IBM (Editor)
Tom Freund, IBM
Mary Ann Hondo, IBM
John Ibbotson, IBM
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
John Shewchuk, Microsoft
Tony Storey, IBM

Copyright Notice

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 an independent manner allowing it to be implemented using different network transport 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

WS-ReliableMessaging and related specifications are provided for use as-is and for review and evaluation only. BEA, IBM, Microsoft, and TIBCO Software will solicit your contributions and suggestions in the near future. BEA, IBM, Microsoft, and TIBCO Software make no warrantees or representations regarding the specification in any manner whatsoever.

Acknowledgments

The following individuals have provided invaluable input into the design of the WS-ReliableMessaging specification:

Keith Ballinger, Microsoft
Michael Conner, IBM
Francisco Curbera, IBM
Allen Brown, Microsoft
Steve Graham, IBM
Pat Helland, Microsoft
Rick Hill, Microsoft
Scott Hinkelman, IBM
Tim Holloway, IBM
Efim Hudis, Microsoft
Johannes Klein, Microsoft
Frank Leymann, IBM
Martin Nally, IBM
Peter Niblett, IBM
Jeffrey Schlimmer, Microsoft
James Snell, IBM
Keith Stobie, Microsoft
Stephen Todd, IBM
Satish Thatte, Microsoft
Sanjiva Weerawarana, IBM
Roger Wolter, Microsoft

We also wish to thank the technical writers and development reviewers who provided feedback to improve the readability of the specification.

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 which 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 compliments the WS-Security, 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 RFC2119 [KEYWORDS].

Namespace URIs of the general form "some-URI" represents some application-dependent or context-dependent URI as defined in RFC2396 [URI].

This specification uses an informal syntax to describe the XML grammar of the XML fragments below:

- The syntax appears as an XML instance, but the values indicate the data types instead of values.
- Element names ending in "…" (such as `<element…/>` or `<element…>`) indicate that elements/attributes irrelevant to the context are being omitted.
- Attributed names ending in "…" (such as `name=...`) indicate that the values are specified below.
- Grammar in bold has not been introduced earlier in the document, or is of particular interest in an example.
- `<-- description -->` is a placeholder for elements from some "other" namespace (like ##other in XSD).
- Characters are appended to elements, attributes, and `<!-- descriptions -->` as follows: "?" (0 or 1), "*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to the "?", "*", or "+" characters.
- The XML namespace prefixes (defined below) are used to indicate the namespace of the element being defined.
- Examples starting with `<?xml` contain enough information to conform to this specification; others examples are fragments and require additional information to be specified in order to conform.

XSD schemas and WSDL definitions are provided as a formal definition of grammars [xml-schema1] [WSDL].

1.2 Namespace

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

```
http://schemas.xmlsoap.org/ws/2003/03/rm
```

The namespace prefix "wsrm" used in this specification is associated with this URI.

The following namespaces are used in this document:

Prefix	**Namespace**
wsrm	http://schemas.xmlsoap.org/ws/2003/03/rm
wsu	http://schemas.xmlsoap.org/ws/2002/07/utility
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy

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

```
http://schemas.xmlsoap.org/ws/2003/03/rm/wsrm.xsd
```

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

If an action URI is used then the action URI MUST consist of the reliable messaging namespace URI concatenated with the '#' character and the element name. For example:

```
http://schemas.xmlsoap.org/ws/2003/03/rm#SequenceAcknowledgement
```

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 Reliable Messaging model provides the guarantee that messages sent by the initial sender will be delivered to the ultimate receiver. The guarantee is specified by delivery assurances defined in Section 2.4.

The diagram below illustrates the entities and events in a simple reliable message exchange. First, the Initial Sender sends a message for reliable delivery. The Source accepts the message and transmits it one or more times. After receiving the message the Destination acknowledges it. Finally, the Destination delivers the message to the Ultimate Receiver. The exact roles the entities play and the complete meaning of the events will be defined throughout this specification.

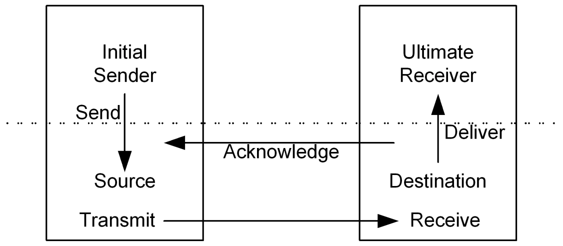

Figure 1: Reliable Messaging Model

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.

Initial Sender: The endpoint which sends a message.

Ultimate Receiver: The endpoint to which a message is delivered.

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

Source: The endpoint that transmits the message.

Destination: The endpoint that receives the message.

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

Deliver: The act of transferring a message to the ultimate recipient. The reliability guarantee is fulfilled at this point.

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

Receipt: The act of reading a message from a network connection

Acknowledgement: The communication from the destination to the 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 source MUST have an endpoint reference that uniquely identifies the destination endpoint; correlations across messages addressed to the unique endpoint are meaningful.
- The source MUST have knowledge of the destination’s policies, if any, and the source MUST be capable of formulating messages that adhere to this policy.
- If a secure exchange of messages is required, then the source and destination MUST have a security context.

2.3 Protocol Invariants

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

- The 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 destination MUST include within an acknowledgement range or ranges the sequence number of every message successfully received by the destination and MUST exclude sequence numbers of any messages not yet received.

2.4 Reliable Messaging Delivery Assurances

Endpoints which implement the WS-ReliableMessaging protocol provide delivery assurances for the delivery of messages sent from the initial sender to the ultimate receiver. The protocol supports the endpoints in providing these delivery assurances. It is the responsibility of the source and destination to fulfill the delivery assurances in the Sequence's policy declarations, or raise an error and terminate the Sequence. An endpoint MAY fail to fulfill its delivery assurance for any reason including an internal error, security violation, a denial of service attack, a network partition or a service outage. If either endpoint fails to fulfill its delivery assurance commitments, an error MUST be raised on either the initial sender or the ultimate receiver or both. The protocol defined here allows endpoints to meet this guarantee for the delivery assurances defined below. These delivery assurances begin with a successful send by the Initial Sender and end with a successful delivery. 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 sequence observed by the ultimate receiver be non-decreasing. It says nothing about duplications or omissions.

2.5 Example Message Exchange

The following figure illustrates a possible message exchange between two reliable messaging endpoints.

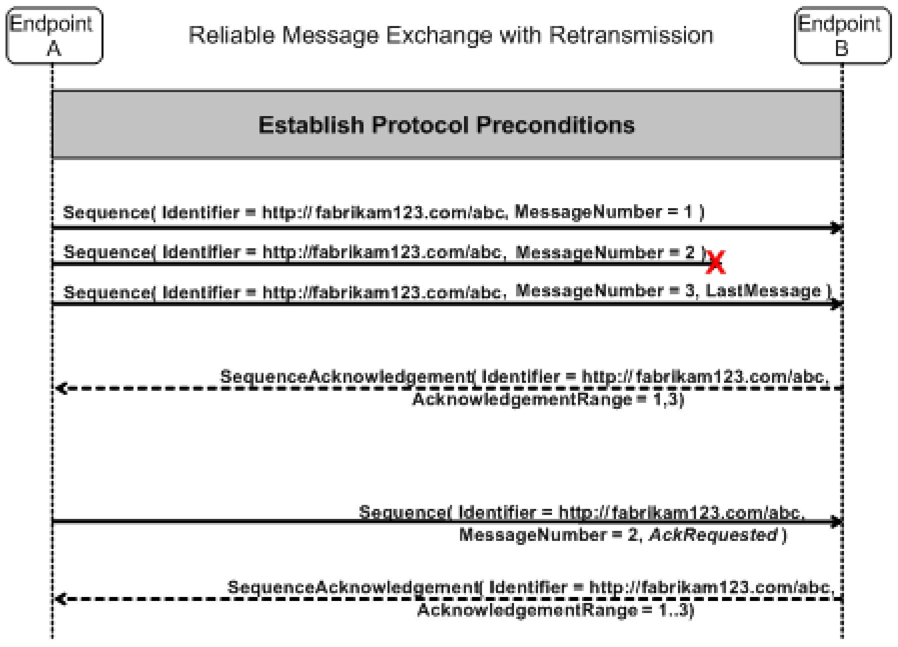
Endpoint A
Reliable Message Exchange with Retransmission
Endpoint B
Establish Protocol Preconditions
Sequence(Identifier = http://fabrikam123.com/abc, MessageNumber = 1)
Sequence(Identifier = http://fabrikam123.com/abc, MessageNumber = 2)
Sequence(Identifier = http://fabrikam123.com/abc, MessageNumber = 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)

Figure 2: The WS-ReliableMessaging Protocol

1. The protocol preconditions are established. These include policy exchange, endpoint resolution, establishing trust.
2. The Source establishes a new sequence by assigning a unique identifier and begins sending messages beginning with MessageNumber 1. In the figure the Source sends 3 messages.
3. Since the 3rd message is the last in this exchange, the Source includes a LastMessage token.
4. The 2nd message is lost in transit.
5. The Destination acknowledges receipt of message numbers 1 and 3.
6. The 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 Destination can recognize it as equivalent to the earlier message, in case both are received.
7. The Source includes an AckRequested element so the Destination will expedite an acknowledgement.
8. The Destination receives the second transmission of the message with MessageNumber 2 and acknowledges receipt of message numbers 1, 2, and 3.
9. The Source receives this acknowledgement and knows the sequence is completed.

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

3. RM Protocol Elements

3.1 Sequences

The RM protocol uses a <Sequence> element to track and manage the reliable delivery of messages. Messages for which the delivery assurance applies MUST contain a <Sequence> element. Each Sequence MUST have a unique <Identifier> element and each message within a Sequence MUST have a <MessageNumber> element which increments by 1 from an initial value of 1. These values are contained within a <Sequence> element accompanying each message being delivered in the context of a Sequence. In addition to mandatory <Identifier> and <MessageNumber> elements, the header MAY include a <LastMessage> element, a <wsu:Expires> element, or both.

There MUST be no more than one <Sequence> element in any message.

The purpose of the <LastMessage> element is to signal to the destination that the message represents the last message in the Sequence.

A following pseudo schema fragment defines the <Sequence> element.

```
<wsrm:Sequence ...>
    <wsu:Identifier> [URI] </wsu:Identifier>
    <wsrm:MessageNumber> [unsignedLong] </wsrm:MessageNumber>
    <wsrm:LastMessage/>?
    <wsu:Expires> [dateTime] </wsu:Expires>?
    ...
</wsrm:Sequence>
```

The following example illustrates a Sequence element.

```
<wsrm:Sequence>
   <wsu:Identifier>http://fabrikam123.com/abc</wsu:Identifier>
   <wsrm:MessageNumber>10</wsrm:MessageNumber>
   <wsrm:LastMessage/>
</wsrm:Sequence>
```

The following describes the content model of the Sequence header.

/wsrm:Sequence
   This is the element containing Sequence information for WS-ReliableMessaging

/wsrm:Sequence/wsu:Identifier
   This required element MUST contain a URI conformant with [RFC2396] that uniquely identifies the Sequence. The normative definition of this element may be found in [WSCoordination].

/wsrm:Sequence/wsrm:MessageNumber
   This required element MUST contain an 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 reaches the maximum value of an unsignedLong (18,446,744,073,709,551,615), no

new sequence numbers can be generated and the source MUST issue a MessageNumberRollover fault.

/wsrm:Sequence/wsrm:LastMessage
This element MAY be included by the sending endpoint. The <LastMessage> element has no content. A sending endpoint MUST include a <LastMessage> element in the <Sequence> element for the last message in a Sequence. A Sequence MUST NOT use a <MessageNumber> value greater than that which accompanies a <LastMessage> element. A destination MUST generate a LastMessageNumberExceeded fault upon receipt of such a message.

/wsrm:Sequence/wsu:Expires
This optional element indicates the point in time at which the Sequence will expire. Its value is a dateTime. Its presence MUST override any previously established expiration. Expiration should be established as described in Section 5.2. A SequenceAcknowledgment that covers the MessageNumber of the message that carries a wsu:Expires element indicates acceptance of the new value.

/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.

3.2. Sequence Acknowledgement

The destination informs the source of successful message receipt using an acknowledgement. Acknowledgements may be transmitted individually or included on return messages. The destination MAY send an acknowledgement at any point. The timing of acknowledgements can be advertised using policy and acknowledgments can be explicitly requested using the <AckRequested> directive.

The following pseudo schema defines the <SequenceAcknowledgement> element:

```
<wsrm:SequenceAcknowledgement ...>
    <wsu:Identifier> [URI] </wsu:Identifier>
    <wsrm:AcknowledgementRange ...
        Upper="[unsignedLong]"
        Lower="[unsignedLong]"/> +
    ...
<wsrm:SequenceAcknowledgement>
```

The following describes the content model of the <SequenceAcknowledgement> element.

/wsrm:SequenceAcknowledgement
This element contains the Sequence acknowledgement information.

/wsrm:SequenceAcknowledgement/wsu:Identifier

This required element MUST contain a URI conformant with [RFC2396] that uniquely identifies the Sequence. The normative definition of this element may be found in [WSCoordination].

/wsrm:SequenceAcknowledgement/wsrm:AcknowledgementRange
This required element can occur 1 or more times. It contains a range of message Sequence MessageNumbers successfully received by the receiving endpoint manager.

/wsrm:SequenceAcknowledgement/wsrm:AcknowledgmentRange/@Upper
This required attribute contains an unsignedLong representing the <MessageNumber> of the highest contiguous message in a Sequence range.

/wsrm:SequenceAcknowledgement/wsrm:AcknowledgmentRange/@Lower
This required attribute contains an unsignedLong representing the <MessageNumber> of the lowest contiguous message in a Sequence range.

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

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

The following examples illustrate <SequenceAcknowledgement> elements:

- Message numbers 1..10 inclusive in a Sequence have been received by the receiving endpoint.

```
<wsrm:SequenceAcknowledgment>
   <wsu:Identifier>http://fabrikam123.com/abc</wsu:Identifier>
   <wsrm:AcknowledgmentRange Upper="10" Lower="1"/>
</wsrm:SequenceAcknowledgment>
```

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

```
<wsrm:SequenceAcknowledgment>
   <wsu:Identifier>http://fabrikam123.com/abc</wsu:Identifier>
   <wsrm:AcknowledgmentRange Upper="2" Lower="1"/>
   <wsrm:AcknowledgmentRange Upper="6" Lower="4"/>
   <wsrm:AcknowledgmentRange Upper="10" Lower="8"/>
</wsrm:SequenceAcknowledgment>
```

3.3. Request Acknowledgement

The purpose of the <AckRequested> element is to signal to the destination that the source is requesting that a <SequenceAcknowledgment> be returned.

At any time, the source may request an acknowledgement message from the destination point using an <AckRequested> element.

The sending endpoint requests this acknowledgement by including an <AckRequested> element in the message. A destination that receives a message that contains an <AckRequested> element MUST respond with a message containing a <SequenceAcknowledgement> element.

The following pseudo schema defines the <AckRequested> element:

```
<wsrm:AckRequested ...>
    <wsu:Identifier> [URI] </wsu:Identifier>
    ...
</wsrm:AckRequested>
```

/wsrm:AckRequested
This element requests an acknowledgement for the identified sequence.

/wsrm:AckRequested/wsu:Identifier
This required element contains the URI of the sequence to which the request applies. The normative definition of this element may be found in [WSCoordination].

/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.

4. Policy Assertions

WS-Policy [WS-Policy], WS-PolicyAttachments [WS-PolicyAttachments] and WS-PolicyAssertions [PolicyAssertions] collectively define a framework, model and grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web Services-based system. This specification leverages the WS-Policy family of specifications to enable a destination endpoint to describe and advertise its capabilities and/or requirements, and to enable a source destination to communicate to the destination the selected characteristics that apply for a given Sequence. The set of policy assertions for reliable delivery is defined below.

4.1. Spec Version

The protocol determines invariants maintained by the reliable messaging endpoints and the directives used to track and manage the delivery of messages. The assertion that will be used to identify the protocol (and version) either used or supported (depending on context) is the wsp:SpecVersion assertion that is defined in the WS-PolicyAssertions specification [PolicyAssertions].

An example use of this assertion to indicate an endpoint’s support for the WS-ReliableMessaging protocol follows:

```
<wsp:SpecVersion
wsp:URI="http://schemas.xmlsoap.org/ws/2003/03/rm"
wsp:Usage="wsp:Required"/>
```

4.2. Delivery Assurance

This section establishes well known names for application level delivery assurances between the ultimate receiver and the destination. The delivery assurance is an informational policy assertion as it does not affect the transmission protocol defined above.

The following pseudo schema defines this element:

```
<wsrm:DeliveryAssurance Value="[QName]" ...> ... </wsrm:DeliveryAssurance>
```

The following describes the attributes and tags listed in the syntax above:

/wsrm:DeliveryAssurance
This element is a policy assertion as defined in WS-PolicyAssertions. It contains an identifier that characterizes the reliability aspects of the sequence.

/wsrm:DeliveryAssurance/@Value
This attribute contains one aspect of a sequence's reliability guarantee. Its value is a QName. The following QNames are defined to have these specific meanings:

QName	**Delivery assurance**
wsrm:AtMostOnce	The messages in the sequence will be delivered to the application without duplication.
wsrm:AtLeastOnce	The messages in the sequence are assured to be delivered to the application at least once.
wsrm:ExactlyOnce	The messages in the sequence are assured to be delivered exactly once. This assertion is equivalent to AtMostOnce and AtLeastOnce.
wsrm:InOrder	The messages in the sequence are assured to be delivered to the application in the order they were sent.

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

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

The example below indicates that messages should be delivered at least once.

```
<wsrm:DeliveryAssurance Value="wsrm:AtLeastOnce"/>
```

4.3. Sequence Expiration

The lifetime of a sequence may be established using the <wsu:Expires> policy assertion defined in Web Services Security Addendum [SecurityAddendum].

The following pseudo schema defines this element:

```
<wsu:Expires ...> [dateTime] <wsu:Expires>
```

The following describes the attributes and tags listed in the syntax above:

wsu:Expires

This element is defined in WS-Security Addendum. It indicates the expiration time of the sequence.

The example below indicates should expire January 1, 2100.

```
<wsu:Expires>2100-01-01T12:00:00.000-00:00</wsu:Expires>
```

4.4. InactivityTimeout

This assertion specifies (in milliseconds) a period of inactivity for a Sequence. If during this duration an endpoint has received no application or control messages, the endpoint MAY consider the Sequence to have been terminated due to inactivity.

The following pseudo schema defines this element:

```
<wsrm:InactivityTimeout Milliseconds="[unsignedLong]" ...> ...
</wsrm:InactivityTimeout>
```

The following describes the attributes and tags listed in the syntax above:

/wsrm:InactivityTimeout
This element is a policy assertion as defined in WS-PolicyAssertions. It specifies a period of inactivity after which the endpoints MAY consider the Sequence to have been terminated due to inactivity.

/wsrm:InactivityTimeout/@Milliseconds
This attribute carries the value of the inactivity timeout duration, specified in milliseconds, expressed as an unsignedLong.

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

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

4.5. Retransmission Interval

A source may optionally specify a base retransmission interval for a sequence. If no acknowledgement has been received for a given message within the interval, the source will retransmit the message. The retransmission interval may be modified at the source's discretion during the lifetime of the sequence. This assertion does not alter the formulation of messages as transmitted, only the timing of their transmission.

The sequence may optionally specify that the interval will be adjusted using the commonly known exponential backoff algorithm.

The following pseudo schema defines these elements:

```
<wsrm:BaseRetransmissionInterval Milliseconds="[unsignedLong]" ...> ...
</wsrm:BaseRetransmissionInterval>


<wsrm:ExponentialBackoff ...> ... </wsrm:ExponentialBackoff>
```

The following describes the attributes and tags listed in the syntax above:

/wsrm:BaseRetransmissionInterval
This element is a policy assertion as defined in WS-PolicyAssertions. This assertion specifies the interval in milliseconds the source will wait after transmitting a message and before retransmitting the message.

/wsrm:BaseRetransmissionInterval/@Milliseconds
This attribute specifies the interval in milliseconds the source will wait after transmitting a message and before retransmitting the message.

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

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

/wsrm:ExponentialBackoff
This element is a policy assertion as defined in WS-PolicyAssertions. This assertion specifies that the retransmission interval will be adjusted using the exponential backoff algorithm.

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

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

The example below indicates that unacknowledged messages will be retransmitted after two seconds and that exponential backoff algorithm will be used for timing of successive retransmissions should the message continue to go unacknowledged.

```
<wsp:Policy>
    <wsrm:BaseRetransmissionInterval wsp:Usage="wsp:Observed"
        Milliseconds="2000"/>
    <wsrm:ExponentialBackoff wsp:Usage="wsp:Observed"/>
</wsp:Policy>
```

4.6. Acknowledgement Interval

Acknowledgements can be sent on return messages or sent stand alone. In the case that a return message is not available to send an acknowledgement a destination may wait for up to the acknowledgement interval before sending a stand alone acknowledgement. If there are no unacknowledged messages, the destination may choose not to send an acknowledgement.

This assertion does not alter the formulation of messages or acknowledgements as transmitted. Its purpose is to communicate the timing of acknowledgements so that the source may tune appropriately. It does not alter the meaning of the <AckRequested> directive.

The following pseudo schema defines this element:

```
<wsrm:AcknowledgementInterval Milliseconds="[unsignedLong]" ...> ...
</wsrm:AcknowledgementInterval>
```

The following describes the attributes and tags listed in the syntax above:

/wsrm:AcknowledgementInterval

This assertion specifies the duration in milliseconds after which the destination will transmit an acknowledgement.

/wsrm:AcknowledgementInterval/@Milliseconds

This required attribute holds the value of the assertion.

/wsrm:AcknowledgementInterval/{any}

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

/wsrm:AcknowledgementInterval/{@any}

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

The example below indicates the destination MAY buffer acknowledgements for up to 2/10ths of a second.

```
<wsp:Policy>
    <wsrm:AcknowledgementInterval wsp:Usage="wsp:Observed"
            Milliseconds="200" />
</wsp:Policy>
```

5. Attaching Policy Assertions to Sequences

The policy assertions defined in Section 4. Policy Assertions above can be attached to an individual Sequence or to Sequences in general. The policy assertions related to a Sequence are established using an agreement as discussed in Section 5.2.

The WS-PolicyAttachment specification provides a means of attaching policy expressions to domain expressions. This specification recommends the use of the <PolicyAttachment> element to attach policy to a Sequence and defines the extension below as a means of attaching policy to a Sequence.

5.1 SequenceRef

The <SequenceRef> expression is provided to enable policy assertions to be declared for individual Sequences, or Sequences whose URI match a specified prefix. These assertions apply to the Sequence itself, not to messages in the Sequence.

The following pseudo schema defines this element:

```
<wsrm:SequenceRef Match="wsrm:Exact|wsrm:Prefix" ?>
    [URI]
</wsrm:SequenceRef>
```

The following describes the attributes and tags listed in the syntax above:

/wsrm:SequenceRef
: This element is a policy attachments domain expression which identifies a Sequence or Sequences. Its content is a URI.

/wsrm:SequenceRef/@Match
: This attribute specifies whether the URI in the body must match exactly or must only match a prefix of the URI for a Sequence.

Policy assertions can be attached to an individual Sequence by embedding a <SequenceRef> element that identifies that Sequence within a `<PolicyAttachment>` element.

The following example attaches the policy "profile" defined in Appendix B with the Sequence with the Identifier of "`http://fabrikam123.com/abc`". Additionally, it requires that the Sequence use this version of the WS-ReliableMessaging specification in a manner that provides the delivery assurances associated with "AtMostOnce":

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsrm:SequenceRef>
      <wsu:Identifier>http://fabrikam123.com/abc</wsu:Identifier>
    </wsrm:SequenceRef>
  </wsp:AppliesTo>
  <wsp:Policy>
    <wsp:SpecVersion
        wsp:URI="http://schemas.xmlsoap.org/ws/2003/03/rm"
        wsp:Usage="wsp:Required"/>
    <wsrm:DeliveryAssurance Value="wsrm:AtMostOnce"
        wsp:Usage="wsp:Required"/>
  </wsp:Policy>
  <wsp:PolicyReference
    Ref="http://schemas.xmlsoap.org/ws/2003/03/rm/baseTimingProfile.xml"/>
</wsp:PolicyAttachment>
```

5.2 Metadata Exchange

There are many ways that Web service partners can exchange WSDL interface and WS-Policy information. Some will be informal, out of band approaches using e-mail, posting documents on a Web site or in UDDI, etc.

To facilitate interoperability there is a need for a metadata exchange model. We anticipate that future specifications will define a WSDL interface and message model that Web service clients MAY use to obtain WSDL and WS-Policy information for a service at a URL or endpoint reference.

6. SequenceFault Element

The purpose of the <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 fault container is meant to be used in conjunction with the SOAP fault mechanism. For example, when used as a detail element in SOAP 1.2 the identifier in the container correlates the fault with a specific sequence.

The following pseudo schema fragment defines the <SequenceFault> element.

```
<wsrm:SequenceFault>
  <wsu:Identifier> [Sequence URI] </wsu:Identifier>
  <wsrm:FaultCode> ... </wsrm:FaultCode>
    <wsrm:AcknowledgementRange ...
        Upper="[unsignedLong]"
        Lower="[unsignedLong]"/> ?
  ...
</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/wsu:Identifier
  This required element MUST contain a URI conformant with [RFC2396] that uniquely identifies the Sequence for which the fault was triggered.

/wsrm:SequenceFault/wsrm:FaultCode
  This required element MUST contain a qualified name from the set of fault codes defined below.

/wsrm:SequenceFault/wsrm:AcknowledgementRange
  This element MUST only appear if the <FaultCode> is `wsrm:InvalidAcknowledgement`. This element MUST occur 1 or more times if the <FaultCode> is `wsrm:InvalidAcknowledgment`. Each value is copied from the <SequenceAcknowledgment> element that triggers the fault.

/wsrm:SequenceFault/wsrm:AcknowledgmentRange/@Upper
  This required attribute contains an unsignedLong representing the <MessageNumber> of the highest contiguous message in a Sequence range.

/wsrm:SequenceFault/wsrm:AcknowledgmentRange/@Lower
  This required attribute contains an unsignedLong representing the <MessageNumber> of the lowest contiguous message in a Sequence range.

/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.

6.1. Sequence Terminated

This fault is sent by either the source or the destination to indicate that the endpoint that generates 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. The qualified name of the fault code is:

```
wsrm:SequenceTerminated
```

6.2. Unknown Sequence

This fault is sent by either the source or the destination in response to a message containing an unknown sequence identifier. The qualified name of the fault code is:

```
wsrm:UnknownSequence
```

6.3. Invalid Acknowledgement

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

The qualified name of the fault code is:

```
wsrm:InvalidAcknowledgement
```

An InvalidAcknowledgment SequenceFault MUST contain the set of <AcknowledgementRange> elements that were carried in the <SequenceAcknowledgment> that triggered the fault. For example:

```
<wsrm:SequenceFault>
   <wsu:Identifier> [Sequence URI] </wsu:Identifier>
   <wsrm:FaultCode>wsrm:InvalidAcknowledgment</wsrm:FaultCode>
   <wsrm:AcknowledgmentRange Upper="1" Lower="10"/>
</wsrm:SequenceFault>
```

6.4. Message Number Rollover

This fault is sent by the source to indicate that it has run out of message numbers for a sequence. It is an unrecoverable error and terminates the Sequence. The qualified name of the fault code is:

```
wsrm:MessageNumberRollover
```

6.5. Last Message Number Exceeded

This fault is sent by a destination to indicate that it has received a message that has a <MessageNumber> within a Sequence that exceeds the value of the <MessageNumber> element that accompanied a <LastMessage> element for the Sequence. This is an unrecoverable error and terminates the Sequence. The qualified name of the fault code is:

```
wsrm:LastMessageNumberExceeded
```

6.6. Sequence Refused

This fault is sent by a destination to indicate that it cannot begin a requested Sequence:

```
wsrm:SequenceRefused
```

7. 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:SequenceAcknowlegement> header MAY be signed independently because 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. 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 an 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 tenant 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 race 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. 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.

8. References

[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.

[WSPolicy]
Web Services Policy Framework, Microsoft, IBM, BEA, SAP, 18 December, 2002.

[PolicyAttachment]

Web Services Policy Framework, Microsoft, IBM, BEA, SAP, 18 December, 2002.

[PolicyAssertions]
"Web Services Policy Assertions Language (WS-PolicyAssertions)," Microsoft, IBM, BEA, SAP, 18 December 2002.

[WSCoordination]
Web Services Coordination Framework, Microsoft, IBM, BEA, 9 August, 2002.

[WSSecurity]
"Web Services Security (WS-Security)," Microsoft, IBM, VeriSign, 5 April 2002.

[SecurityAddendum]
"Web Services Security Addendum," Microsoft, IBM, VeriSign, 18 August 2002.

[SecureConversation]
"Web Services Secure Conversation Language (WS-SecureConversation)," Microsoft, IBM, VeriSign, RSA Security, 18 December 2002.

[SecurityPolicy]
"Web Services Security Policy Language (WS-SecurityPolicy)," Microsoft, IBM, VeriSign, RSA Security, 18 December 2002.

[WSTrust]
"Web Services Trust Language (WS-Trust)," Microsoft, IBM, VeriSign, RSA Security, 18 December 2002.

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

Appendix A Example Timing Profile

The efficiency of WS-ReliableMessaging implementations can be improved using widely known timing profiles. The following profile can be found at:

```
http://schemas.xmlsoap.org/ws/2003/03/rm/baseTimingProfile.xml
```

This profile defines a base retransmission interval of 3 seconds with exponential backoff. Destinations can buffer acknowledgements for 1 second. If one endpoint has not received messages from the other for 1 day or longer, the sequence MAY be terminated.

```
<wsp:Policy>
  <wsrm:BaseRetransmissionInterval Milliseconds="3000"
    wsp:Usage="wsp:Observed" />
  <wsrm:ExponentialBackoff wsp:Usage="wsp:Observed" />
  <wsrm:InactivityTimeout Milliseconds="86400000"
    wsp:Usage="wsp:Observed" />
  <wsrm:AcknowledgementInterval Milliseconds="1000"
    wsp:Usage="wsp:Observed" />
</wsp:Policy>
```

Appendix B Schema

The normative schema for WS-Reliable Message is located at:

```
http://schemas.xmlsoap.org/ws/2003/03/rm/rm.xsd
```

The following copy is provided for reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:import namespace="http://schemas.xmlsoap.org/ws/2002/12/policy"
schemaLocation="http://schemas.xmlsoap.org/ws/2002/12/policy/policy.xsd"/>
  <!-- Protocol Elements -->
  <xs:complexType name="SequenceType">
    <xs:sequence>
      <xs:element ref="wsu:Identifier"/>
      <xs:element name="MessageNumber" type="xs:unsignedLong"/>
      <xs:element name="LastMessage" type="xs:ENTITY" minOccurs="0"/>
      <xs:element ref="wsu:Expires" 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="Sequence" type="wsrm:SequenceType"/>
  <xs:element name="SequenceTerminate">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="wsu:Identifier"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="SequenceAcknowledgment">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="wsu:Identifier"/>
        <xs:element name="AcknowledgmentRange" 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: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>
  <xs:complexType name="AckRequestedType">
    <xs:sequence>
      <xs:element ref="wsu: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"/>
  <!-- Policy Assertions -->
  <xs:element name="InactivityTimeout">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="wsrm:PolicyAssertionType">
          <xs:attribute name="Milliseconds" type="xs:unsignedLong"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="BaseRetransmissionInterval">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="wsrm:PolicyAssertionType">
          <xs:attribute name="Milliseconds" type="xs:unsignedLong"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="ExponentialBackoff">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="wsrm:PolicyAssertionType">
          <xs:attribute name="Value" type="xs:boolean" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="AcknowledgementInterval">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="wsrm:PolicyAssertionType">
          <xs:attribute name="Milliseconds" type="xs:unsignedLong"
use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="PolicyAssertionType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##other"/>
```

```
</xs:complexType>
<xs:simpleType name="DeliveryAssuranceEnum">
  <xs:restriction base="xs:QName">
    <xs:enumeration value="wsrm:AtMostOnce"/>
    <xs:enumeration value="wsrm:AtLeastOnce"/>
    <xs:enumeration value="wsrm:ExactlyOnce"/>
    <xs:enumeration value="wsrm:InOrder"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="DeliveryAssurance">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="wsrm:PolicyAssertionType">
        <xs:attribute name="Value" type="xs:QName" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </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:restriction>
</xs:simpleType>
<xs:complexType name="SequenceFaultType">
  <xs:sequence>
    <xs:element ref="wsu:Identifier"/>
    <xs:element name="FaultCode" type="xs:QName"/>
    <xs:any namespace="##any"/>
  </xs:sequence>
  <xs:anyAttribute/>
</xs:complexType>
<xs:element name="SequenceFault" type="wsrm:SequenceFaultType"/>
```

```
  <!-- Sequence Reference Domain Expression -->
  <xs:complexType name="SequenceRefType">
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute/>
    <xs:attribute name="Identifier" type="xs:anyURI" use="required"/>
    <xs:attribute name="Match" type="wsrm:MatchChoiceType" use="optional"
/>
  </xs:complexType>
  <xs:simpleType name="MatchChoiceType">
    <xs:restriction base="xs:QName">
      <xs:enumeration value="wsrm:Exact"/>
      <xs:enumeration value="wsrm:Prefix"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="SequenceRef" type="wsrm:SequenceRefType"/>
</xs:schema>
```

Appendix C Message Examples

C.1 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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfcbc9e
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsrm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu: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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsrm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu: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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
 <S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
  </wsa:MessageID>
  <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
  <wsa:ReplyTo>
   <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:ReplyTo>
  <wsrm:Sequence>
   <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
   <wsrm:MessageNumber>3</wsrm:MessageNumber>
   <wsrm:LastMessage/>
  </wsrm:Sequence>
 </S:Header>
 <S:Body>
  <!-- Some Application Data -->
 </S:Body>
</S:Envelope>
```

C.2 First Acknowledgement

Message number 2 has not been received by the 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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
 <S:Header>
  <wsa:MessageID>
   http://fabrikam123.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
  </wsa:MessageID>
  <wsa:To>http://Business456.com/serviceA/789</wsa:To>
  <wsa:ReplyTo>
   <wsa:Address>http://fabrikam123.com/serviceB/123</wsa:Address>
  </wsa:ReplyTo>
  <wsrm:SequenceAcknowledgment>
   <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
   <wsrm:AcknowledgmentRange Upper="1" Lower="1"/>
   <wsrm:AcknowledgmentRange Upper="3" Lower="3"/>
  </wsrm:SequenceAcknowledgment>
 </S:Header>
 <S:Body/>
</S:Envelope>
```

C.3 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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
```

```
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
 <S:Header>
  <wsa:MessageID>
   http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
  </wsa:MessageID>
  <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
  <wsa:ReplyTo>
   <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
  </wsa:ReplyTo>
  <wsrm:Sequence>
   <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
   <wsrm:MessageNumber>2</wsrm:MessageNumber>
  </wsrm:Sequence>
  <wsrm:AckRequested>
   <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
  </wsrm:AckRequested>
 </S:Header>
 <S:Body>
  <!-- Some Application Data -->
 </S:Body>
</S:Envelope>
```

C.4 Final Acknowledgment

The 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/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
xmlns:wsrm="http://schemas.xmlsoap.org/ws/2003/03/rm"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
 <S:Header>
  <wsa:MessageID>
   http://fabrikam123.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
  </wsa:MessageID>
  <wsa:To>http://Business456.com/serviceA/789</wsa:To>
  <wsa:ReplyTo>
   <wsa:Address>http://fabrikam123.com/serviceB/123</wsa:Address>
  </wsa:ReplyTo>
  <wsrm:SequenceAcknowledgment>
   <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
   <wsrm:AcknowledgmentRange Upper="3" Lower="1"/>
  </wsrm:SequenceAcknowledgment>
 </S:Header>
 <S:Body/>
</S:Envelope>
```