

The Replay Model

Using WS-RM 1.0 across the public Internet and with anonymous clients

November 5th, 2007

Version 1.0.3

Authors

Paul Fremantle, WSO2

Marc Goodner, Microsoft

Center for ServiceOriented Infrastructure, Danish National IT and Telecom Agency (NITA)

Copyright Notice

(c) 2007 Microsoft Corporation, Inc., NITA and WSO2 Inc. All rights reserved.

Microsoft, NITA and WSO2 (collectively the "Authors") hereby grant permission to copy and display 'The Replay Model' paper (the "Document"), in any medium without fee or royalty, provided that you include the following on ALL copies of the Document, that you make:

1. A link or URL to the Document at one of the Authors' websites
2. The copyright notice as shown in the Document.

THE DOCUMENT IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DOCUMENT.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Document or its contents without specific, written prior permission. Title to copyright in the Document will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

Summary

This document does not define a new protocol. Rather it defines a particular usage of the WS-ReliableMessaging 1.0 and WS-Addressing protocols in order to enable reliable request-response Web service messages for anonymous clients. As such this document introduces no new wire formats, schema WSDL or Policy assertions.

When a Web service client is not directly addressable it is said to be anonymous. The classic example of an anonymous client is when the requestor is behind a firewall or a NAT service. In this case only the requestor can initiate communication to the responder. The responder is limited to sending responses on an underlying transport backchannel created by the request, the canonical example being the HTTP response channel that is created for any HTTP request. This is the defining characteristic of an anonymous client: it can only receive responses from a Web service on a transport backchannel provided by a request that it initiates. This document addresses problems faced by anonymous Web service clients that use WS-RM. When an anonymous client is using WS-RM the way the responder communicates acknowledgements and response retransmissions required by the WS-RM protocol to the requestor are restricted. This document describes how work within these restrictions to enable the use of WS-RM with request-response Web service messages for anonymous clients.

This paper is intended for an audience familiar with general web services topics. Familiarity with WS-ReliableMessaging and WS-Addressing is not absolutely essential but would be helpful.

Contents

1. Introduction.....	2
2. The Replay Model.....	3
Appendix A – Acknowledgements.....	10
Appendix B – XML Namespaces	10
Appendix C – References.....	10

1. Introduction

SOAP is a widely used XML based communications protocol that was designed to be web-friendly. In the current Internet environment, a large part of being web-friendly is working with non-addressable clients, also known as anonymous clients. An anonymous client is one that cannot receive incoming connections because it is behind a firewall or network address translation (NAT) router. The way this is usually achieved is by layering over HTTP, which is a well understood protocol and supported by most firewalls, and in particular the usage of HTTP request-response.

For example, imagine you are sitting at your favourite coffee shop, browsing the web over WiFi. The IP address of your laptop will most likely be a “private IP address” [[RFC1918](#)] which is actually being handled by a firewall that uses Network Address Translation. Here your laptop is acting as an anonymous client. What this means is that while your laptop can initiate connections to remote sites, it is not possible for machines on the wider Internet to initiate

connections to your machine. All the information coming to your laptop is being sent on the response channel of the HTTP connections that are being initiated from your laptop. This response channel we will refer to as the backchannel.

Now let's look at the WS-ReliableMessaging 1.0 specification [WS-ReliableMessaging]. This specification enables SOAP based reliable messaging based on *acknowledgements*. The receiver of messages, known as the Reliable Messaging Destination (RMD), sends acknowledgements back to the Sender (RMS). If a message goes missing and is unacknowledged, the RMS will resend it. The sender must provide an address where acknowledgements will be sent, this is referred to as the *AcksTo* address.

This works fine for one-way SOAP messages with anonymous clients. This is because the acknowledgements can flow on the backchannel and the client (the sender) can always contact the HTTP server (the destination).

Now let's take the case where there are request-response flows at the SOAP level. To transmit a request and application SOAP response the HTTP server is now also acting as a RM Sender and the client is acting as an RM Destination. The problem comes if a SOAP response message goes missing, the very problem WS-RM is intended to address. The HTTP server will need to resend this message, but remember it cannot initiate new connections to the client. So effectively it cannot resend messages even though it is using WS-RM.

The result is that without employing some clever techniques, we cannot implement two-way reliability over a synchronous HTTP connection. This paper describes the techniques employed to make this work with WS-RM 1.0.

This is a non-normative document and does not provide a definitive specification of the WS-ReliableMessaging or WS-Addressing languages. This document does use a normative style of language to describe the usage of each of these protocols with respect to the Replay Model described by this paper. This paper does not override the referenced protocols. The use of normative language is only used to emphasize by common convention the particular usage of these protocols that is laid out below. Appendix B lists all the XML Namespaces that are used in this document.

2. The Replay Model

Before we see how we can use WS-RM for request-response with an anonymous client, it is worth explaining how this works in the case with an addressable client. In the case where the server can contact the client, this is handled by using WS-Addressing [WS-Addressing] to specify a real address for the client. WS-Addressing has the capability of specifying ReplyTo addresses (EndpointReferences). These allow the client to define its own endpoints where messages should be sent. It is also known as the *Duplex* model because there are two independent HTTP

flows happening – one from the client to the server carrying requests, and a separate flow from the server to the client carrying responses.

The synchronous HTTP case is defined as the *Anonymous* case in WS-Addressing and is the default if no ReplyTo header is passed to the server. In addition, WS-Addressing defines a specific URL that can be used to indicate the anonymous case (e.g. <http://www.w3.org/2005/08/addressing/anonymous>).

In order to resolve this, some WS-RM systems use a particular model we will call the replay model.

What happens in this model is that the client will continue to resend the request message until it receives a response. The client is replaying the request message until it receives a valid response. In fact the request message may have already been successfully received by the server but despite this the client will resend it, thereby creating a valid backchannel on which to receive the response. In this model the client is initiating connections so that the server can always send any response messages required on the backchannel.

Server Processing Model

The processing model can be defined from the perspective of the client and the server. The client is defined as the initiator of the initial CreateSequence. The server is defined as the acceptor of the CreateSequence.

This section covers the server perspective of the replay model.

- A) The Server **MUST** use this model when it accepts a wsrn:CreateSequence operation that:
 - includes an Offered Sequence and
 - where the AcksTo and ReplyTo both reference the WS-Addressing anonymous endpoint.
- B) The Server **SHOULD** add a wsrn:SequenceAcknowledgement header to every successful response message.
- C) When the server is responding to a request with a SOAP message, then either:
 - The server **MUST** acknowledge the request in the same message as the response flows, or
 - If the response to that message has already been acknowledged, then the server **MAY** return an acknowledgement for the request without resending the response.

- D) If there is no response to a given request – i.e. the request is a one-way message, then the Server **MUST** respond with an acknowledgement that includes the request message in one of the ranges.
- E) The Server **MUST** respond to a replayed request message with the correct response to that message, unless it has already received an acknowledgement for that response, in which case it can respond with an HTTP 202 Accepted and/or a SequenceAcknowledgement.
- F) The Server **SHOULD** respond to an incoming LastMessage with a LastMessage for the Offered Sequence
- G) The Server **SHOULD** respond to an incoming TerminateSequence with a corresponding TerminateSequence for the corresponding Offered Sequence.

Client Processing Model

The processing model can be defined from the perspective of the client and the server. The client is defined as the initiator of the initial CreateSequence. The server is defined as the acceptor of the CreateSequence.

This section covers the client perspective of the replay model.

- H) If the Client aims to use this model, it **MUST** send a CreateSequence with an Offer. The wsrn:AcksTo address must be a valid WS-Addressing Anonymous URI and the wsa:ReplyTo address and wsrn:AcksTo **MUST** match.
- I) The Client **SHOULD** transmit messages in the Sequence message order.
- J) If the response to a request does not include a SOAP payload, the Client **MUST** replay the request, or raise a fault.
- K) If the Client receives a response to a given request containing a SOAP payload, and the payload includes a wsrn:SequenceAcknowledgement including the request message in a range, then the body of the SOAP message will be the response to the request, or empty in case of a one-way operation.
- L) The Client **SHOULD NOT** send a TerminateSequence if it is aware that there are messages waiting to be transmitted by the Server.

Implication

Looking at the Server processing model items (C) and (D) we can deduce that: the Server **MUST NOT** respond with an HTTP 200 OK and without a SOAP payload to any requests using this replay model.

Piggybacking

The message flows in this model contain what is commonly referred to as piggy backing of acknowledgements, e.g. an acknowledgement for an endpoint is carried on another message to the same endpoint. We see this in carrying acknowledgements to requests from the client in item C of the Server processing model for example. The relationship of the two sequences through the use of an Offer ensures that the Client can reasonably be expected to process the piggy backed acknowledgements.

Examples

In the following diagrams the marker A5 indicates a Sequence header for Sequence A with message number 5, and the marker Ack-A[1-2] indicates a SequenceAcknowledgement header for Sequence A with range 1-2.

The first diagram demonstrates a simple request-response interaction with two sets of calls. The second response fails due to a timeout on the incoming socket and therefore the client replays the request.

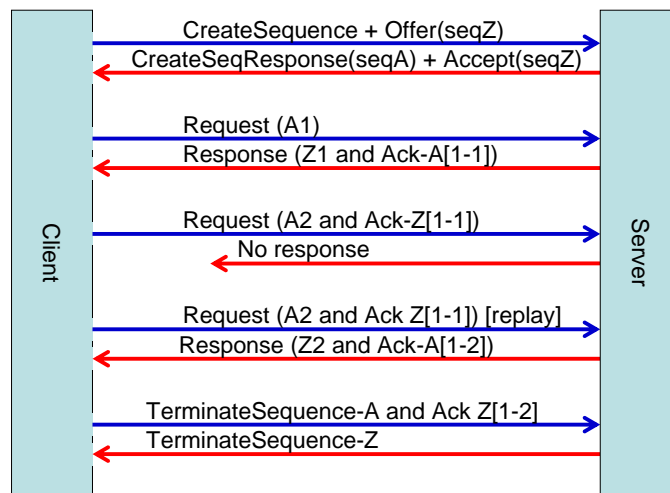


Figure 1: Req-resp example flow

In the second diagram, there is a mixture of one-way (Event) and two-way (Request/Response) calls. This demonstrates that the message numbers in the two sequences are not necessarily aligned – for example, message A2 is correlated at the application level with message Z1.

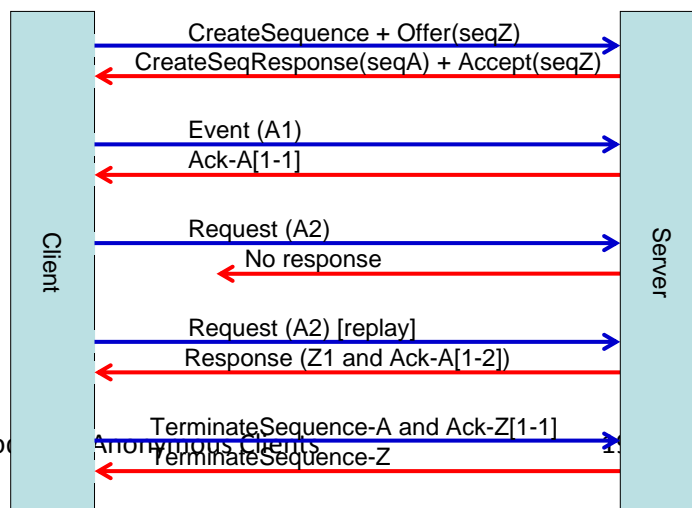


Figure 2: Req-resp and one-way mixed example flow

Message Traces

Below are message traces of the replay model in use.

Create Sequence

This message pair shows the establishment of the sequence.

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
    </a:Action>
    <a:MessageID>urn:uuid:72307b82-8d74-466b-a5fb-09a3a76dbe34</a:MessageID>
    <a:To s:mustUnderstand="1">http://Server/WsRequestReplyService</a:To>
  </s:Header>
  <s:Body>
    <CreateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <AcksTo>
        <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
      </AcksTo>
      <Offer>
        <Identifier>urn:uuid:e1a0f03b-bd16-4a48-b409-e6022baa0d35</Identifier>
      </Offer>
    </CreateSequence>
  </s:Body>
</s:Envelope>
```

CreateSequence Request

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequenceResponse
    </a:Action>
    <a:RelatesTo>urn:uuid:72307b82-8d74-466b-a5fb-09a3a76dbe34</a:RelatesTo>
  </s:Header>
  <s:Body>
    <CreateSequenceResponse xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</Identifier>
      <Accept>
        <AcksTo>
          <a:Address>http:// Server/WsRequestReplyService</a:Address>
        </AcksTo>
      </Accept>
    </CreateSequenceResponse>
  </s:Body>
</s:Envelope>
```

CreateSequence Response

One-way

This message pair shows a one-way application request and the sequence acknowledgement for the request.

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"
  xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <r:Sequence s:mustUnderstand="1">
```

```

    <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
    <r:MessageNumber>1</r:MessageNumber>
  </r:Sequence>
  <a:Action
    s:mustUnderstand="1">http://tempuri.org/IRequestReplyService/Ping
  </a:Action>
  <a:To s:mustUnderstand="1">http://Server/WsRequestReplyService</a:To>
</s:Header>
<s:Body>
  <Ping xmlns="http://tempuri.org/">
    <message>One way call #0</message>
  </Ping>
</s:Body>
</s:Envelope>

```

One-way request message

```

<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"
  xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <r:SequenceAcknowledgement>
      <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
      <r:AcknowledgementRange Lower="1" Upper="1"/>
    </r:SequenceAcknowledgement>
    <a:Action s:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/SequenceAcknowledgement
    </a:Action>
  </s:Header>
  <s:Body/>
</s:Envelope>

```

Acknowledgement on response, no application response

Request / Response

This message pair shows application request/response messages with sequence acknowledgements.

```

<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"
  xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <r:Sequence s:mustUnderstand="1">
      <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
      <r:MessageNumber>2</r:MessageNumber>
    </r:Sequence>
    <a:Action s:mustUnderstand="1">
      http://tempuri.org/IRequestReplyService/PingWithEcho
    </a:Action>
    <a:MessageID>urn:uuid:e9188ba0-dd1d-4008-86e8-c21377eaac8a</a:MessageID>
    <a:ReplyTo>
      <a:Address>http://www.w3.org/2005/08/addressing/anonymous</a:Address>
    </a:ReplyTo>
    <a:To s:mustUnderstand="1">http://Server/WsRequestReplyService</a:To>
  </s:Header>
  <s:Body>
    <PingWithEcho xmlns="http://tempuri.org/">
      <message>Request-Reply call #0</message>
    </PingWithEcho>
  </s:Body>
</s:Envelope>

```

Request message

```

<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"

```

```

xmlns:a="http://www.w3.org/2005/08/addressing">
<s:Header>
  <r:Sequence s:mustUnderstand="1">
    <r:Identifier>urn:uuid:e1a0f03b-bd16-4a48-b409-e6022baa0d35</r:Identifier>
    <r:MessageNumber>1</r:MessageNumber>
  </r:Sequence>
  <r:SequenceAcknowledgement>
    <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
    <r:AcknowledgementRange Lower="1" Upper="2"/>
  </r:SequenceAcknowledgement>
  <a:Action s:mustUnderstand="1">
    http://tempuri.org/IRequestReplyService/PingWithEchoResponse
  </a:Action>
  <a:RelatesTo>urn:uuid:e9188ba0-dd1d-4008-86e8-c21377eaac8a</a:RelatesTo>
</s:Header>
<s:Body>
  <PingWithEchoResponse xmlns="http://tempuri.org/">
    <PingWithEchoResult>Request-Reply call #0</PingWithEchoResult>
  </PingWithEchoResponse>
</s:Body>
</s:Envelope>

```

Response message

Terminate Sequence

This message pair shows corresponding TerminateSequence messages for each sequence being used. This would be triggered after receipt of LastMessage on the sequences (see F above), not shown in the traces presented here.

```

<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"
xmlns:a="http://www.w3.org/2005/08/addressing">
<s:Header>
  <r:SequenceAcknowledgement>
    <r:Identifier>urn:uuid:e1a0f03b-bd16-4a48-b409-e6022baa0d35</r:Identifier>
    <r:AcknowledgementRange Lower="1" Upper="3"/>
  </r:SequenceAcknowledgement>
  <a:Action s:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
  </a:Action>
  <a:MessageID>urn:uuid:16684278-3ef7-493a-9ab7-3fa0898cec76</a:MessageID>
  <a:To s:mustUnderstand="1">http://Server/WsRequestReplyService</a:To>
</s:Header>
<s:Body>
  <r:TerminateSequence>
    <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
  </r:TerminateSequence>
</s:Body>
</s:Envelope>

```

Terminate Sequence Request

```

<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:r="http://schemas.xmlsoap.org/ws/2005/02/rm"
xmlns:a="http://www.w3.org/2005/08/addressing">
<s:Header>
  <r:SequenceAcknowledgement>
    <r:Identifier>urn:uuid:901def33-29da-447a-a3e6-05e39afd279d</r:Identifier>
    <r:AcknowledgementRange Lower="1" Upper="5"/>
  </r:SequenceAcknowledgement>
  <a:Action s:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
  </a:Action>
</s:Header>
<s:Body>
  <r:TerminateSequence>

```

```

    <r:Identifier>urn:uuid:e1a0f03b-bd16-4a48-b409-e6022baa0d35</r:Identifier>
  </r:TerminateSequence>
</s:Body>
</s:Envelope>

```

Terminate Sequence Response

Appendix A – Acknowledgements

This document has been developed as a result of joint work with many individuals and teams, including:
 Stefan Batres, Microsoft
 Chamikara Jayalath, Purdue University
 Margaret Lye, Microsoft

Appendix B – XML Namespaces

The following table lists XML namespaces that are used in this document. The choice of any namespace prefix is arbitrary and not semantically significant.

Prefix	Namespace	Specification(s)
wsrn	http://schemas.xmlsoap.org/ws/2005/02/rm	[WS-ReliableMessaging]
s	http://schemas.xmlsoap.org/soap/envelope/ or http://www.w3.org/2003/05/soap-envelope	[SOAP]
wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]

Appendix C – References

[WS-ReliableMessaging]	R. Bilorusets, et al, "Web Services Reliable Messaging (WS-ReliableMessaging)," February 2005. http://schemas.xmlsoap.org/ws/2005/02/rm
[SOAP]	W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509

- [HTTP] IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1"
January 1997
<http://www.ietf.org/rfc/rfc2068.txt>
- [HTTPS] IETF Standard, "The TLS Protocol", January 1999.
<http://www.ietf.org/rfc/rfc2246.txt>
- [RFC 2119] S. Bradner, "Key words for use in RFCs to Indicate
Requirement Levels," RFC 2119, Harvard University, March
1997.
<http://www.ietf.org/rfc/rfc2119.txt>