SOAP-over-UDP

September 2004

Authors

Harold Combs, Lexmark
Martin Gudgin (editor), Microsoft
John Justice, Microsoft
Gopal Kakivaya, Microsoft
David Lindsey, Lexmark
David Orchard, BEA
Alain Regnier, Ricoh
Jeffrey Schlimmer, Microsoft
Stacy Simpson, Lexmark
Hiroshi Tamura, Ricoh
Don Wright, Lexmark
Kenny Wolf, Microsoft

Copyright Notice

Abstract

This specification defines a binding for SOAP envelopes to user datagrams.

Status

SOAP-over-UDP and related specifications are provided as-is and for review and evaluation only. BEA Systems, Lexmark, Microsoft, and Ricoh make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

Many application protocol patterns match the semantics of the User Datagram Protocol (UDP) [RFC 768]. Some do not require the delivery guarantees of TCP while others make use of multicast transmission. In order to allow Web services to support these patterns, we need a way to map SOAP envelopes to user datagrams. This support is essential for services

using WS-Discovery, where the use of multicast and need for low connection overhead makes UDP a natural choice. It is anticipated that other protocols will have similar requirements. This specification defines a binding of SOAP to user datagrams, including message patterns, addressing requirements, and security considerations.

1.1 Requirements

This specification intends to meet the following requirements:

- Support a one-way message-exchange pattern (MEP) where a SOAP envelope is carried in a user datagram.
- Support a request-response message-exchange pattern (MEP) where SOAP envelopes are carried in user datagrams.
- Support multicast transmission of SOAP envelopes carried in user datagrams.
- Support both SOAP 1.1 [SOAP 1.1] and SOAP 1.2 [SOAP 1.2] envelopes.

1.2 Relationship to Web Service Specifications

This specification provides a binding appropriate for:

- SOAP 1.1 [SOAP 1.1]
- SOAP 1.2 [SOAP 1.2]

Messages conforming to either SOAP specification can use this binding. This specification relies on WS-Addressing [WS-Addressing].

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

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

2.2 Terminology

Receiver
: The endpoint terminating a SOAP/UDP datagram

Sender
: The endpoint originating a SOAP/UDP datagram

SOAP/UDP datagram
: A user datagram containing a SOAP envelope in the data octets

User datagram
: A User Datagram Protocol (UDP) packet

3. UDP Packet

Except as noted explicitly below, this specification does not constrain RFC 768 [RFC 768].

3.1 Source Address and Port

For security reasons, the source address MUST be supplied at the UDP packet level and MUST be the IPv4 or IPv6 address of the sender; the receiver SHOULD reject SOAP/UDP datagrams that have inappropriate values for the source address.

A source port MAY be specified. If a source port is not specified then the source port is assumed to be the default port for the SOAP-over-UDP protocol.

3.2 Data Octets

The data octets MUST contain a SOAP envelope [SOAP 1.1, SOAP 1.2]. The SOAP envelope MUST fit within a single datagram, that is it MUST be small enough that the overall datagram is less than 65,536 (2^16) octets.

The SOAP envelope MUST use the mechanisms defined in WS-Addressing [WS-Addressing].

4. Message Patterns

This specification supports the following message patterns:

- Unicast one-way
- Multicast one-way
- Unicast request, unicast response
- Multicast request, unicast response

as detailed in the rest of this section.

4.1 One-way

This specification uses the constructs in WS-Addressing for one-way messages. The one-way message is sent in a user datagram.

4.1.1 One-way Example

```
(001) <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
         xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
(002)   <S:Header>
(003)     <wsa:To>http://fabrikam.com/Server</wsa:To>
(004)     <wsa:Action>http://fabrikam.com/Probe</wsa:Action>
(005)     <wsa:MessageId>
          uuid:1da72f1a-5546-493c-934c-a9e3577e206a
          </wsa:MessageId>
(006)   </S:Header>
(007)   <S:Body>
(008)     ...
(009)   </S:Body>
(010) </S:Envelope>
```

This example shows a one-way SOAP message. Lines 001-002 are standard SOAP elements. Lines 003-005 specify various WS-Addressing headers. Note that despite the fact that the **[destination]** for the message is specified using a URI that uses the http scheme, the message is still transmitted over UDP. Lines 006-010 show standard SOAP elements.

4.2 Request-response

This specification uses the constructs in WS-Addressing for request and response messages. The request message is sent in one user datagram, response messages are sent in separate user datagrams.

4.2.1 Anonymous [reply endpoint]

WS-Addressing defines a URI, "http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous", that can appear in the **[address]** property of an endpoint reference. If the **[reply endpoint]** property of a SOAP message transmitted over UDP has an **[address]** property with this value, the UDP source address (and source port) is considered to be the address to which reply messages should be sent.

The implied value of the **[reply endpoint]** property for SOAP messages transmitted over UDP is an endpoint reference with an **[address]** property whose value is "http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous".

4.2.2 Request Example

```
(001) <S:Envelope xmlns:S=http://www.w3.org/2003/05/soap-envelope
         xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
(002)   <S:Header>
(003)     <wsa:To>http://fabrikam.com/Server</wsa:To>
(004)     <wsa:Action>http://fabrikam.com/Probe</wsa:Action>
(005)     <wsa:MessageId>
          uuid:9ceada16-2403-4404-a8cc-60799acd9d1c
          </wsa:MessageId>
(006)     <wsa:ReplyTo>
            <wsa:Address>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
            </wsa:Address>
          </wsa:ReplyTo>
(007)   </S:Header>
(008)   <S:Body>
(009)     ...
(010)   </S:Body>
(011) </S:Envelope>
```

This example shows a request SOAP message. Lines 001-002 are standard SOAP elements. Lines 003-005 specify various WS-Addressing headers. Note that despite the fact that the **[destination]** for the message is specified using a URI that uses the http scheme, the message is still transmitted over UDP. Line 6 shows a **[reply endpoint]** header specifying the anonymous URI (see section 4.2.1). Lines 007-011 show standard SOAP elements.

4.2.3 Response Example

```
(001) <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
         xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
(002)    <S:Header>
(003)      <wsa:To>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
           </wsa:To>
(004)      <wsa:Action>http://fabrikam.com/ProbeMatch</wsa:Action>
(005)      <wsa:MessageId>
           uuid:5a6ed11a-7a80-409a-82bf-43c4c5092911
           </wsa:MessageId>
(006)      <wsa:RelatesTo>
           uuid:9ceada16-2403-4404-a8cc-60799acd9d1c
           </wsa:RelatesTo>
(007)    </S:Header>
(008)    <S:Body>
(009)      ...
(010)    </S:Body>
(011) </S:Envelope>
```

This example shows a response SOAP message. Lines 001-002 are standard SOAP elements. Lines 003-005 specify various WS-Addressing headers. Note that the **[destination]** for the message is specified as the anonymous URI. Line 006 shows a **[relationship]** header indicating that this message is a reply to the example message in Section 4.2.2. Lines 007-011 show standard SOAP elements.

4.3 Multicast

The message patterns defined above can be used with unicast or multicast transmission of UDP datagrams with the following restriction: The response in a request-response message pattern MUST NOT be multicast.

Note that in the case of a multicast request, unicast response MEP, the sender of the request might receive multiple responses.

Multicast SOAP/UDP datagrams SHOULD be scoped to ensure they are not forwarded beyond the boundaries of the administrative system. This MAY be done with either TTL or administrative scopes [RFC 2365], depending on what is implemented in the network. If TTL is used it is RECOMMENDED that the TTL value be set to 1.

The destination IP address of a multicast message MUST be a multicast group.

4.4 Retransmission

To avoid repeated packet collisions, implementation retransmission SHOULD observe good practices such as using exponential back-off algorithms and spreading. An implementation MAY use the algorithm defined in Appendix A. If a message is to be retransmitted it MUST

have a **[message id]** property. For each transmission of such a message, the value of the **[message id]** property MUST be the same.

5. Message Encoding

The algorithm defined in Appendix F of XML 1.0 [XML 1.0] should be used to determine whether a message is encoded as XML. If use of said algorithm does not result in an XML serialization, the encoding is undefined.

6. URI Scheme

This section defines a URI scheme for UDP endpoints. The scheme allows hostname and port to be specified. Resolving such a URI provides the information needed to send messages to a UDP endpoint per the protocol defined in this document.

6.1 Syntax

The syntax of the URI scheme is as follows:

```
soap.udp: // <host> [ : <port> ] [/ <rel_path>] [ ? <query> ]
```

The syntax and interpretation of the host, port, rel_path and query portions is as defined in RFC 2396 [RFC2396].

6.2 Semantics

The semantics of resolving a soap.udp URI are as follows:

1. Use the port portion as the port number, if specified, otherwise use the default port as the port number.
2. Resolve the host portion to an IP address.
3. Using the message protocol defined in this document, send a message to the IP address determined in step 2 using the port number determined in step 1.

7. Security Considerations

It is recommended that all messages be secured using the mechanisms described in [WS-Security] to prevent tampering or falsification.

All critical headers, such as those described in [WS-Addressing], and the message body, need to be included in signatures to bind all parts of the message together.

Recipients should verify that the sender has the right to speak for the specified source or response location (if one is provided).

Messages should be accepted and processed only from trusted sources (either directly trusted or indirectly trusted via third parties).

The UDP packet size introduces a challenge for secure messages due to its limited size. For this reason it is recommended that security tokens not be passed but referenced using the Key Identifier mechanisms described in [WS-Security].

8. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Erik Christensen (Microsoft), David Langworthy (Microsoft), Yaniv Pessach (Microsoft), Stefan Pharies (Microsoft), Sam Rhodus (Lexmark), Jerry Thrasher (Lexmark), Mike Vernal (Microsoft), Elliot Waingold (Microsoft), Dave Whitehead (Lexmark).

9. References

[**RFC 768**]
J. Postel, "User Datagram Protocol," RFC 768, August 1980.

[**RFC 2119**]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, March 1997.

[**RFC 2365**]
D. Meyer, "Administratively Scoped IP Multicast," RFC 2365, July 1998.

[**RFC 2396**]
T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, August 1998.

[**SOAP 1.1**]
D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," May 2000.

[**SOAP 1.2 Part 1**]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

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

[**WS-Security**]
A. Nadalin, et al, "Web Services Security: SOAP Message Security V1.0," March 2004.

[**XML 1.0**]
T. Bray, et al, "Extensible Markup Language (XML) 1.0 (Second Edition)," October 2000.

Appendix I (non-normative) – Example retransmission algorithm

Constants referenced within the algorithm are defined in Table 1 (for unicast messages) and Table 2 (for unreliable multicast messages).

Retry and back-off algorithm.

1. Transmit; *_UDP_REPEAT--;
2. If *_UDP_REPEAT <= 0 goto Step 9;
3. Generate a random number T in [UDP_MIN_DELAY .. UDP_MAX_DELAY];
4. Wait T milliseconds;
5. Transmit; *_UDP_REPEAT--;
6. If *_UDP_REPEAT <= 0 goto Step 9;
7. T = T * 2;If T > UDP_UPPER_DELAY then T = UDP_UPPER_DELAY;
8. goto 4
9. Done.

Table 1: Protocol Retry and back-off constants for unicast messages

Constant / Message	Value
UNICAST_UDP_REPEAT	2
UDP_MIN_DELAY	50
UDP_MAX_DELAY	250

UDP_UPPER_DELAY	500

Table 2: Protocol Retry and back-off constants for unreliable multicast messages

Constant / Message	Value
MULTICAST_UDP_REPEAT	4
UDP_MIN_DELAY	50
UDP_MAX_DELAY	250
UDP_UPPER_DELAY	500

Appendix II (non-normative) — Example duplicate detection mechanisms

1. A receiver keeps a list of the last *n* messages received along with their **[message id]** properties [WS-Addressing]. When a new (non-duplicate) message arrives, the oldest message is removed from the list.
2. A receiver tracks all messages received in the last *x* milliseconds along with their **[message id]** property [WS-Addressing]. Messages received more than *x* milliseconds ago are removed from the list.

For both approaches any message arriving with a **[message id]** property identical to one of those the receiver has in its list is a duplicate. Messages with unique values for the **[message id]** property are not duplicates.

The timestamp specified in the Security header block [WS-Security] MAY be used to limit the duration for which **[message id]** properties need to be remembered.