Web Services Addressing (WS-Addressing)

W3C Member Submission 10 August 2004

This version:
http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/
Latest version:
http://www.w3.org/Submission/ws-addressing/
Authors:
Don Box, Microsoft (Editor)
Erik Christensen, Microsoft
Francisco Curbera, IBM (Editor)
Donald Ferguson, IBM
Jeffrey Frey, IBM
Marc Hadley, Sun Microsystems, Inc.
Chris Kaler, Microsoft
David Langworthy, Microsoft
Frank Leymann, IBM
Brad Lovering, Microsoft
Steve Lucco, Microsoft
Steve Millet, Microsoft
Nirmal Mukhi, IBM
Mark Nottingham, BEA
David Orchard, BEA
John Shewchuk, Microsoft
Eugène Sindambiwe, SAP
Tony Storey, IBM
Sanjiva Weerawarana, IBM
Steve Winkler, SAP

Copyright Notice

Abstract

WS-Addressing provides transport-neutral mechanisms to address Web services and

messages. Specifically, this specification defines XML [XML 1.0, XML Namespaces] elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. This specification enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document.

By publishing this document, W3C acknowledges that BEA, IBM, Microsoft, SAP, and Sun Microsystems, Inc. have made a formal submission to W3C for discussion. Publication of this document by W3C indicates no endorsement of its content by W3C, nor that W3C has, is, or will be allocating any resources to the issues addressed by it. This document is not the product of a chartered W3C group, but is published as potential input to the W3C Process. Publication of acknowledged Member Submissions at the W3C site is one of the benefits of W3C Membership. Please consult the requirements associated with Member Submissions of section 3.3 of the W3C Patent Policy. Please consult the complete list of acknowledged W3C Member Submissions.

Table of Contents

1. Introduction

Web Services Addressing (WS-Addressing) defines two interoperable constructs that

convey information that is typically provided by transport protocols and messaging systems. These constructs normalize this underlying information into a uniform format that can be processed independently of transport or application. The two constructs are *endpoint references* and *message information headers*.

A Web service endpoint is a (referenceable) entity, processor, or resource where Web service messages can be targeted. Endpoint references convey the information needed to identify/reference a Web service endpoint, and may be used in several different ways: endpoint references are suitable for conveying the information needed to access a Web service endpoint, but are also used to provide addresses for individual messages sent to and from Web services. To deal with this last usage case this specification defines a family of message information headers that allows uniform addressing of messages independent of underlying transport. These message information headers convey end-to-end message characteristics including addressing for source and destination endpoints as well as message identity.

Both of these constructs are designed to be extensible and re-usable so that other specifications can build on and leverage endpoint references and message information headers.

The following example illustrates the use of these mechanisms in a SOAP 1.2 message being sent from http://business456.example/client1 to http://fabrikam123.example /Purchasing:

```
(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:MessageID>
(004)      uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
(005)    </wsa:MessageID>
(006)    <wsa:ReplyTo>
(007)      <wsa:Address>http://business456.example/client1</wsa:Address>
(008)    </wsa:ReplyTo>
(009)    <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
(010)    <wsa:Action>http://fabrikam123.example/SubmitPO</wsa:Action>
(011)   </S:Header>
(012)   <S:Body>
(013)     ...
(014)   </S:Body>
(015) </S:Envelope>
```

Lines (002) to (011) represent the header of the SOAP message where the mechanisms defined in the specification are used. The body is represented by lines (012) to (014).

Lines (003) to (010) contain the message information header blocks. Specifically, lines (003) to (005) specify the identifier for this message and lines (006) to (008) specify the endpoint to which replies to this message should be sent as an Endpoint Reference. Line (009) specifies the address URI of the ultimate receiver of this message. Line (010) specifies an Action URI identifying expected semantics.

1.1. Notational Conventions

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

When describing abstract data models, this specification uses the notational convention used by the XML Infoset [XML Infoset]. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas [XML Schema Part 1, Part 2], this specification uses the notational convention of WS-Security [WS-Security]. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

1.2. Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [XML Namespaces]).

Prefix	**Namespace**
S	http://www.w3.org/2003/05/soap-envelope
S11	http://schemas.xmlsoap.org/soap/envelope
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
xs	http://www.w3.org/2001/XMLSchema

Table 1 Prefixes and Namespaces used in this specification

WS-Addressing is defined in terms of the XML Information Set [XML Infoset].
WS-Addressing is conformant to the SOAP 1.2 [SOAP 1.2] processing model; SOAP 1.2 is not a requirement for using the constructs defined in this specification.
WS-Addressing is also designed to be able work with WSDL 1.1 [WSDL 1.1] described

services. The examples in this specification use an XML 1.0 [XML 1.0] representation but this is not a requirement.

All information items defined by WS-Addressing are identified by the XML namespace URI [XMLNamespaces] "`http://schemas.xmlsoap.org/ws/2004/08/addressing`". A normative XML Schema [XML Schema Part 1, Part 2] document can be obtained by dereferencing the XML namespace URI.

2. Endpoint References

This section defines the model and syntax of an endpoint reference.

This specification introduces a new description element type, the endpoint reference, with the intent of supporting a set of dynamic usage patterns not currently appropriately covered by WSDL 1.1 [WSDL 1.1]. In particular, this specification intends to facilitate the following usage scenarios:

- Dynamic generation and customization of service endpoint descriptions.
- Identification and description of specific service instances that are created as the result of stateful interactions.
- Flexible and dynamic exchange of endpoint information in tightly coupled environments where communicating parties share a set of common assumptions about specific policies or protocols that are used during the interaction.

To support these scenarios, we define a lightweight and extensible mechanism to dynamically identify and describe service endpoints and instances. Because of the current limits of the WSDL 1.1 extensibility model, the WSDL 1.1 service and port elements cannot be used to cover the use cases listed above. Endpoint references logically extend the WSDL description model (e.g., portTypes, bindings, etc.), but do not replace it. Endpoint references will be used instead of WSDL <service/> elements in the following cases:

- Specific instances of a stateful service need to be identified or its instance-specific configuration details need to be transmitted.
- A lightweight, self-contained description of a service endpoint needs to be communicated. In particular, this may be necessary when the details of the endpoint configuration are already shared by the communicating parties, but specific policy information needs to be added or updated, typically as a result of a dynamic configuration process.

Endpoint references complement and do not replace the WSDL/1.1 <wsdl:service> element. We expect solutions built on WSDL/1.1 to continue to utilize a service element. Moving forward we anticipate that endpoint references and WSDL will evolve coherently. The endpoint references may link to service elements in WSDL/1.1, and support additional scenarios in which the WSDL information is not known by a party processing a message. These scenarios may include dynamic messaging or limited capability message processors.

2.1. Information Model for Endpoint References

An endpoint reference consists of the following abstract properties:

[address] : URI (mandatory)
An address URI that identifies the endpoint. This may be a network address or a logical address.

[reference properties] : xs:any (0..unbounded).
A reference may contain a number of individual properties that are required to identify the entity or resource being conveyed. Reference identification properties are element information items that are named by QName and are required to properly dispatch messages to endpoints at the endpoint side of the interaction. Reference properties are provided by the issuer of the endpoint reference and are otherwise assumed to be opaque to consuming applications. The interpretation of these properties (as the use of the endpoint reference in general) is dependent upon the protocol binding and data encoding used to interact with the endpoint. Section 2.3 below defines the default binding for the SOAP protocol. Consuming applications SHOULD assume that endpoints represented by endpoint references with different [reference properties] may accept different sets of messages or follow a different set of policies, and consequently may have different associated metadata (WSDL, XML Schema, and WS-Policy policies). The relationship between reference properties and endpoint policies is further explained in Section 2.4.

[reference parameters] : xs:any (0..unbounded).
A reference may contain a number of individual parameters which are associated with the endpoint to facilitate a particular interaction. Reference parameters are element information items that are named by QName and are required to properly interact with the endpoint. Reference parameters are also provided by the issuer of the endpoint reference and are otherwise assumed to be opaque to consuming applications. The use of reference parameters is dependent upon the protocol binding and data encoding used to interact with the endpoint. Section 2.3 describes the default binding for the SOAP protocol. Unlike [reference properties], the [reference parameters] of two endpoint references may differ without an implication that different XML Schema, WSDL or policies apply to the endpoints.

[selected port type] : QName (0..1)
The QName of the primary portType of the endpoint being conveyed.

[service-port] : (QName, NCName (0..1)) (0..1)
This is the QName identifying the WSDL service element that contains the definition of the endpoint being conveyed. The service name provides a link to a full description of the service endpoint. An optional non-qualified name identifies the specific port in the service that corresponds to the endpoint.

[policy] : wsp:policy (0..unbounded)
A variable number of XML policy elements as described in WS-Policy [WS-Policy] describing the behavior, requirements and capabilities of the endpoint. Policies may be included in an endpoint to facilitate easier processing by the consuming application, or because the policy was dynamically generated. However, embedded policies are not authoritative and may be stale or incoherent with the policies associated with the endpoint at the time when the interaction occurs.

2.2. Endpoint Reference XML Infoset Representation

This section defines an XML Infoset-based representation for an endpoint reference as both an XML type (`wsa:EndpointReferenceType`) and as an XML element (`<wsa:EndpointReference>`).

The `wsa:EndpointReferenceType` type is used wherever a Web service endpoint is referenced. The following describes the contents of this type:

```
<wsa:EndpointReference>
    <wsa:Address>xs:anyURI</wsa:Address>
    <wsa:ReferenceProperties>... </wsa:ReferenceProperties> ?
    <wsa:ReferenceParameters>... </wsa:ReferenceParameters> ?
    <wsa:PortType>xs:QName</wsa:PortType> ?
    <wsa:ServiceName PortName="xs:NCName"?>xs:QName</wsa:ServiceName> ?
    <wsp:Policy> ... </wsp:Policy>*
</wsa:EndpointReference>
```

The following describes the attributes and elements listed in the schema overview above:

/wsa:EndpointReference
: This represents some element of type `wsa:EndpointReferenceType`. This example uses the predefined `<wsa:EndpointReference>` element, but any element of type wsa:EndpointReferenceType may be used.

/wsa:EndpointReference/wsa:Address
: This REQUIRED element (of type `xs:anyURI`) specifies the [address] property of the endpoint reference. This address may be a logical address or identifier for the service endpoint.

/wsa:EndpointReference/wsa:ReferenceProperties/
: This OPTIONAL element contains the elements that convey the [reference properties] of the reference.

/wsa:EndpointReference/wsa:ReferenceProperties/{any}
: Each child element of ReferenceProperties represents an individual [reference property].

/wsa:EndpointReference/wsa:ReferenceParameters/
: This OPTIONAL element contains the elements that convey the [reference parameters] of the reference.

/wsa:EndpointReference/wsa:ReferenceParameters/{any}
: Each child element of ReferenceParameters represents an individual [reference parameter].

/wsa:EndpointReference/wsa:PortType
: This OPTIONAL element (of type `xs:Qname`) specifies the value of the [selected port type] property of the endpoint reference.

/wsa:EndpointReference/wsa:ServiceName
: This OPTIONAL element (of type `xs:QName`) specifies the `<wsdl:service>` definition that contains a WSDL description of the endpoint being referenced.

/wsa:EndpointReference/wsa:ServiceName/@PortName
: This OPTIONAL attribute (of type `xs:NCName`) specifies the name of the `<wsdl:port>` definition that corresponds to the endpoint being referenced.

/wsa:EndpointReference/wsp:Policy
: This OPTIONAL element specifies a policy that is relevant to the interaction with the endpoint.

/wsa:EndpointReference/{any}
: This is an extensibility mechanism to allow additional elements to be specified.

/wsa:EndpointReference/@{any}
: This is an extensibility mechanism to allow additional attributes to be specified.

The following illustrates an endpoint reference. This element references the port of type "fabrikam:InventoryPortType" at the URI "http://www.fabrikam123.example/acct".

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
   <wsa:Address>http://www.fabrikam123.example/acct</wsa:Address>
   <wsa:PortType>fabrikam:InventoryPortType</wsa:PortType>
</wsa:EndpointReference>
```

2.3. Binding Endpoint References

When a message needs to be addressed to the endpoint, the information contained in the endpoint reference is mapped to the message according to a transformation that is dependent on the protocol and data representation used to send the message. Protocol-specific mappings (or bindings) will define how the information in the endpoint reference is copied to message and protocol fields. This specification defines the SOAP binding for endpoint references. This mapping MAY be explicitly replaced by other bindings (defined as WSDL bindings or as policies); however, in the absence of an applicable policy stating that a different mapping must be used, the SOAP binding defined here is assumed to apply. To ensure interoperability with a broad range of devices, all conformant implementations MUST support the SOAP binding.

The SOAP binding for endpoint references is defined by the following two rules:

- The [address] property in the endpoint reference is copied in the [destination] header field of the SOAP message.
- Each [reference property] and [reference parameter] element becomes a header block in the SOAP message. The element information item of each [reference property] or [reference parameter] (including all of its [children], [attributes] and [in-scope namespaces]) is to be added as a header block in the new message.

The next example shows how the default SOAP binding for endpoint references is used to construct a message addressed to the endpoint:

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:fabrikam="...">
   <wsa:Address>http://www.fabrikam123.example/acct</wsa:Address>
   <wsa:ReferenceProperties>
       <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
   </wsa:ReferenceProperties>
   <wsa:ReferenceParameters>
       <fabrikam:ShoppingCart>ABCDEFG</fabrikam:ShoppingCart>
   </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

According to the mapping rules stated before, the address value is copied in the "To" header and the "CustomerKey" element should be copied literally as a header in a SOAP message addressed to this endpoint. The SOAP message would look as follows:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
         xmlns:wsa="..." xmlns:fabrikam="... ">
   <S:Header>
     ...
    <wsa:To>http://www.fabrikam123.example/acct</wsa:To>
    <fabrikam:CustomerKey>123456789</fabrikam:CustomerKey>
    <fabrikam:ShoppingCart>ABCDEFG</fabrikam:ShoppingCart>
     ...
   </S:Header>
   <S:Body>
     ...
   </S:Body>
</S:Envelope>
```

2.4 Endpoint Reference Comparison

During the course of Web services interactions applications may receive multiple endpoint references describing the endpoints it needs to interact with. Different copies of an endpoint reference may also be received over time.

The following rules clarify the relation between the behaviors of the endpoints represented by two endpoint references with the same [address] and the same [reference properties].

- The two endpoints accept the same sets of messages, and follow and require the same set of policies. That is, the XML Schema, WSDL, and WS-Policy metadata applicable to the two references are the same.
- In particular, the policies applicable to the two endpoints are the same regardless of the values of the embedded [policy]. Embedded policies are not authoritative and may be stale or incoherent with the policies associated with the endpoint.

The [address] properties of two endpoint references are compared according to Section 6 of [RFC 2396]. The [reference properties] of two endpoint references are equal if:

- they contain the same number of individual properties;
- for each reference property in one endpoint reference there exists an equivalent

reference property in the other. One [reference property] is equivalent to another [reference property] if their byte streams per Exclusive XML Canonicalization are equal.

Therefore, a consuming application should assume that different XML Schemas, WSDL definitions and policies apply to endpoint references whose address or reference properties differ.

3. Message Information Headers

This section defines the model and syntax of a message information header.

The message information headers collectively augment a message with the following abstract properties. These properties enable the identification and location of the endpoints involved in an interaction. The basic interaction pattern from which all others are composed is "one way". In this pattern a source sends a message to a destination without any further definition of the interaction.

"Request Reply" is a common interaction pattern that consists of an initial message sent by a source endpoint (the request) and a subsequent message sent from the destination of the request back to the source (the reply). A reply can be either an application message, a fault, or any other message.

The properties below support one way, request reply, and any other interaction pattern:

[destination] : URI (mandatory)
The address of the intended receiver of this message.

[source endpoint] : endpoint reference (0..1)
Reference of the endpoint where the message originated from.

[reply endpoint] : endpoint reference (0..1)
An endpoint reference that identifies the intended receiver for replies to this message. If a reply is expected, a message MUST contain a [reply endpoint]. The sender MUST use the contents of the [reply endpoint] to formulate the reply message as defined in Section 3.2. If the [reply endpoint] is absent, the contents of the [source endpoint] may be used to formulate a message to the source. This property MAY be absent if the message has no meaningful reply. If this property is present, the [message id] property is REQUIRED.

[fault endpoint] : endpoint reference (0..1)
An endpoint reference that identifies the intended receiver for faults related to this message. When formulating a fault message as defined in Section 3.2 and 4, the sender MUST use the contents of the [fault endpoint] of the message being replied to to formulate the fault message. If the [fault endpoint] is absent, the sender MAY use the contents of the [reply endpoint] to formulate the fault message. If both the [fault endpoint] and [reply endpoint] are absent, the sender MAY use the contents of the [source endpoint] to formulate the fault message. This property may be absent if the sender cannot receive fault messages (e.g., is a one-way application message). If this property is present, the [message id] property is REQUIRED.

[action] : URI (mandatory)

An identifier that uniquely (and opaquely) identifies the semantics implied by this message.

It is RECOMMENDED that value of the [action] property is a URI identifying an input, output, or fault message within a WSDL port type. An action may be explicitly or implicitly associated with the corresponding WSDL definition. Section 3.3 below describes the mechanisms of association. Finally, if in addition to the [action] property, a SOAP Action URI is encoded in a request, the URI of the SOAP Action MUST be the same as the one specified by the [action] property.

[message id] : URI (0..1)

A URI that uniquely identifies this message in time and space. No two messages with a distinct application intent may share a [message id] property. A message MAY be retransmitted for any purpose including communications failure and MAY use the same [message id] property. The value of this property is an opaque URI whose interpretation beyond equivalence is not defined in this specification. If a reply is expected, this property MUST be present.

[relationship] : (QName, URI) (0..unbounded)

A pair of values that indicate how this message relates to another message. The type of the relationship is identified by a QName. The related message is identified by a URI that corresponds to the related message's [message id] property. The message identifier URI may refer to a specific message, or be the following well-known URI that means "unspecified message":

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/id/unspecified
```

This specification has one predefined relationship type:

QName	Description
wsa:Reply	Indicates that this is a reply to the message identified by the URI.

A reply message MUST contain a [relationship] property consisting of wsa:Reply and the message id property of the request message.

The dispatching of incoming messages is based on two message properties. The mandatory "destination" and "action" fields identify the target processing location and the verb or intent of the message.

Due to the range of network technologies currently in wide-spread use (e.g., NAT, DHCP, firewalls), many deployments cannot assign a meaningful global URI to a given endpoint. To allow these "anonymous" endpoints to initiate message exchange patterns and receive replies, WS-Addressing defines the following well-known URI for use by endpoints that cannot have a stable, resolvable URI.

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
```

Requests whose [reply endpoint], [source endpoint] and/or [fault endpoint] use this address MUST provide some out-of-band mechanism for delivering replies or faults (e.g. returning the reply on the same transport connection). This mechanism may be a simple request/reply transport protocol (e.g., HTTP GET or POST). This URI MAY be used as the [destination] for reply messages and SHOULD NOT be used as the [destination] in other circumstances.

3.1. Message Information Headers XML Infoset Representation

The message information header blocks provide end-to-end characteristics of a message that can be easily secured as a unit. The information in these headers is immutable and not intended to be modified along the message path.

The following describes the contents of the message information header blocks:

```
<wsa:MessageID> xs:anyURI </wsa:MessageID>
<wsa:RelatesTo RelationshipType="..."?>xs:anyURI</wsa:RelatesTo>
<wsa:To>xs:anyURI</wsa:To>
<wsa:Action>xs:anyURI</wsa:Action>
<wsa:From>endpoint-reference</wsa:From>
<wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
<wsa:FaultTo>endpoint-reference</wsa:FaultTo>
```

The following describes the attributes and elements listed in the schema overview above:

/wsa:MessageID
: This OPTIONAL element (of type `xs:anyURI`) conveys the [message id] property. This element MUST be present if wsa:ReplyTo or wsa:FaultTo is present.

/wsa:RelatesTo
: This OPTIONAL (repeating) element information item contributes one abstract [relationship] property value, in the form of a (URI, QName) pair. The [children] property of this element (which is of type `xs:anyURI`) conveys the [message id] of the related message. This element MUST be present if the message is a reply.

/wsa:RelatesTo/@RelationshipType
: This OPTIONAL attribute (of type `xs:QName`) conveys the relationship type as a QName. When absent, the implied value of this attribute is wsa:Reply.

/wsa:ReplyTo
: This OPTIONAL element (of type `wsa:EndpointReferenceType`) provides the value for the [reply endpoint] property. This element MUST be present if a reply is expected. If this element is present, wsa:MessageID MUST be present.

/wsa:From
: This OPTIONAL element (of type `wsa:EndpointReferenceType`) provides the value for the [source endpoint] property.

/wsa:FaultTo
: This OPTIONAL element (of type `wsa:EndpointReferenceType`) provides the

value for the [fault endpoint] property. If this element is present, wsa:MessageID MUST be present.

/wsa:To
: This REQUIRED element (of type `xs:anyURI`) provides the value for the [destination] property.

/wsa:Action
: This REQUIRED element of type `xs:anyURI` conveys the [action] property. The [children] of this element convey the value of this property.

3.2 Formulating a Reply Message

The reply to a WS-Addressing compliant request message MUST be compliant to WS-Addressing and be constructed according to the rules defined in this section.

The following example illustrates a request message using message information header blocks in a SOAP 1.2 message:

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:f123="http://www.fabrikam123.example/svc53">
  <S:Header>
  <wsa:MessageID>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
    </wsa:MessageID>
    <wsa:ReplyTo>
    <wsa:Address>http://business456.example/client1</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To S:mustUnderstand="1">mailto:joe@fabrikam123.example</wsa:To>
    <wsa:Action>http://fabrikam123.example/mail/Delete</wsa:Action>
  </S:Header>
  <S:Body>
    <f123:Delete>
       <maxCount>42</maxCount>
    </f123:Delete>
  </S:Body>
</S:Envelope>
```

This message would have the following property values:

[destination] The URI mailto:joe@fabrikam123.example

[reply endpoint] The endpoint with [address] http://business456.example/client1

[action] http://fabrikam123.example/mail/Delete

[message id] uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff

The following example illustrates a reply message using message information header blocks in a SOAP 1.2 message:

```
<S:Envelope
  xmlns:S="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:f123="http://www.fabrikam123.example/svc53">
  <S:Header>
    <wsa:MessageID>
      uuid:aaaabbbb-cccc-dddd-eeee-wwwwwwwwww
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
    </wsa:RelatesTo>
    <wsa:To S:mustUnderstand="1">
      http://business456.example/client1
    </wsa:To>
    <wsa:Action>http://fabrikam123.example/mail/DeleteAck</wsa:Action>
  </S:Header>
  <S:Body>
    <f123:DeleteAck/>
  </S:Body>
</S:Envelope>
```

This message would have the following property values:

[destination] http://business456.example/client1

[action] http://fabrikam123.example/mail/DeleteAck

[message id] uuid:aaaabbbb-cccc-dddd-eeee-wwwwwwwww

[relationship] (wsa:Reply, uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff)

3.3 Associating Action with WSDL Operations

WS-Addressing defines two mechanisms to associate an action with input, output and fault elements within a WSDL port type.

3.3.1 Explicit Association

The action may be explicitly associated using the wsa:Action attribute or in the absence of the attribute the action is defined by the rule in section 3.3.2.

For example consider the following WSDL excerpt:

```
<definitions targetNamespace="http://example.com/stockquote" ...>
  ...
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetTradePricesInput"
            wsa:Action="http://example.com/GetQuote"/>
```

```
      <output message="tns:GetTradePricesOutput"
            wsa:Action="http://example.com/Quote"/>
    </operation>
  </portType>
  ...
</definitions>
```

The action for the input of the `GetLastTradePrice` operation within the `StockQuotePortType` is explicitly defined to be `http://example.com/GetQuote`. The action for the output of this same operation is `http://example.com/Quote`.

3.3.2 Default Action Pattern

In the absence of the `wsa:Action` attribute, the following pattern is used to construct a default action for inputs and outputs. The general form of an action URI is as follows:

```
[target namespace]/[port type name]/[input|output name]
```

The "/" is a literal character to be included in the action. The values of the properties are as defined below.

[target namespace] is the target namespace (/definition/@targetNamespace). If [target namespace] ends with a "/" an additional "/" is not added.

[port type name] is the name of the port type (/definition/portType/@name).

[input|output name] is the name of the element as defined in Section 2.4.5 of WSDL 1.1.

For fault messages, this pattern is not applied. Instead, the following URI is the default action URI for fault messages:

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
```

For example consider the following WSDL excerpt:

```
<definitions targetNamespace="http://example.com/stockquote" ...>
  ...
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetTradePricesInput" name="GetQuote"/>
      <output message="tns:GetTradePricesOutput" name="Quote"/>
    </operation>
  </portType>
  ...
</definitions>
```

[targetNamespace] = http://example.com/stockquote

[port type name] = StockQuotePortType

[input name] = GetQuote

[output name] = Quote

Applying the pattern above with these values we have:

input action = http://example.com/stockquote/StockQuotePortType/GetQuote

output action = http://example.com/stockquote/StockQuotePortType/Quote

WSDL defines rules for a default input or output name if the name attribute is not present. Consider the following example:

For example consider the following WSDL excerpt:

```
<definitions targetNamespace="http://example.com/stockquote" ...>
  ...
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetTradePricesInput"/>
      <output message="tns:GetTradePricesOutput"/>
    </operation>
  </portType>
  ...
</definitions>
```

[targetNamespace] = http://example.com/stockquote

[port type name] = StockQuotePortType

According to the rules defined in WSDL 2.4.5, if the name attribute is absent for the input of a request response operation the default value is the name of the operation "Request" appended.

[input name] = GetLastTradePriceRequest

Likewise, the output defaults to the operation name with "Response" appended.

[output name] = GetLastTradePriceResponse

Applying the pattern above with these values we have:

input action = http://example.com/stockquote/StockQuotePortType /GetLastTradePriceRequest

output action = http://example.com/stockquote/StockQuotePortType /GetLastTradePriceResponse

4. Faults

The faults defined in this section are generated if the condition stated in the preamble in each subsection is met. They are sent to the [fault endpoint], if present and valid. Otherwise they are sent to the [reply endpoint] if present. If neither is present faults may be sent to the [source endpoint].

Endpoints compliant with this specification MUST include required message information headers on all fault messages. Fault messages are correlated as replies using the [relationship] property as defined in Section 3. The [action] property below designates WS-Addressing fault messages (this URI is also used as default Action value for WSDL fault messages, as described in Section 3.3.2):

```
http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
```

The definitions of faults use the following properties:

[Code] The fault code.

[Subcode] The fault subcode.

[Reason] The English language reason element.

[Detail] The detail element. If absent, no detail element is defined for the fault.

The properties above bind to a SOAP 1.2 fault as follows:

```
<S:Envelope>
 <S:Header>
   <wsa:Action>
     http://schemas.xmlsoap.org/ws/2004/08/addressing/fault
   </wsa:Action>
   <!-- Headers elided for clarity.  -->
 </S:Header>
 <S:Body>
  <S:Fault>
   <S:Code>
    <S:Value>[Code]</S:Value>
     <S:Subcode>
    <S:Value>[Subcode]</S:Value>
     </S:Subcode>
   </S:Code>
   <S:Reason>
     <S:Text xml:lang="en">[Reason]</S:Text>
   </S:Reason>
   <S:Detail>
     [Detail]
  </S:Detail>
  </S:Fault>
 </S:Body>
</S:Envelope>
```

The SOAP 1.1 fault is less expressive and map only [Subcode] and [Reason]. These the properties bind to a SOAP 1.1 fault as follows:

```
<S11:Envelope>
 <S11:Body>
```

```
  <S11:Fault>
   <faultcode>[Subcode]</faultcode>
   <faultstring xml:lang="en">[Reason]</faultstring>
  </S11:Fault>
 </S11:Body>
</S11:Envelope>
```

4.1 Invalid Message Information Header

A message information header cannot be processed.

[Code] S:Sender

[Subcode] wsa:InvalidMessageInformationHeader

[Reason] A message information header is not valid and the message cannot be processed. The validity failure can be either structural or semantic, e.g. a [destination] that is not a URI or a [relationship] to a [message id] that was never issued.

[Detail] [invalid header]

4.2 Message Information Header Required

A required message information header is absent.

[Code] S:Sender

[Subcode] wsa:MessageInformationHeaderRequired

[Reason] A required message information header, To, MessageID, or Action, is not present.

[Detail] [Missing Header QName]

4.3 Destination Unreachable

The no endpoint can be found capable of acting in the role of the [destination] property.

[Code] S:Sender

[Subcode] wsa:DestinationUnreachable

[Reason] No route can be determined to reach the destination role defined by the WS-Addressing To.

[Detail] empty

4.4 Action Not Supported

The [action] property in the message is not supported at this endpoint.

The contents of this fault are as follows:

[Code] S:Sender

[Subcode] wsa:ActionNotSupported

[Reason] The [action] cannot be processed at the receiver.

[Detail] [action]

4.5 Endpoint Unavailable

The endpoint is unable to process the message at this time either due to some transient issue or a permanent failure.

The endpoint may optionally include a RetryAfter parameter in the detail. The source should not retransmit the message until this duration has passed.

[Code] S:Receiver

[Subcode] wsa:EndpointUnavailable

[Reason] The endpoint is unable to process the message at this time.

[Detail] `<wsa:RetryAfter ...>[xs:NonNegativeInteger]</wsa:RetryAfter>`

The following describes the attributes and elements listed above:

/wsa:RetryAfter
: This element (of type `xs:NonNegativeInteger`) is a suggested minimum duration in milliseconds to wait before retransmitting the message. If this element is omitted from the detail, the value is infinite.

/wsa:RetryAfter/@{any}
: These optional extensibility attributes do not affect processing.

5. Security Considerations

It is strongly recommended that the communication between services be secured using the mechanisms described in WS-Security [WS-Security]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the message information headers described in this specification (e.g. <wsa:To>) need to be signed with the body in order to "bind" the two together. It should be noted that for messages traveling through intermediaries, it is possible that some or all of the message information headers may have multiple signatures when the message arrives at the ultimate receiver. It is strongly recommended that the initial sender include a signature to prevent any spoofing by intermediaries.

Whenever an address is specified (e.g. <wsa:From>, <wsa:ReplyTo>, <wsa:FaultTo>, ...), the processor should ensure that a signature is provided with claims allowing it to

speak for the specified target in order to prevent certain classes of attacks (e.g. redirects). As well, care should be taken if the specified endpoint contains resource properties or parameters as unverified endpoint references could cause certain classes of header insertion attacks.

The message information headers blocks may have their contents encrypted in order to obtain end-to-end privacy, but care should be taken to ensure that intermediary processors have access to required information (e.g. <wsa:To>).

Some processors may use message identifiers (<wsa:MessageID>) as part of a uniqueness metric in order to detect replays of messages. Care should be taken to ensure that a unique identifier is actually used. For example, it may be appropriate in some scenarios to combine the message identifier with a timestamp.

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.
- Address spoofing – Address spoofing is prevented by ensuring that all address are signed by a party authorized to speak for (or on behalf of) the address.
- Key integrity – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy [WS-Policy] and WS-SecurityPolicy [WS-SecurityPolicy]).
- Authentication – Authentication is established using the mechanisms described in WS-Security and WS-Trust [WS-Trust]. Each message is authenticated using the mechanisms described in WS-Security.
- Accountability – Accountability is a function of the type of and strength 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 and/or caching of message identifiers. 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.
- Replay – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

6. Acknowledgements

Keith Ballinger, Microsoft; Michael Coulson, Microsoft; Giovanni Della-Libera, Microsoft; Christopher Ferris, IBM; Tom Freund, IBM; Steve Graham, IBM; Christoph Hofmann, SAP; Maryann Hondo, IBM; Efim Hudis, Microsoft; John Ibbotson, IBM; Gopal Kakivaya, Microsoft; Al Lee, Microsoft; Anthony Nadalin, IBM; Bill Nagy, IBM; Martin Nally, IBM; Henrik Frystyk Nielsen, Microsoft; Vladimir Savchenko, SAP; Jeffrey Schlimmer, Microsoft; Chris Sharp, IBM; Keith Stobie, Microsoft; Vladimir Videlov, SAP; Volker Wiechers, SAP; Hervey Wilson, Microsoft;

7. References

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

[RFC 2396]
T. Berners-Lee, et al, "Uniform Resource Identifier (URI): Generic Syntax," RFC 2396bis, W3C/MIT, July 2004.

[XML 1.0]
W3C Recommendation "Extensible Markup Language (XML) 1.0 (Second Edition)", Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, 6 October 2000

[XML Namespaces]
W3C Recommendation "Namespaces in XML", Tim Bray, Dave Hollander, Andrew Layman, 14 January 1999

[XML Infoset]
W3C Recommendation "XML Information Set", John Cowan, Richard Tobin, 24 October 2001

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," May 2001.

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

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001.

[WS-Security]
OASIS, " Web Services Security: SOAP Message Security", March 2004.

[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language

(WS-SecurityPolicy)," December 2002.

[WS-Trust]
S. Anderson, et al, "Web Services Trust Language (WS-Trust)," May 2004.

[WS-Policy]
D. Box, et al, "Web Services Policy Framework (WS-Policy)," May 2003.