Proposed Infoset Addendum to SOAP Messages with Attachments

Version 0.61 Draft

1 April 2003

Authors

Adam Bosworth, BEA
Don Box, Microsoft
Martin Gudgin, Microsoft
Mark Jones, AT&T
Franz-Josef Fritz, SAP
Amy Lewis, Tibco
Jean-Jacques Moreau, Canon
Mark Nottingham, BEA
David Orchard, BEA
Hervé Ruellan, Canon
Jeffrey Schlimmer, Microsoft
Volker Wiechers, SAP
TBD

Copyright Notice

OR ANY OTHER AGREEMENT RELATING TO THIS DOCUMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Abstract

This specification defines a small number of XML and SOAP conventions that clarify an earlier proposal and collectively allow opaque data and web references to be used in an Infoset-based messaging model.

Status

This specification is provided as-is and for review and evaluation only. AT&T, BEA, Canon, Microsoft, SAP, and Tibco hope to solicit your contributions and suggestions in the near future. AT&T, BEA, Canon, Microsoft, SAP, and Tibco make no warrantees or representations regarding the specification in any manner whatsoever.

Table of Contents

1. Introduction

The desire to integrate XML [XML] with pre-existing data formats has been a long-standing and persistent issue for the XML community. Users often want to leverage the structured, extensible markup conventions of XML without abandoning existing data formats that do not readily adhere to XML 1.0 syntax. Often, users want to leave their existing non-XML formats as is, to be treated as opaque sequences of octets by XML tools and infrastructure. Such an approach would allow widely used formats such as JPEG and WAV to peacefully coexist with XML.

As XML is increasingly used as a message format (e.g., SOAP [SOAP11, SOAP12]), the interest in integrating opaque data with XML has increased to the point where there are at least two concrete proposals for doing so: SOAP Messages with Attachments 1.0 [SWA1] and WS-Attachments [WSA]. The former has gained some traction within the community but is under-specified with respect to the XML Infoset [Infoset] and with respect to the processing model of SOAP. (See [InfosetWP] for details.)

This document proposes a set of concrete idioms and conventions that build on SOAP Messages with Attachments, yielding the following enhancements:

- Alignment with the XML Infoset-based data model and the SOAP processing model – opaque data may be correctly processed by intermediaries and may be secured
- Backwards-compatible message syntax – every message conforming to this proposal is a legal SwA/1.0 message
- Alternate message syntax for SOAP processors that have no knowledge of SwA or this proposal – message content can be faithfully serialized in a form that is understandable by SOAP processors that do not comply with this specification

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 Namespaces

The XML namespace URI that MUST be used by implementations of this specification are:

```
http://schemas.xmlsoap.org/2003/04/swa
http://schemas.xmlsoap.org/2003/04/xbinc
http://schemas.xmlsoap.org/2003/04/xmime
```

The following namespaces are used in this document:

Prefix	Namespace
soap11	http://schemas.xmlsoap.org/soap/envelope/
soap12	http://www.w3.org/2002/12/soap-envelope

swa	http://schemas.xmlsoap.org/2003/04/swa
xbinc	http://schemas.xmlsoap.org/2003/04/xbinc
xmime	http://schemas.xmlsoap.org/2003/04/xmime
xs	http://www.w3.org/2001/XMLSchema

3. Using Media Types in XML

Opaque data whose native representation is a sequence of octets may be encoded as base64 [base64] text in XML [XML] elements without loss of information. However, the industry has invested heavily in the MIME Content-Type type [RFC 2045] system for annotating the expected format (if not interpretation) of raw octet sequences. This information is not captured in today's XML Schema [XMLSchema2] type xs:base64Binary. (This specification refers to the xs:base64Binary data type that is defined in Part II of XML Schema [XMLSchema2]. This reference in no way mandates XML Schema processing or description of XML instances that use this specification.)

This specification defines a global attribute (xmime:MediaType) that may be applied to elements whose children contain base64-encoded binary data. This specification also defines an XML Schema [XMLSchema1] complexType that augment the xs:base64Binary type with this attribute.

3.1 xmime:MediaType attribute

The MediaType attribute specifies the media type [RFC 2045] of the base64-encoded content of its [owner] element. Its normalized value is a media type as defined by Section 5.1 of RFC 2045 and RFC 2046 [RFC 2046]. When the MediaType attribute is not present the media type "application/octet-stream" is assumed.

3.2 xmime:Binary type

The Binary type is an XML Schema complexType whose base is xs:base64Binary. The type carries optional xmime:MediaType attribute. This type can be used by elements that need to carry base64-encoded data along with optional media type information.

3.3 Example

In the following example, the m:photo, m:sound, and m:sig elements are of type xmime:Binary. The xmime:MediaType attribute defined for that type labels the MIME type of the base64-encoded content for each of these elements. Note that this message may be correctly processed by a SOAP node that does not explicitly comply with this document.

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff' >
      <m:photo xmime:MediaType='image/png' >
        /aWKKapGGyQ=
      </m:photo>
      <m:sound xmime:MediaType='audio/mpeg' >
        sdcfo2JTiXE=
```

```
      </m:sound>
      <m:sig xmime:MediaType='application/pkcs7-signature' >
        Faa7vROi2VQ=
      </m:sig>
    </m:data>
  </soap:Body>
</soap:Envelope>
```

4. Incorporating External Data into the SOAP Envelope

For many applications, the use of base64 [base64] encoding for opaque data does not present a significant performance overhead, especially when weighed against the costs of a conformant XML 1.0 [XML] parser. However, for applications that wish to avoid the overhead of base64 encoding, this specification defines an XML element (xbinc:Include) that can reference opaque data for inclusion as children of the referencing element. The opaque data is referenced by a URI, and the resultant base64-ized version of the octet sequence *logically* replaces the xbinc:Include element. This allows an XML processor to behave as if all binary data is base64-encoded character content within the document, independent of its wire format, allowing the processor to apply an Infoset-based processing model to the document.

In the course of *actual* processing, the replacement MAY be implemented by brute-force conversion of the raw octets to base64 or it MAY avoid the explicit conversion to base64 characters. The degree to which a given implementation elects to optimize this style of access is completely implementation-specific. That stated, it is trivial to implement a brute force conversion technique as a SAX or System.Xml.XmlReader filter in Java or C# (respectively). Implementing a model in which the base64 conversion is bypassed is also relatively straightforward provided the consuming application can explicitly take advantage of such a technique.

The behavior of the xbinc:Include element is closely related to the behavior of the include element defined in XInclude 1.0 [XInclude]; if the latter specification is extended to enable including binary content (perhaps by defining a suitable value for the parse attribute), this specification should reference it.

The specification also defines an xbinc:DoInclude header element which controls Include processing.

4.1 xbinc:Include element

The Include element is used to reference opaque data for logical inclusion. The Include element carries a single attribute. xbinc:Include MUST NOT be a child of, but MAY be a descendant of, soap11:Envelope, soap11:Header, soap11:Body, soap12:Envelope, soap12:Header, or soap12:Body.

4.1.1 href attribute

The href attribute provides the URI of the opaque data to be included. The normalized value of the href attribute MUST resolve to a resource within the message serialization. A base64-encoding of the octet stream resulting from resolving the URI replaces the Include element that the URI attribute appears on.

4.2 xbinc:DoInclude element

The xbinc:DoInclude SOAP header block indicates that messages SHOULD be processed for xbinc:Include elements. The xbinc:DoInclude header block MUST be included if any of the descendants of the SOAP Envelope are xbinc:Include elements.

For SOAP 1.1,

- The mustUnderstand attribute, if present, MUST have a normalized value of “0”
- The actor attribute MUST have a normalized value of “http://schemas.xmlsoap.org/soap/actor/next”

For SOAP 1.2,

- The mustUnderstand attribute, if present, MUST have a normalized value of “false” or "0"
- The role attribute MUST have a normalized value of “http://www.w3.org/2002/12/soap-envelope/role/next”
- The relay attribute MUST have a normalized value of “true”

If a SOAP intermediary forwards a SOAP message to another SOAP node, the intermediary MUST re-insert the xbinc:DoInclude header block without change with respect to the signature canonicalization algorithm in use (if any).

This header block is invoked upon access; it SHOULD be invoked in the processing model before other header blocks that reference or manipulate the data within. As a result, a naïve implementation MAY just invoke xbinc:Include once at the start of message processing, whilst a more sophisticated implementation MAY dereference the included data “lazily”, that is, only upon access.

Note that because xbinc:Include elements cannot be children of soap11:Header or soap12:Header, a SOAP node MAY perform Step 4 of the SOAP processing model, i.e., process mandatory SOAP header blocks, without first processing xbinc:Include elements.

In some cases, ordering of header block processing becomes important; this document does not define a means to order processing but expects other mechanisms will address this need. In the absence of such information, the xbinc:DoInclude header block SHOULD be processed before any header blocks that access parts of the Envelope that contain xbinc:Include elements.

4.3 FatalIncludeFault

If the xbinc:Include processor encounters a Fatal Error, the FatalIncludeFault is generated.

4.4 Include example

The following example illustrates the use of xbinc:Include in a multipart MIME [RFC 2387] serialization. Note that while in this example all opaque data is carried in the multipart MIME packaging, this is not an intrinsic characteristic of include processing, and Include elements could conceivably be used with other message serialization schemes.

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
start="<mymessage.xml@example.org>"
```

```
Content-Description: An XML document with my pic, warcry and sig in it

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
  </soap:Header>
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff' >
      <m:photo xmime:MediaType='image/png' >
        <xbinc:Include href='cid:http://example.org/me.png' />
      </m:photo>
      <m:sound xmime:MediaType='audio/mpeg' >
        <xbinc:Include href='cid:http://example.org/it.mp3' />
      </m:sound>
      <m:sig xmime:MediaType='application/pkcs7-signature' >
        <xbinc:Include href='cid:http://example.org/my.hsh' />
      </m:sig>
    </m:data>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/me.png>

fd a5 8a 29 aa 46 1b 24

--MIME_boundary
Content-Type: audio/mpeg
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/it.mp3>
```

```
b1 d7 1f a3 62 53 89 71

--MIME_boundary
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/my.hsh>

15 a6 bb bd 13 a2 d9 54

--MIME_boundary--
```

The resultant Infoset is the same as that of the following:

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
  </soap:Header>
  <soap:Body>
    <m:data xmlns:m='http://example.org/stuff' >
      <m:photo xmime:MediaType='image/png' >
        /aWKKapGGyQ=
      </m:photo>
      <m:sound xmime:MediaType='audio/mpeg' >
        sdcfo2JTiXE=
      </m:sound>
      <m:sig xmime:MediaType='application/pkcs7-signature' >
        Faa7vROi2VQ=
      </m:sig>
    </m:data>
  </soap:Body>
</soap:Envelope>
```

5. Web References in the SOAP Envelope

The technique described in Section 3 provided the ability to add MIME type information [RFC 2046] to opaque binary data in XML [XML]. This technique is applicable whether or not the opaque data is in fact associated with a URI-based web reference.

There is one scenario that is not completely satisfied by the technique described in Section 3. That scenario is the M/HTML-esque [RFC 2557] scenario in which the message content contains URI-based web references and the sender wishes to send

the representations behind these references as part of the aggregate message. To facilitate this usage, this specification defines a SOAP [SOAP11, SOAP12] header block (swa:Representation) that allows a SOAP node to send cached representations of web resources to either the ultimate receiver or a specific intermediary.

5.1 swa:Representation element

The Representation element contains base64-encoded [base64] content and carries an href attribute. It also carries an optional xmime:MediaType attribute as defined in Section 3 of this specification.

The content of the element is the base64-encoding of the web resource referred to by the URI attribute. If this representation is appropriately secured (see Section 8), applications that resolve URIs MUST use this representation of the web resource. Specifically, when a URI is dereferenced, the contents of the Representation element with the matching URI attribute value MUST be used as the representation returned if it is appropriately secured. However, note that this does not offer all of the functionality of HTTP caching and content negotiation mechanisms.

The Representation element MAY be used as a header block and/or as a body block. As a header block, the Representation element MAY also carry soap11:mustUnderstand and/or soap11:actor attributes per the SOAP 1.1 specification [SOAP11] or soap12:mustUnderstand, soap12:role and/or soap12:relay attributes per the SOAP 1.2 specification [SOAP12]; as a SOAP header block, different representations can be targeted at different nodes via the soap11:actor/soap12:role attributes; each node would then be responsible for removing any Representation headers targeted at it as per the SOAP processing model.

5.1.1 URI attribute

The value of the URI attribute specifies the identifier of the Web resource whose base64-encoded representation the Representation element contains.

When comparing URIs [RFC 2396, RFC 2732] to find an appropriate representation:

- If absolute URIs are given, then following RFC 2396, the scheme name SHOULD treat upper-case letters as equivalent to lower case.
- If a scheme appears to follow the "Generic URI" syntax for representing hierarchical relationships and uses a Server-based Naming Authority, then per RFC 1034 [RFC 1034], domain name comparisons are done in a case-insensitive manner, assuming an ASCII character set, and a high order zero bit.

Besides the scheme name (and possibly the domain name), only the lexical form should be considered; that is, they should be compared character-by-character.

5.2 Example of an external resource

In this example, a representation of an external resource, an image, is cached with the SOAP message. The representation of the image is carried in an swa:Representation header block in the SOAP message. The representation is referred to by the src attribute of an img element in the body of the message.

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:swa='http://schemas.xmlsoap.org/2003/04/swa'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
```

```
  <soap:Header>
    <swa:Representation URI='http://example.org/me.png'
                        xmime:MediaType='image/png' >
        /aWKKapGGyQ=
    </swa:Representation>
  </soap:Header>
  <soap:Body>
    <x:MyData xmlns:x='http://example.org/mystuff' >
      <x:name>Don Box</x:name>
      <x:img x:src='http://example.org/me.png' />
    </x:MyData>
  </soap:Body>
</soap:Envelope>
```

Combining this header with the xbinc:Include element described in Section 4 would yield the following serialization:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
start="<mymessage.xml@example.org>"
Content-Description: An XML document with my picture

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:swa='http://schemas.xmlsoap.org/2003/04/swa'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
    <swa:Representation URI='http://example.org/me.png'
                        xmime:MediaType='image/png' >
        <xbinc:Include href='cid:me@example.com' />
    </swa:Representation>
  </soap:Header>
  <soap:Body>
    <x:MyData xmlns:x='http://example.org/mystuff' >
      <x:name>Don Box</x:name>
      <x:img src='http://example.org/me.png' />
```

```
    </x:MyData>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <me@example.com>

fd a5 8a 29 aa 46 1b 24

--MIME_boundary--
```

5.3 Example of multiple references

In this example, a SOAP message references a cached resource in multiple places within the SOAP message. The representation of the resource (an image) is carried in a swa:Representation header block and is referred to by the src attribute of an img and a picture element in the body of the message.

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:swa='http://schemas.xmlsoap.org/2003/04/swa'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Header>
    <swa:Representation URI='http://example.org/me.png'
                        xmime:MediaType='image/png' >
        /aWKKapGGyQ=
    </swa:Representation>
  </soap:Header>
  <soap:Body>
    <x:MyData xmlns:x='http://example.org/mystuff' >
      <x:name>Don Box</x:name>
      <x:img x:src='http://example.org/me.png' />
    </x:MyData>
    <y:AltData xmlns:y='http://example.org/altstuff' >
      <y:name>
        <y:given>Don</y:given>
        <y:surname>Box</y:surname>
      </y:name>
      <y:picture y:src='http://example.org/me.png' />
    </y:AltData>
  </soap:Body>
</soap:Envelope>
```

The representation header block in this example could be combined with an xbinc:Include element (as the previous example in Section 5.2 was) to yield an alternate serialization.

5.4 Example of 'unreferenced' representation

In this example, a SOAP message does not explicitly reference a resource but a cached representation of the resource is included for application processing. The representation of the resource (audio) is carried in a swa:Representation header block.

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:swa='http://schemas.xmlsoap.org/2003/04/swa'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap:Header>
    <swa:Representation URI='http://example.org/alert.mp3'
                        xmime:MediaType='audio/mpeg' >
        sdcfo2JTiXE=
    </swa:Representation>
  </soap:Header>
  <soap:Body>
    <x:MyData xmlns:x='http://example.org/mystuff' >
      <x:name>Don Box</x:name>
    </x:MyData>
  </soap:Body>
</soap:Envelope>
```

The representation header block in this example could be combined with an xbinc:Include element (as the example in Section 5.2 was) to yield an alternate serialization.

6. Processing model

The SOAP [SOAP11, SOAP12] processing model is defined in terms of an Infoset [Infoset]. As defined in Section 4.2, processing MUST behave as if the xbinc:DoInclude header is processed first. SOAP messages containing xbinc:Include elements MUST be treated as if SOAP processing occurs post-inclusion. Thus if a SOAP header block is removed by an intermediary and that header block referred to opaque data via an xbinc:Include element, the opaque data would also be removed from the message.

Since the xbinc:Include element is transfer syntax, if a SOAP intermediary forwards a message, it MAY serialize opaque data in the message Infoset using base64 encoding or using an xbinc:Include element independent of the original message transfer syntax.

6.1 Example

The following message arrives at a security intermediary which acts in the role 'http://schemas.xmlsoap.org/security':

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
```

```
start="<mymessage.xml@example.org>"
Content-Description: A SOAP envelope containing a thumbprint image for
authentication purposes

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime'
               xmlns:m='http://example.org/stuff' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
    <m:Thumbprint xmime:MediaType='image/png'
                  soap:role='http://schemas.xmlsoap.org/security' >
      <xbinc:Include href='cid:http://example.org/thumb.png' />
    </m:Thumbprint>
  </soap:Header>
  <soap:Body>
    <m:sound xmime:MediaType='audio/mpeg' >
      <xbinc:Include href='cid:http://example.org/it.mp3' />
    </m:sound>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/png
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/thumb.png>

fd a5 8a 29 aa 46 1b 24

--MIME_boundary
Content-Type: audio/mpeg
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/it.mp3>

b1 d7 1f a3 62 53 89 71
```

```
--MIME_boundary--
```

The resultant Infoset is the same as that of the following:

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime'
               xmlns:m='http://example.org/stuff' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
    <m:Thumbprint xmime:MediaType='image/png'
                  soap:role='http://schemas.xmlsoap.org/security' >
      /aWKKapGGyQ=
    </m:Thumbprint>
  </soap:Header>
  <soap:Body>
    <m:sound xmime:MediaType='audio/mpeg' >
      sdcfo2JTiXE=
    </m:sound>
  </soap:Body>
</soap:Envelope>
```

After processing by the security intermediary the resultant Infoset is the same as that of the following:

```
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime'
               xmlns:m='http://example.org/stuff' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
  </soap:Header>
  <soap:Body>
    <m:sound xmime:MediaType='audio/mpeg' >
      sdcfo2JTiXE=
    </m:sound>
  </soap:Body>
</soap:Envelope>
```

The security intermediary MAY choose to serialize that Infoset as the following:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
start="<mymessage.xml@example.org>"
Content-Description: A SOAP envelope containing a thumbprint image for
authentication purposes

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope'
               xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
               xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime'
               xmlns:m='http://example.org/stuff' >
  <soap:Header>
    <xbinc:DoInclude
        soap:role='http://www.w3.org/2002/12/soap-envelope/role/next'
        soap:mustUnderstand='false'
        soap:relay='true' />
  </soap:Header>
  <soap:Body>
    <m:sound xmime:MediaType='audio/mpeg' >
      <xbinc:Include href='cid:http://example.org/it.mp3' />
    </m:sound>
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: audio/mpeg
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/it.mp3>

b1 d7 1f a3 62 53 89 71

--MIME_boundary--
```

7. Schema and WSDL Constructs

Applications often have a need to specify a set of acceptable media types for opaque data. To satisfy this need, this specification defines the xmime:Accept which can be used to annotate schema declarations of elements of type xmime:Binary.

7.1 xmime:Accept attribute

The Accept attribute may be used on element declarations in schema to specify a list of accepted media types of the base64-encoded content of instances of the element. Its normalized value is a space-delimited list of media types with “q” parameters as defined in Section 14.1 of [RFC 2616]. When the Accept attribute is not present the media type "*/*" is assumed.

The following WSDL shows an example message that contains an element of type xmime:Binary:

```
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://example.org/CustomerExample"
    xmlns:tns="http://example.org/CustomerExample"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" >

<wsdl:types>
  <xs:schema xmlns="http://www.w3.org/2001/XMLSchema"
             targetNamespace="http://example.org/CustomerExample"
             xmlns:xmime="http://schemas.xmlsoap.org/2003/04/xmime" >
    <xs:import namespace="http://schemas.xmlsoap.org/2003/04/xmime" />
    <xs:element name="Customer" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="xs:string" />
          <xs:element name="Photo"
                      type="xmime:Binary"
                      xmime:Accept="image/jpeg image/gif;p=0.5" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Status" type="xs:string" />
  </xs:schema>
</wsdl:types>

<wsdl:message name="InMessage" >
  <wsdl:part name="InPart" element="tns:Customer" />
</wsdl:message>
<wsdl:message name="OutMessage" >
  <wsdl:part name="OutPart" element="tns:Status" />
</wsdl:message>

<wsdl:portType name="ThePortType" >
  <wsdl:operation name="CustomerInfo" >
```

```
    <wsdl:input message="tns:InMessage" />
    <wsdl:output message="tns:OutMessage" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="SoapBinding" type="tns:ThePortType" >
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="CustomerInfo" >
    <soap:operation
        soapAction="http://example.org/CustomerExample/CustomerInfo"
        style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

</wsdl:definitions>
```

The following is the corresponding SOAP message with contents of Photo serialized using base64 encoding:

```
<soap11:Envelope
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap11:Body>
    <d:Customer xmlns:d="http://example.org/CustomerExample" >
      <d:Name>John Doe</d:Name>
      <d:Photo xmime:MediaType="image/jpeg" >
        /aWKKapGGyQ=
      </d:Photo>
    </d:Customer>
  </soap11:Body>
</soap11:Envelope>
```

Alternatively, the message may use multipart MIME and xbinc:Include as described in Section 4:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
start='<mymessage.xml@example.org>'
Content-Description: A SOAP envelope containing a photo
```

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@example.org>

<soap11:Envelope
    xmlns:soap11='http://schemas.xmls.org/soap/envelope/'
    xmlns:xbinc='http://schemas.xmlsoap.org/2003/04/xbinc'
    xmlns:xmime='http://schemas.xmlsoap.org/2003/04/xmime' >
  <soap11:Header>
    <xbinc:DoInclude
        soap11:actor='http://schemas.xmlsoap.org/soap/actor/next'
        soap11:mustUnderstand='false' />
  </soap11:Header>
  <soap11:Body>
    <d:Customer xmlns:d='http://example.org/CustomerExample' >
      <d:Name>John Doe</d:Name>
      <d:Photo xmime:MediaType='image/jpeg' >
        <xbinc:Include href='cid:http://example.org/Customer.jpg' />
      </d:Photo>
    </d:Customer>
  </soap11:Body>
</soap11:Envelope>

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <http://example.org/Customer.jpg>

fd a5 8a 29 aa 46 1b 24

--MIME_boundary--
```

8. Security Considerations

Given that SOAP processing happens post inclusion, signatures over elements with xbinc:Include children MUST NOT be signatures over the xbinc:Include element and its href attribute; signatures MUST be over the included data. Current XML signature algorithms require signing the included data as base64-encoded characters; the lexical form of such characters SHOULD be canonicalized. (A suitable algorithm may be under development by the XML Schema WG.) However, note that an include-aware canonicalization algorithm may be able to eliminate the need to convert between the raw octets and base64-encoded characters.

In general, it is RECOMMENDED that signatures be against elements and their content (not just the content of elements) to ensure the context is not altered.

Specifically, if the xmime:MediaType attribute is used on an element, then it SHOULD be included in the signature to prevent certain types of attacks.

To ensure that the URI associated with a swa:Representation is not tampered with, the swa:Representation element and its URI attribute SHOULD be signed. References SHOULD be signed by a party who has the right to "speak for" the domain of the reference.

To reduce the risk of denial of service and elevated privilege, senders MUST NOT include and receivers SHOULD discard MIME parts that contain neither the SOAP Envelope nor are referenced by a xbinc:Include from within the SOAP Envelope.

9. References

[base64]
"Base64 Content-Transfer-Encoding," RFC 2045 (Section 6.8), N. Freed and N. Borenstein (editors), November 1996.

[Infoset]
"XML Information Set," W3C Recommendation, John Cowan and Richard Tobin (editors), 24 October 2001.

[InfosetWP]
"XML, SOAP and Binary Data," Adam Bosworth, Don Box, Martin Gudgin, Mark Nottingham, David Orchard, and Jeffrey Schlimmer, 26 February 2003.

[RFC 1034]
"Domain Names - Concepts and Facilities," RFC 1034, P. Mockapetris (editor), November 1987.

[RFC 2045]
"Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," RFC 2045, N. Freed and N. Borenstein (editors), November 1996.

[RFC 2046]
"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," RFC 2046, N. Freed and N. Borenstein (editors), November 1996.

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

[RFC 2387]
"The MIME Multipart/Related Content-type," RFC 2387, E. Levinson (editor), August 1998.

[RFC 2396]
"Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, T. Berners-Lee, R. Fielding, and L. Masinter (editors), August 1998.

[RFC 2557]
"MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)," RFC 2557, J. Palme, A. Hopmann, and N. Shelness (editors), March 1999.

[RFC 2616]
"Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee (editors), June 1999.

[RFC 2732]
"Format for Literal IPv6 Addresses in URL's," RFC 2732, R. Hinden, B. Carpenter, and L. Masinter (editors), December 1999.

[SOAP11]
"Simple Object Access Protocol (SOAP) 1.1," W3C Note, Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer, 8 May 2000.

[SOAP12]
"SOAP Version 1.2 Part 1: Messaging Framework," W3C Candidate Recommendation, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen (editors), 19 December 2002.

[SWA1]
"SOAP Messages with Attachments," W3C Note, John J. Barton, Satish Thatte, and Henrik Frystyk Nielsen, 11 December 2000.

[XInclude]
"XML Inclusions (XInclude) Version 1.0," W3C Candidate Recommendation, Jonathan Marsh and David Orchard (editors), 17 September 2002.

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

[XMLSchema1]
"XML Schema Part 1: Structures," W3C Recommendation, Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn (editors), 2 May 2001.

[XMLSchema2]
"XML Schema Part 2: Datatypes," W3C Recommendation, Paul V. Biron and Ashok Malhotra (editors), 2 May 2001.

[WSA]
"WS-Attachments," IETF Draft, Henrik Frystyk Nielsen, Erik Christensen, and Joel Farrell, 17 June 2002.

Appendix I. XML Schemas

This appendix provides an XML Schema [XMLSchema1] definition for the http://schemas.xmlsoap.org/2003/04/swa, http://schemas.xmlsoap.org/2003/04/xbinc, and http://schemas.xmlsoap.org/2003/04/xmime namespaces.

The definition for the http://schemas.xmlsoap.org/2003/04/swa namespace is:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:soap12="http://www.w3.org/2002/12/soap-envelope"
           xmlns:xmime="http://schemas.xmlsoap.org/2003/04/xmime"
           xmlns:tns="http://schemas.xmlsoap.org/2003/04/swa"
           targetNamespace="http://schemas.xmlsoap.org/2003/04/swa" >

  <xs:import
      namespace="http://schemas.xmlsoap.org/soap/envelope/"
```

```
      schemaLocation="http://schemas.xmlsoap.org/soap/envelope" />
  <xs:import
      namespace="http://www.w3.org/2002/12/soap-envelope"
      schemaLocation="http://www.w3.org/2002/12/soap-envelope" />
  <xs:import
      namespace="http://schemas.xmlsoap.org/2003/04/xmime"
      schemaLocation="http://schemas.xmlsoap.org/2003/04/xmime" />

  <xs:element name="Representation" >
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xmime:Binary" >
          <xs:attribute name="URI" type="xs:anyURI" use="required" />
          <xs:attribute ref="soap11:mustUnderstand" use="optional" />
          <xs:attribute ref="soap12:mustUnderstand" use="optional" />
          <xs:attribute ref="soap11:actor" use="optional" />
          <xs:attribute ref="soap12:role" use="optional" />
          <xs:attribute ref="soap12:relay" use="optional" />
          <xs:anyAttribute namespace="##any" processContents="lax" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

The definition for the http://schemas.xmlsoap.org/2003/04/xbinc namespace is:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tns="http://schemas.xmlsoap.org/2003/04/xbinc"
           targetNamespace="http://schemas.xmlsoap.org/2003/04/xbinc" >

  <xs:import namespace="http://schemas.xmlsoap.org/soap/envelope/" />
  <xs:import namespace="http://www.w3.org/2002/12/soap-envelope" />

  <xs:element name="DoInclude" >
    <xs:complexType>
      <xs:attribute ref="soap11:mustUnderstand" use="optional" />
      <xs:attribute ref="soap12:mustUnderstand" use="optional" />
      <xs:attribute ref="soap11:actor" use="optional" />
      <xs:attribute ref="soap12:role" use="optional" />
      <xs:attribute ref="soap12:relay" use="optional" />
      <xs:anyAttribute namespace="##any" processContents="lax" />
    </xs:complexType>
```

```
  </xs:element>

  <xs:element name="Include" type="tns:Include" />
  <xs:complexType name="Include" >
    <xs:attribute name="href" type="xs:anyURI" use="required" />
  </xs:complexType>

</xs:schema>
```

The definition for the http://schemas.xmlsoap.org/2003/04/xmime namespace is:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tns="http://schemas.xmlsoap.org/2003/04/xmime"
           targetNamespace="http://schemas.xmlsoap.org/2003/04/xmime" >

  <xs:attribute name="MediaType" >
    <xs:simpleType>
      <xs:restriction base="xs:string" >
      <xs:pattern
value="(text|application|image|audio|video|model|multipart|message|x-[-
.a-z0-9]+)/[a-z0-9][-.+a-z0-9]+(;\s?.+=.+)*" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

  <xs:attribute name="Accept" >
    <xs:simpleType>
      <xs:restriction base="xs:string" >
        <xs:pattern
value="(text|application|image|audio|video|model|multipart|message|x-[-
.a-z0-9]+)/[a-z0-9][-.+a-z0-9]+(;p=(1\.0|0\.\d+))?" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

  <xs:complexType name="Binary" >
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary" >
        <xs:attribute ref="tns:MediaType" use="optional" />
        <xs:anyAttribute namespace="##any" processContents="lax" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

```
</xs:schema>
```