INTERNATIONAL STANDARD

ISO/IEC 29362

First edition
2008-06-15

Corrected version
2008-10-01

Information technology — Web Services Interoperability — WS-I Attachments Profile Version 1.0

Technologies de l'information — Interopérabilité des services du Web — Profil des fichiers joints WS-I, version 1.0

Reference number
ISO/IEC 29362:2008(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29362 was prepared by the Web Services Interoperability Organization (WS-I) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

This corrected version of ISO/IEC 29362:2008 includes characters which were missing from the sample code in 3.1, 3.4 and 4.4 of the original version.

Information technology — Web Services Interoperability — WS-I Attachments Profile Version 1.0

1 Scope and introduction

1.1 Scope

This International Standard defines the WS-I Attachments Profile 1.0 (hereafter, "Profile"), consisting of a set of non-proprietary Web services specifications, along with clarifications to and amplifications of those specifications that are intended to promote interoperability. This profile complements the WS-I Basic Profile 1.1 to add support for conveying interoperable SOAP Messages with Attachments-based attachments with SOAP messages.

SOAP Messages with Attachments (SwA) defines a MIME multipart/related structure for packaging attachments with SOAP messages. This profile complements the WS-I Basic Profile 1.1 to add support for conveying interoperable SwA-based attachments with SOAP messages.

Section 1 introduces the Profile, and explains its relationships to other profiles.

Section 2, "Profile Conformance," explains what it means to be conformant to the Profile.

Each subsequent section addresses a component of the Profile, and consists of two parts: an overview detailing the component specifications and their extensibility points, followed by subsections that address individual parts of the component specifications.

1.2 Relationship to other Profiles

This Profile adds support for SOAP with Attachments and MIME bindings, and is intended to be used in combination with the Basic Profile 1.1.

1.3 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 in RFC2119.

Normative statements of requirements in the Profile (i.e., those impacting conformance, as outlined in "Conformance Requirements") are presented in the following manner:

Rnnnn*Statement text here.*

where "nnnn" is replaced by a number that is unique among the requirements in the Profile, thereby forming a unique requirement identifier.

Requirement identifiers can be considered to be namespace qualified, in such a way as to be compatible with QNames from Namespaces in XML. If there is no explicit namespace prefix on a requirement's identifier (e.g., "R9999" as opposed to "bp10:R9999"), it should be interpreted as being in the namespace identified by the conformance URI of the document section it occurs in. If it is qualified, the prefix should be interpreted according to the namespace mappings in effect, as documented below.

Some requirements clarify the referenced specification(s), but do not place additional constraints upon implementations. For convenience, clarifications are annotated in the following manner: C

Some requirements are derived from ongoing standardization work on the referenced specification(s). For convenience, such forward-derived statements are annotated in the following manner: xxxx, where "xxxx" is an identifier for the specification (e.g., "WSDL20" for WSDL Version 2.0). Note that because such work was not complete when this document was published, the specification that the requirement is derived from may change; this information is included only as a convenience to implementers.

Extensibility points in underlying specifications (see "Conformance Scope") are presented in a similar manner:

Ennnn*Extensibility Point Name - Description*

where "nnnn" is replaced by a number that is unique among the extensibility points in the Profile. As with requirement statements, extensibility statements can be considered namespace-qualified.

This specification uses a number of namespace prefixes throughout; their associated URIs are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

- **soap** - "http://schemas.xmlsoap.org/soap/envelope/"
- **xsi** - "http://www.w3.org/2001/XMLSchema-instance"
- **xsd** - "http://www.w3.org/2001/XMLSchema"
- **soapenc** - "http://schemas.xmlsoap.org/soap/encoding/"
- **wsdl** - "http://schemas.xmlsoap.org/wsdl/"
- **soapbind** - "http://schemas.xmlsoap.org/wsdl/soap/"
- **mime** - "http://schemas.xmlsoap.org/wsdl/mime/"
- **uddi** - "urn:uddi-org:api_v2"
- **wsi** - "http://www.ws-i.org/schemas/conformanceClaim"
- **ref** - "http://ws-i.org/profiles/basic/1.1/xsd"

1.4 Profile Identification and Versioning

This document is identified by a name (in this case, Attachments Profile) and a version number (here, 1.0). Together, they identify a particular *profile instance*.

Version numbers are composed of a major and minor portion, in the form "major.minor". They can be used to determine the precedence of a profile instance; a higher version number (considering both the major and minor components) indicates that an instance is more recent, and therefore supersedes earlier instances.

Instances of profiles with the same name (e.g., "Example Profile 1.1" and "Example Profile 5.0") address interoperability problems in the same general scope (although some developments may require the exact scope of a profile to change between instances).

One can also use this information to determine whether two instances of a profile are backwards-compatible; that is, whether one can assume that conformance to an earlier profile instance implies conformance to a later one. Profile instances with the same name and major version number (e.g., "Example Profile 1.0" and "Example Profile 1.1") MAY be considered compatible. Note that this does not imply anything about compatibility in the other direction; that is, one cannot assume that conformance with a later profile instance implies conformance to an earlier one.

2 Profile Conformance

Conformance to the Profile is defined by adherence to the set of *requirements* defined for a specific *target*, within the *scope* of the Profile. This section explains these terms and describes how conformance is defined and used.

2.1 Conformance Requirements

Requirements state the criteria for conformance to the Profile. They typically refer to an existing specification and embody refinements, amplifications, interpretations and clarifications to it in order to improve interoperability. All requirements in the Profile are considered normative, and those in the specifications it references that are in-scope (see "Conformance Scope") should likewise be considered normative. When requirements in the Profile and its referenced specifications contradict each other, the Profile's requirements take precedence for purposes of Profile conformance.

Requirement levels, using RFC2119 language (e.g., MUST, MAY, SHOULD) indicate the nature of the requirement and its impact on conformance. Each requirement is individually identified (e.g., R9999) for convenience.

For example;

R9999 *WIDGETs SHOULD be round in shape.*

This requirement is identified by "R9999", applies to the target WIDGET (see below), and places a conditional requirement upon widgets; i.e., although this requirement must be met to maintain conformance in most cases, there are some situations where there may be valid reasons for it not being met (which are explained in the requirement itself, or in its accompanying text).

Each requirement statement contains exactly one requirement level keyword (e.g., "MUST") and one conformance target keyword (e.g., "MESSAGE"). The conformance target keyword appears in bold text (e.g. "**MESSAGE**"). Other conformance targets appearing in non-bold text are being used strictly for their definition and NOT as a conformance target. Additional text may be included to illuminate a requirement or group of requirements (e.g., rationale and examples); however, prose surrounding requirement statements must not be considered in determining conformance.

Definitions of terms in the Profile are considered authoritative for the purposes of determining conformance.

None of the requirements in the Profile, regardless of their conformance level, should be interpreted as limiting the ability of an otherwise conforming implementation to apply security countermeasures in response to a real or perceived threat (e.g., a denial of service attack).

2.2 Conformance Targets

Conformance targets identify what artifacts (e.g., SOAP message, WSDL description, UDDI registry data) or parties (e.g., SOAP processor, end user) requirements apply to.

This allows for the definition of conformance in different contexts, to assure unambiguous interpretation of the applicability of requirements, and to allow conformance testing of artifacts (e.g., SOAP messages and WSDL descriptions) and the behavior of various parties to a Web service (e.g., clients and service instances).

Requirements' conformance targets are physical artifacts wherever possible, to simplify testing and avoid ambiguity.

The following conformance targets are used in the Profile:

- **MESSAGE** - protocol elements that transport the ENVELOPE (e.g., SOAP/HTTP messages) (from ISO/IEC 29361)
- **ENVELOPE** - the serialization of the soap:Envelope element and its content (from ISO/IEC 29361)
- **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions) (from ISO/IEC 29361)

- **INSTANCE** - software that implements a wsdl:port or a uddi:bindingTemplate (from ISO/IEC 29361)
- **CONSUMER** - software that invokes an INSTANCE (from ISO/IEC 29361)
- **SENDER** - software that generates a message according to the protocol(s) associated with it (from ISO/IEC 29361)
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it (e.g., SOAP processors) (from ISO/IEC 29361)

2.3 Conformance Scope

The scope of the Profile delineates the technologies that it addresses; in other words, the Profile only attempts to improve interoperability within its own scope. Generally, the Profile's scope is bounded by the specifications referenced by it.

The Profile's scope is further refined by extensibility points. Referenced specifications often provide extension mechanisms and unspecified or open-ended configuration parameters; when identified in the Profile as an extensibility point, such a mechanism or parameter is outside the scope of the Profile, and its use or non-use is not relevant to conformance.

Note that the Profile may still place requirements on the use of an extensibility point. Also, specific uses of extensibility points may be further restricted by other profiles, to improve interoperability when used in conjunction with the Profile.

Because the use of extensibility points may impair interoperability, their use should be negotiated or documented in some fashion by the parties to a Web service; for example, this could take the form of an out-of-band agreement.

The Profile's scope is defined by the referenced specifications in Appendix A, as refined by the extensibility points in Appendix B.

2.4 Claiming Conformance

Claims of conformance to the Profile can be made using the following mechanisms, as described in Conformance Claim Attachment Mechanisms, when the applicable Profile requirements associated with the listed targets have been met:

- **WSDL 1.1 Claim Attachment Mechanism for Web Services Instances** - MESSAGE DESCRIPTION INSTANCE RECEIVER
- **WSDL 1.1 Claim Attachment Mechanism for Description Constructs** - DESCRIPTION
- **UDDI Claim Attachment Mechanism for Web Services Instances** - MESSAGE DESCRIPTION INSTANCE RECEIVER

The conformance claim URI for this Profile is "http://ws-i.org/profiles/attachments/1.0" .

3 Attachments Packaging

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- SOAP Messages with Attachments
  Extensibility points:
  - E0001 - MIME parts - SOAP Messages with Attachments places no restriction on the type of any non-root part in a multipart/related message.
- Extensible Markup Language (XML) 1.0 (Second Edition)
- Namespaces in XML 1.0
- RFC2557 MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)
- RFC2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- RFC2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
- RFC2392 Content-ID and Message-ID Uniform Resource Locators

SOAP Messages with Attachments defines a MIME `multipart/related` structure for packaging SOAP envelope with attachments. The Profile mandates the use of that structure, and places the following constraints on its use:

3.1 Root Part

R2931 *The entity body of the root part of multipart/related* **MESSAGE** *MUST be a soap:Envelope.*

R2945 *The Content-Type HTTP header field-value in a* **MESSAGE** *MUST be either "multipart/related" or "text/xml".* C

R2932 *If the Content-Type HTTP header field-value in a* **MESSAGE** *has a media-type of "multipart/related" then the Content-Type HTTP header field-value in that message MUST have the* `type` *parameter with a value of "text/xml".* C

Any MIME part may contain a soap:Envelope, but only the entity body of the root-part of the MIME package is treated as the primary SOAP envelope. Non-root parts are referred to as attachments.

For example,

CORRECT:

In the message below the the Content-Type HTTP header field-value has a media-type of 'Multipart/Related' and a parameter 'type' with value of 'text/xml'.

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
Content-Description: This is the optional message description.
```

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
...
</SOAP-ENV:Envelope>

--MIME_boundary

...
--MIME_boundary--
```

3.2 Encoding of Root Part

R2915 *The entity body of the root part of a `multipart/related` **MESSAGE** MUST be serialized using either UTF-8 or UTF-16 character encoding.*

R2916 *Non-root parts of a `multipart/related` **MESSAGE** MAY use any character encoding.*

3.3 Media Type of Message

R2925 *If the WSDL description lists at least one non-root MIME part, the corresponding MESSAGE MUST have a `Content-Type` HTTP header field-value with a media-type of "multipart/related".*

3.4 Messages with No Attachments

If a receiver expects zero or more attachments in a message, the sender of that message can use the text/xml media type for a message that has no attachments.

R2917 *A **MESSAGE** containing zero attachment parts MUST be sent using a content-type of either "text/xml" as though a SOAP HTTP binding were used or "multipart/related" when the WSDL description for the message specifies the `mime:multipartRelated` element on the corresponding `wsdl:input` or `wsdl:output` element in its `wsdl:binding`.*

R2902 *A **SENDER** MUST NOT send a message using SOAP with Attachments if the corresponding `wsdl:input` or `wsdl:output` element in the `wsdl:binding` does not specify the WSDL MIME Binding.*

This can happen only when the WSDL description specifies a `mime:multipartRelated` that has only one `mime:part` child element containing `soapbind:body`.

For example,

CORRECT:

A WSDL Description that is as follows:

```
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

...

    <wsdl:binding name="aBinding" type="tns:aPortType">
        <soapbind:binding style="rpc"
                          transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="anOperation">
            <soap:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body use="literal"
                                       namespace="http://example.com/mimetypes"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <soapbind:body use="literal"
                               namespace="http://example.com/mimetypes"/>
            </wsdl:output>
        </wsdl:operation>
 </wsdl:binding>
</wsdl:definitions>
```

may result in an input message which uses the SOAP HTTP Binding as follows:

```
<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:anOperation>
 ...
         </types:anOperation>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

must not result in an output message which uses the MIME Binding as follows:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<rootpart@example.com>"
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:anOperationResponse>
...
        </types:anOpertionResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary--
```

3.5 Dereferencing Attachments

Applications using Web services can use URIs in a variety of ways, including as a means of fetching content from a network. Although attachments can be used to provide content which is identified with a URI, implementations are not required to prefer attached content, to use it exclusively, or to use it at all. Likewise, applications may choose to ignore the specified dereference function of a URI (e.g., fetching content from a network) and only use attached content.

When using the CID URI scheme, the syntax and rules defined in RFC 2392 apply.

> R2918 *A* **RECEIVER** *MAY ignore a URI reference to an attachment in an envelope.*

3.6 Carrying Additional SOAP Envelopes

The Profile places no constraints on the content of attachment parts. Additional XML documents that contain a `soap:Envelope` may be sent as attachments, but only the root-part of the MIME message should be treated as the primary soap envelope in a MIME package.

> R2919 *A* **MESSAGE** *MAY contain `soap:Envelope`s carried as attachments in parts that are not the root part of the message.*

3.7 Fault Messages with Attachments

> R2920 *An* **INSTANCE** *MAY send a fault with attachments if and only if the `wsdl:output` element is described using the WSDL MIME binding.*

3.8 Value-space of Content-Id Header

Definition: content-id part encoding

The "content-id part encoding" consists of the concatenation of:

- The value of the name attribute of the `wsdl:part` element referenced by the `mime:content`, in which characters disallowed in content-id headers (non-ASCII characters as represented by code points above 0x7F) are escaped as follows:
  - Each disallowed character is converted to UTF-8 as one or more bytes.
  - Any bytes corresponding to a disallowed character are escaped with the URI escaping mechanism (that is, converted to %HH, where HH is the hexadecimal notation of the byte value).
  - The original character is replaced by the resulting character sequence.
- The character '=' (0x3D).

- A globally unique value such as a UUID.
- The character '@' (0x40).
- A valid domain name under the authority of the entity constructing the message.

R2933 *If a description binds a `wsdl:message` part to a `mime:content` element, then the corresponding MIME part's `content-id` field-value in a* **MESSAGE** *MUST conform to the content-id part encoding.*

For example,

CORRECT:

In the WSDL fragment below, the name of the part bound to the mime:content is the value appended to the content-id value.

```
<wsdl:message name="fooMsg">
  <wsdl:part name="body" type="ns1:Claim"/>
  <wsdl:part name="fooPart" type="xs:base64binary"/>
</wsdl:message>
...
<wsdl:binding
    ...
    <mime:multipartRelated>
      <mime:part>
        <soapbind:body parts="body" use="literal"/>
      </mime:part>
      <mime:part>
        <mime:content part="fooPart" type="application/octet-stream"/>
      </mime:part>
    </mime:multipartRelated>
    ...
</wsdl:binding>
```

Here's a fragment of the multipart package containing the `fooPart` binary stream highlighting how the "name" attribute of `wsdl:part` is incorporated into the content-id value.

```
...
--MIME_boundary
Content-Type: application/octet-stream
Content-Transfer-Encoding: 8bit
Content-ID: <fooPart=somereallybignumberlikeauuid@example.com>
...
```

3.9 Ordering of MIME Parts

It is possible that intermediaries might reorder the parts in a `multipart/related` message. Hence semantics should be neither given to, nor implied by the ordering of parts in a message.

R2921 *A* **RECEIVER** *MUST NOT infer any semantics from the ordering of non-root MIME parts in a message.*

R2929 *A* **MESSAGE** *MAY have its MIME parts in any order provided that the identity of the root part is maintained.*

A receiver must not assume that the order of `mime:part` elements specified in a WSDL description is the same as the order of MIME parts in the message. The order of MIME parts specified in a WSDL description must be considered independent of the order of MIME parts in the message.

3.10 Position of Root Part

If the `start` parameter is present, then the value of the `start` parameter is the `content-ID` of the root part of the message. In the absence of a `start` parameter, the root part is the first body part in the package, as defined by RFC 2387 Section 3.2.

> R2922 *If the `Content-Type` HTTP header field-value in a message does not have a `start` parameter, a* **RECEIVER** *MUST treat the first body part of the MIME package as the root part.* C

For example,

CORRECT:

In the message below the first MIME part (which has the Content-ID header of "<rootpart@example.com>") is the root part.

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:SendClaim>
            <ClaimDetail>............................</ClaimDetail>
            <photo>
              <href>cid:claimphoto@example.com</href>
            </photo>
        </types:SendClaim>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <claimphoto@example.com>

...binary photograph...
--MIME_boundary--
```

3.11 Content-Transfer-Encoding

Content-Transfer-Encodings allow messages to be sent across transports that do not support binary content; for example, some e-mail systems are only capable of transferring character-based messages. Because Web services messages may originate on or be destined for such systems, the Profile allows this mechanism's use.

If Content-Transfer-Encoding field of a part in mime multipart message is not present, the body of that part must conform to 7 bit ascii encoding as specified RFC 2045.

R2934 *The Content-Transfer-Encoding field of a part in a multipart/related* **MESSAGE** *MUST have a value of "7bit", "8bit", "binary", "quoted-printable" or "base64".*

R2935 *The encoding of the body of a part in a multipart/related* **MESSAGE** *MUST conform to the encoding indicated by the Content-Transfer-Encoding field-value, as specified by RFC2045.* C

The Profile restricts its legal values to those that are well-known to improve interoperability.

3.12 MIME Boundary String

Certain implementations have been shown to produce messages in which the MIME encapsulation boundary string is not preceded with a CRLF (carriage-return line-feed). This creates problems for implementations which correctly expect that the encapsulation boundary string is preceded by a CRLF.

R2936 *In a* **MESSAGE***, all MIME encapsulation boundary strings MUST be preceded with the ascii characters CR (13) and LF (10) in that sequence.* C

RFC2046 section 5.5.1 clearly requires that all encapsulation boundaries must be preceded with a CRLF (carriage-return line-feed).

4 Attachments Description

This section of the Profile incorporates the following specifications by reference:

- WSDL 1.1, Section 5.0

WSDL 1.1 section 5 defines the MIME binding. The Profile permits the use of the WSDL MIME binding, but limits it to the SOAP Messages with Attachments protocol. The Profile places the following constraints on its use:

4.1 Use of MIME Binding Extension

There may be use cases where a sender might be capable of sending a message using SOAP with Attachments yet incapable of receiving and processing such a message.

R2901 *A* **DESCRIPTION** *MUST use either the WSDL MIME Binding as described in WSDL 1.1 Section 5 or WSDL SOAP binding as described in WSDL 1.1 Section 3 on each of the* `wsdl:input` or `wsdl:output` *elements of a* `wsdl:binding`.

4.2 Unbound portType Element Contents

WSDL 1.1 is not explicit about whether it is permissible for a `wsdl:binding` to leave the binding for portions of the content defined by a `wsdl:portType` unspecified.

R2941 *A `wsdl:binding` in a* **DESCRIPTION** *SHOULD bind every `wsdl:part` of a `wsdl:message` in the `wsdl:portType` to which it refers to one of `soapbind:body`, `soapbind:header`, `soapbind:fault` , `soapbind:headerfault`, or `mime:content`.*

A portType defines an abstract contract with a named set of operations and associated abstract messages. Although not disallowed, it is expected that every part of the abstract input, output and fault messages specified in a portType is bound to `soapbind:body` or `soapbind:header` (and so forth) or a `mime:content` as appropriate when using the MIME binding as defined in WSDL 1.1 Section 5. Unbound wsdl:parts should be ignored by a consumer.

4.3 Referencing Message Parts

A message part in WSDL can be bound to a particular MIME part (using `mime:content`).Unlike a `soapbind:header` which may reference parts contained in a message that is not part of the contract defined by the `wsdl:porttype`, a `mime:content` must not reference a `wsdl:part` that is not defined in the `wsdl:message` referenced by the `wsdl:operation`. Additionally, message parts in WSDL are considered to be an indivisible unit. Components of a message part which is of complex content cannot be selectively bound to a particular MIME part.

R2903 *A `mime:content` element in a* **DESCRIPTION** *MUST NOT reference a `wsdl:part` that is not present in the respective `wsdl:input` or `wsdl:output` of the corresponding `wsdl:operation` of the corresponding `wsdl:portType`.*

R2904 *A `mime:content` element in a* **DESCRIPTION** *MUST NOT be bound to a sub-component of an element or type referenced by a `wsdl:part`.*

R2946 *In a* **DESCRIPTION**, *a `mime:content` element MUST include the `part` attribute.*

For example,

INCORRECT:
```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types ...>
    <schema xmlns="http://www.w3.org/2001/XMLSchema/"
            targetNamespace="http://example.org/foo"
            xmlns:ns="http://example.org/foo">
      <element name='foo'>
        <complextContent>
          <sequence>
            <element ref='bar1'/>
            <element ref='bar2'/>
         </sequence>
      </complextContent>
    </element>
  </schema>
</types>
 ...
```

```
<message name='aMsg'>
  <part name='apart' element='ns:foo' />
  <part name="body" element="ns:bar"/>
</message>
<portType>
  <operation>
    <input>
      <part name="apart">
    </input>
...
  </operation>
</portType>
<binding>
  <operation>
    <input>
      <mime:multipartRelated>
        <mime:part>
          <soapbind:body part="body" use="literal"/>
        </mime:part>
        <mime:part>
          <mime:content part="ns:bar1"/>
        </mime:part>
      </mime:multipartRelated>
    </input>
...
  </operation>
</binding>
</definitions>
```

4.4 Referencing Attachments from the SOAP Envelope

One of the advantages of having attachments is the ability to include data in a separate MIME part and refer to it from the SOAP envelope contained in the root part of the same MIME package.

This Profile defines a schema type `ref:swaRef` which can be used in a WSDL description to define a message part. When a message part is described using the `ref:swaRef` type, in its instance document, the URI points to an attachment in the same MIME package. This type is offered to application/tool/platform developers as an interoperable way to mark references to attachments in the descriptions. Nevertheless, use of other mechanisms does not make the description non-conformant.

The XML Schema for the type used to refer to attachments from the SOAP envelope is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema targetNamespace="http://ws-i.org/profiles/basic/1.1/xsd"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleType name="swaRef">
    <xsd:restriction base="xsd:anyURI" />
  </xsd:simpleType>
</xsd:schema>
```

As a convenience, WS-I has published the schema for this schema type at: http://ws-i.org/profiles/basic/1.1/swaref.xsd

Please note that there is no way in a WSDL1.1 description to correlate an attachment reference (defined using swaRef) with an attachment (defined using a wsdl:part bound to mime:content). As a best practice, the Profile recommends that when `ref:swaRef` is used, the corresponding attachment should not be described, and vice versa.

R2940 *In a* **DESCRIPTION**, *a `wsdl:part` defined with the `ref:swaRef` schema type SHOULD only be bound to a `soapbind:body`, or a `soapbind:header` in a MIME binding.*

R2928 *In an* **ENVELOPE***, a URI reference that is typed using the* *`ref:swaRef`* *schema type MUST resolve to a MIME part in the same message as the envelope.*

The swaRef type can be used to represent a reference to an attachment either as an element (as shown in the example below) or an attribute. There is no preferred approach.

For example,

CORRECT:

WSDL description for rpc/literal binding:

```
<?xml version="1.0"?>
<wsdl:definitions xmlns:types="http://example.com/mimetypes"
                  xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

    <wsdl:types>
        <xsd:schema targetNamespace="http://example.com/mimetypes"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

            <xsd:import namespace="http://ws-i.org/profiles/basic/1.1/xsd"/>
            <xsd:complexType name="ClaimDetailType">
                <xsd:sequence>
                    <xsd:element name="Name" type="xsd:string"/>
                    <xsd:element name="ClaimForm" type="ref:swaRef"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:schema>
    </wsdl:types>

    <wsdl:message name="ClaimIn">
        <wsdl:part name="ClaimDetail" type="types:ClaimDetailType"/>
        <wsdl:part name="ClaimPhoto" type="xsd:base64Binary"/>
    </wsdl:message>

    <wsdl:message name="ClaimOut">
        <wsdl:part name="ClaimRefNo" type="xsd:string"/>
    </wsdl:message>

    <wsdl:portType name="ClaimPortType">
        <wsdl:operation name="SendClaim">
            <wsdl:input message="tns:ClaimIn"/>
            <wsdl:output message="tns:ClaimOut"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="ClaimBinding" type="tns:ClaimPortType">
        <soapbind:binding style="rpc"
                      transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="SendClaim">
            <soapbind:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body use="literal"
                                       parts="ClaimDetail"
                                       namespace="http://example.com/mimetypes"/>
                    </mime:part>
                    <mime:part>
                        <mime:content part="ClaimPhoto"
                                      type="image/jpeg"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <soapbind:body use="literal"
                               namespace="http://example.com/mimetypes"/>
            </wsdl:output>
        </wsdl:operation>
 </wsdl:binding>
</wsdl:definitions>
```

Resulting input message for "SendClaim" rpc/literal operation:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<rootpart@example.com>"
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:SendClaim>
            <ClaimDetail>
                <Name>...</Name>
                <ClaimForm>cid:claimform@example.com</ClaimForm>
            </ClaimDetail>
        </types:SendClaim>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-ID: <claimform@example.com>

...claim form referenced by the swaRef...

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <ClaimPhoto=4d7a5fa2-14af-451c-961b-5c3abf786796@example.com>

...MIME attachment of binary photograph...
--MIME_boundary--
```

Resulting output message for "SendClaim" rpc/literal operation:

```
MIME-Version: 1.0
Content-Type: text/xml; charset=UTF-8

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:SendClaimResponse>
            <ClaimRefNo>..........................</ClaimRefNo>
        </types:SendClaimResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

CORRECT:

WSDL description for document/literal binding:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:types="http://example.com/mimetypes"
                  xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

    <wsdl:types>
        <xsd:schema targetNamespace="http://example.com/mimetypes"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">

             <xsd:import namespace="http://ws-i.org/profiles/basic/1.1/xsd" />
             <xsd:element name="ClaimDetail" type="types:ClaimDetailType"/>
             <xsd:complexType name="ClaimDetailType">
                <xsd:sequence>
                    <xsd:element name="Name" type="xsd:string"/>
                    <xsd:element name="ClaimForm" type="ref:swaRef"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="ClaimRefNo" type="xsd:string"/>
        </xsd:schema>
    </wsdl:types>

    <wsdl:message name="ClaimIn">
        <wsdl:part name="body" element="types:ClaimDetail"/>
        <wsdl:part name="ClaimPhoto" type="xsd:base64Binary"/>
    </wsdl:message>

    <wsdl:message name="ClaimOut">
        <wsdl:part name="out" element="types:ClaimRefNo"/>
    </wsdl:message>

    <wsdl:portType name="ClaimPortType">
        <wsdl:operation name="SendClaim">
            <wsdl:input message="tns:ClaimIn"/>
            <wsdl:output message="tns:ClaimOut"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="ClaimBinding" type="tns:ClaimPortType">
        <soapbind:binding style="document"
                          transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="SendClaim">
            <soapbind:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body parts="body" use="literal"/>
                    </mime:part>
                    <mime:part>
                        <mime:content part="ClaimPhoto" type="image/jpeg"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <soapbind:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Resulting input message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<rootpart@example.com>"
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
            <types:ClaimDetail>
                <Name>...</Name>
                <ClaimForm>cid:claimform@example.com</ClaimForm>
            </types:ClaimDetail>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: text/xml
Content-Transfer-Encoding: 8bit
Content-ID: <claimform@example.com>

...claim form referenced by the swaRef...

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <ClaimPhoto=4d7a5fa2-14af-451c-961b-5c3abf786796@example.com>

...MIME attachment of binary photograph...
--MIME_boundary--
```

Resulting output message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: text/xml; charset=UTF-8

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:ClaimRefNo>..............</types:ClaimRefNo>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.5 Specifying Root Part

SOAP Messages with Attachments requires that the root part of the multipart/related package must contain the SOAP envelope, but the WSDL MIME binding is unclear on how this is described.

R2911 *A `mime:multipartRelated` element in a* **DESCRIPTION** *MUST contain exactly one `mime:part` element, amongst its child `mime:part` elements, containing a `soapbind:body` child.* C

In a WSDL MIME binding, the `mime:part` that contains a `soapbind:body` describes the root MIME part required by SOAP Messages with Attachments.

For example,

INCORRECT:

```
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

...

    <wsdl:binding name="aBinding" type="tns:aPortType">
        <soapbind:binding style="rpc"
                         transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="anOperation">
            <soap:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body use="literal"
                                       namespace="http://example.com/mimetypes"/>
                    </mime:part>
                    <mime:part>
                        <soapbind:body use="literal"
                                       namespace="http://example.com/mimetypes"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
 ...
             </wsdl:output>
        </wsdl:operation>
 </wsdl:binding>
</wsdl:definitions>
```

4.6 Specifying SOAP Headers in Root Part

The WSDL1.1 specification does not specify whether the `soapbind:header` element is permitted as a child of the `mime:part` element along with the `soapbind:body` element. The SOAP Messages with Attachments specification requires that the root part of the multipart message contain the SOAP envelope, but the WSDL1.1 specification is unclear as to how to describe this part. Since the WSDL1.1 specification specifies that the `mime:part` element is used to describe each part of a multipart/related message, the contents of the `mime:part` element that represents the root part of the multipart message must therefore fully describe the SOAP envelope, including the `soapbind:body` and `soapbind:header` elements just as they would be used in the absence of the WSDL MIME binding extension.

R2905 *The `soapbind:header` element in a* **DESCRIPTION** *MAY be included as a child of the `mime:part` element.* C

R2906 *A `soapbind:header` element in a* **DESCRIPTION** *MUST NOT be included in a `mime:part` that is not the root part, containing the `soapbind:body` element.* C

For example,

INCORRECT:
```
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

...

    <wsdl:binding name="aBinding" type="tns:aPortType">
        <soapbind:binding style="rpc"
                          transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="anOperation">
            <soap:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body use="literal"
                                       namespace="http://example.com/mimetypes"/>
                    </mime:part>
                    <mime:part>
                        <soapbind:header message="tns:headerMessage"
                                         part="aPart"
                                         use="literal"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
 ...
             </wsdl:output>
        </wsdl:operation>
 </wsdl:binding>
</wsdl:definitions>
```

4.7 MIME Binding Schema Fixes

There are a number of discrepancies between the WSDL1.1 specification and the WSDL MIME binding schema. In the case of the `mime:part` element, the schema incorrectly defines it as a local element declaration and it incorrectly adds a `name` attribute that is not described in the WSDL1.1 specification.

These and other fixes to the WSDL MIME Binding extension schema are reflected in the revised schema located at "http://ws-i.org/profiles/basic/1.1/wsdlmime-2004-08-24.xsd".

R2907 *MIME parts in a* **DESCRIPTION** *MUST be defined using an element with a local name of* `part` *in the namespace of the WSDL MIME Binding extension.* C

R2908 *The* `mime:part` *element in a* **DESCRIPTION** *MUST NOT have a* `name` *attribute.*

4.8 Specifying Alternate Media Types

Multiple `mime:content` element children of a `mime:part` are considered alternate acceptable serializations of the referenced `wsdl:part`.

R2909 *Multiple* `mime:content` *child elements of a* `mime:part` *element in a* **DESCRIPTION** *MUST reference the same* `wsdl:part`*.*

For example,

INCORRECT:
```
<mime:part>
  <mime:content part="ns:foo" type="image/jpeg"/>
  <mime:content part="ns:bar" type="image/jpeg"/>
</mime:part>
```

CORRECT:
```
<mime:part>
  <mime:content part="ns:foo" type="image/jpeg"/>
  <mime:content part="ns:foo" type="image/gif"/>
</mime:part>
```

4.9 WSDL Parts

R2910 *A `mime:content` in a* **DESCRIPTION** *MUST reference a `wsdl:part` that is defined using either the `type` attribute or the `element` attribute.*

R2942 *In a* **MESSAGE***, a message part bound to a `mime:content` that refers to global element declaration (via the element attribute of the `wsdl:part` element) MUST be serialized within the MIME part as a serialization of an XML infoset whose root element is described by the referenced element.*

R2943 *In a* **DESCRIPTION***, if a message part is bound to a `mime:content` that refers to a type (via the type attribute of the `wsdl:part` element), then the value of that type attribute MUST be ignored in favor of media type of the type attribute of the `mime:content` element.*

R2944 *In a* **DESCRIPTION***, if a `wsdl:part` element refers to a global element declaration (via the element attribute of the `wsdl:part` element) then the value of the type attribute of a `mime:content` element that binds that part MUST be a content type suitable for carrying an XML serialization.*

4.10 Ordering of Parts

R2912 *A* **RECEIVER** *MUST NOT assume that the order of `mime:part` elements specified in a WSDL description is the same as the order of MIME parts in the message.*

R2947 *In a DESRIPTION, a `mime:part` element that contains a `soapbind:body` child element MAY appear in any position amongst the other child elements of a `mime:multipartRelated` element.*

The order of MIME parts specified in a WSDL description must be considered independent of the order of MIME parts in the message.

4.11 Sending Fault Messages

R2913 *A Fault* **MESSAGE** *MAY be serialized as either* `text/xml` *or* `multipart/related`*, if the* `wsdl:output` *child element of the corresponding binding operation in a description has a child* `mime:multipartRelated` *element.*

4.12 Describing Faults

R2930 *A* `wsdl:fault` *element in a* **DESCRIPTION** *MUST NOT have* `mime:multipartRelated` *element as its child element.*

4.13 Sending Additional Parts Not Described in WSDL

Additional MIME parts may be included in the message beyond those described in the WSDL, and their position or order within the MIME package is not important.

R2923 *A* **SENDER** *MAY send non-root MIME parts not described in the WSDL MIME binding.* C

R2926 *A* **MESSAGE** *MUST include all of the MIME parts described by its WSDL MIME binding.*

4.14 Conformance of SOAP Messages

Profile conformance criteria for the conformance target ENVELOPE only apply to the SOAP envelope contained in the root part of the MIME package. SOAP envelopes in non-root parts may be described in a WSDL description as attachments, in which case, conformance criteria for non-root parts listed in the WSDL description apply.

R2927 *The root part of a* **MESSAGE** *MUST be conformant with all the requirements for an envelope in version 1.1 of the Basic Profile.*

4.15 Example Attachment Description Using mime:conent

For example,

CORRECT:

WSDL description for document/literal:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:types="http://example.com/mimetypes"
                  xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">
    <wsdl:types>
        <xsd:schema targetNamespace="http://example.com/mimetypes"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">

            <xsd:element name="ClaimDetail" type="types:ClaimDetailType"/>
            <xsd:complexType name="ClaimDetailType">
                <xsd:sequence>
                    <xsd:element name="Name" type="xsd:string"/>
                    <!-- lots of other claim detail stuff -->
                </xsd:sequence>
            </xsd:complexType>

            <xsd:element name="ClaimRefNo" type="xsd:string"/>

        </xsd:schema>
    </wsdl:types>

    <wsdl:message name="ClaimIn">
        <wsdl:part name="body" element="types:ClaimDetail"/>
        <wsdl:part name="ClaimPhoto" type="xsd:base64Binary"/>
    </wsdl:message>

    <wsdl:message name="ClaimOut">
        <wsdl:part name="out" element="types:ClaimRefNo"/>
    </wsdl:message>

    <wsdl:portType name="ClaimPortType">
        <wsdl:operation name="SendClaim">
            <wsdl:input message="tns:ClaimIn"/>
            <wsdl:output message="tns:ClaimOut"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="ClaimBinding" type="tns:ClaimPortType">
        <soapbind:binding style="document"
                        transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="SendClaim">
            <soapbind:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body parts="body" use="literal"/>
                    </mime:part>
                    <mime:part>
                        <mime:content part="ClaimPhoto" type="image/jpeg"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <soapbind:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Resulting input message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<rootpart@example.com>"
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
            <types:ClaimDetail>
                <Name>...</Name>
                <!-- lots of other claim detail stuff -->
            </types:ClaimDetail>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <ClaimPhoto=4d7a5fa2-14af-451c-961b-5c3abf786796@example.com>

...MIME attachment of binary photograph...
--MIME_boundary--
```

Resulting output message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: text/xml; charset=UTF-8

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:ClaimRefNo>..............</types:ClaimRefNo>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.16 Example Attachment Description Using swaRef

For example,

CORRECT:

WSDL description for document/literal:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:types="http://example.com/mimetypes"
                  xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  targetNamespace="http://example.com/mimewsdl"
                  xmlns:tns="http://example.com/mimewsdl">

    <wsdl:types>
        <xsd:schema targetNamespace="http://example.com/mimetypes"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">

            <xsd:import namespace="http://ws-i.org/profiles/basic/1.1/xsd" />
            <xsd:element name="ClaimDetail" type="types:ClaimDetailType"/>
            <xsd:complexType name="ClaimDetailType">
                <xsd:sequence>
                    <xsd:element name="Name" type="xsd:string"/>
                    <!-- lots of other claim detail stuff -->
                    <xsd:element name="ClaimPhoto" type="ref:swaRef"/>
                </xsd:sequence>
            </xsd:complexType>

            <xsd:element name="ClaimRefNo" type="xsd:string"/>

        </xsd:schema>
    </wsdl:types>

    <wsdl:message name="ClaimIn">
        <wsdl:part name="body" element="types:ClaimDetail"/>
    </wsdl:message>

    <wsdl:message name="ClaimOut">
        <wsdl:part name="out" element="types:ClaimRefNo"/>
    </wsdl:message>

    <wsdl:portType name="ClaimPortType">
        <wsdl:operation name="SendClaim">
            <wsdl:input message="tns:ClaimIn"/>
            <wsdl:output message="tns:ClaimOut"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="ClaimBinding" type="tns:ClaimPortType">
        <soapbind:binding style="document"
                        transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="SendClaim">
            <soapbind:operation soapAction="http://example.com/soapaction"/>
            <wsdl:input>
                <mime:multipartRelated>
                    <mime:part>
                        <soapbind:body parts="body" use="literal"/>
                    </mime:part>
                </mime:multipartRelated>
            </wsdl:input>
            <wsdl:output>
                <soapbind:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Resulting input message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
        start="<rootpart@example.com>"
Content-Description: This is the optional message description.

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

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
            <types:ClaimDetail>
                <Name>...</Name>
                <!-- lots of other claim detail stuff -->
                <ClaimPhoto>cid:claimphoto@example.com</ClaimPhoto>
            </types:ClaimDetail>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

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

...MIME attachment of binary photograph...
--MIME_boundary--
```

Resulting output message for "SendClaim" document/literal operation:

```
MIME-Version: 1.0
Content-Type: text/xml; charset=UTF-8

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body xmlns:types="http://example.com/mimetypes">
        <types:ClaimRefNo>..............</types:ClaimRefNo>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Appendix A: Referenced Specifications

The following specifications' requirements are incorporated into the Profile by reference, except where superseded by the Profile:

- SOAP Messages with Attachments
- Extensible Markup Language (XML) 1.0 (Second Edition)
- Namespaces in XML 1.0
- RFC2557 MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)
- RFC2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
- RFC2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
- RFC2392 Content-ID and Message-ID Uniform Resource Locators
- WSDL 1.1, Section 5.0

Appendix B: Extensibility Points

This section identifies extensibility points, as defined in "Scope of the Profile," for the Profile's component specifications.

These mechanisms are out of the scope of the Profile; their use may affect interoperability, and may require private agreement between the parties to a Web service.

In SOAP Messages with Attachments:

- E0001 - **MIME parts** - SOAP Messages with Attachments places no restriction on the type of any non-root part in a multipart/related message.

Appendix C: Normative References

In addition to all of the profiled specifications listed in Appendix A, the following specifications are referenced:

- RFC2119, http://ietf.org/rfc/rfc2119, Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, March 1997.
- WS-I Basic Profile 1.0, http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html, K. Ballinger et al., April 2004.
- Namespaces in XML 1.0 (Second Edition), http://www.w3.org/TR/2006/REC-xml-names-20060816, T. Bray et al., August 2006.
- WS-I Conformance Claim Attachment Mechanisms Version 1.0, http://www.ws-i.org/Profiles/ConformanceClaims-1.0-2004-11-15.html, M. Nottingham et al., November 2004.

Appendix D: Defined Terms

The following list of terms have specific definitions that are authoritative for this profile:

content-id part encoding

The "content-id part encoding" consists of the concatenation of:

- The value of the name attribute of the `wsdl:part` element referenced by the `mime:content`, in which characters disallowed in content-id headers (non-ASCII characters as represented by code points above 0x7F) are escaped as follows:
  - Each disallowed character is converted to UTF-8 as one or more bytes.
  - Any bytes corresponding to a disallowed character are escaped with the URI escaping mechanism (that is, converted to %HH, where HH is the hexadecimal notation of the byte value).
  - The original character is replaced by the resulting character sequence.
- The character '=' (0x3D).
- A globally unique value such as a UUID.
- The character '@' (0x40).
- A valid domain name under the authority of the entity constructing the message.

Appendix E: Acknowledgements

This document is the work of the WS-I Basic Profile Working Group, whose members have included:

Mark Allerton (Crystal Decisions Corp), Steve Anderson (OpenNetwork), George Arriola (Talking Blocks, Inc.), Siddharth Bajaj (Verisign), Keith Ballinger (Microsoft Corp.), David Baum (Kantega AS), Ilya Beyer (KANA), Rich Bonneau (IONA Technologies), Don Box (Microsoft Corp.), Andrew Brown (Verisign), Heidi Buelow (Quovadx), David Burdett (Commerce One, Inc.), Luis Felipe Cabrera (Microsoft Corp.), Maud Cahuzac (France Telecom), Mike Chadwick (Kaiser Permanente), Martin Chapman (Oracle Corporation), Richard Chennault (Kaiser Permanente), Roberto Chinnici (Sun Microsystems), Dipak Chopra (SAP AG), Jamie Clark (OASIS), David Cohen (Merrill Lynch), Ugo Corda (SeeBeyond Tech), Paul Cotton (Microsoft Corp.), Joseph Curran (Accenture), Alex Deacon (Verisign), Mike DeNicola (Fujitsu Limited), Paul Downey (BT Group), Jacques Durand (Fujitsu Limited), Aladin Eajani (Hummingbird, Ltd.), Michael Eder (Nokia), Dave Ehnebuske (IBM), Mark Ericson (Mindreef Inc), Colleen Evans (Microsoft Corp.), Tim Ewald (Microsoft Corp.), Chuck Fay (FileNET Corp.), Chris Ferris (IBM), Daniel Foody (Actional Corporation), Satoru Fujita (NEC Corporation), Shishir Garg (France Telecom), Yaron Goland (BEA Systems Inc), Marc Goodner (SAP AG), Pierre Goyette (Hummingbird, Ltd.), Hans Granqvist (Verisign), Martin Gudgin (Microsoft Corp.), Marc Hadley (Sun Microsystems), Norma Hale (Webify Solutions Inc), Bob Hall (Unisys Corporation), Scott Hanselman (Corillian), Muir Harding (Autodesk Inc.), Loren Hart (Verisign), Andrew Hately (IBM), Harry Holstrom (Accenture), Lawrence Hsiung (Quovadx), Hemant Jain (Tata Consultancy), Steve Jenisch (SAS Institute), Erik Johnson (Epicor Software), Bill Jones (Oracle Corporation), Anish Karmarkar (Oracle Corporation), Dana Kaufman (Forum Systems), Takahiro Kawamura (Toshiba), Oldre Kepka (Systinet), Bhushan Khanal (WRQ Inc.), Sandy Khaund (Microsoft Corp.), Jacek Kopecky (Systinet), Sanjay Krishnamurthi (Informatica), Sundar Krishnamurthy (Verisign), Eva Kuiper (Hewlett-Packard), Sunil Kunisetty (Oracle Corporation), Christopher Kurt (Microsoft Corp.), Lars Laakes (Microsoft Corp.), Canyang Kevin Liu (SAP AG), Ted Liu (webMethods Inc.), Donna Locke (Oracle Corporation), Brad Lund (Intel), Michael Mahan (Nokia), Ron Marchi (EDS), Jonathan Marsh (Microsoft Corp.), Eric Matland (Hummingbird, Ltd.), Barbara McKee (IBM), Derek Medland (Hummingbird, Ltd.), David Meyer (Plumtree Software Inc.), Jeff Mischkinsky (Oracle Corporation), Ray Modeen (MITRE Corp.), Tom Moog (Sarvega Inc.), Gilles Mousseau (Hummingbird, Ltd.), Greg Mumford (MCI), Jim Murphy (Mindreef Inc), Bryan Murray (Hewlett-Packard), Richard Nikula (BMC Software, Inc.), Eisaku Nishiyama (Hitachi, Ltd.), Mark Nottingham (BEA Systems Inc), David Orchard (BEA Systems Inc), Vivek Pandey (Sun Microsystems), Jesse Pangburn (Quovadx), Eduardo Pelegri-Llopart (Sun Microsystems), Mike Perham (Webify Solutions Inc), Eric Rajkovic (Oracle Corporation), Shaan Razvi (MITRE Corp.), Rimas Rekasius (IBM), Mark Richards (Fidelity), Graeme Riddell (Bowstreet), Sam Ruby (IBM), Tom Rutt (Fujitsu Limited), Saikat Saha (Commerce One, Inc.), Roger Sanborn (Crystal Decisions Corp), Matt Sanchez (Webify Solutions Inc), Krishna Sankar (Cisco Systems Inc.), Jeffrey Schlimmer (Microsoft Corp.), Don Schricker (Micro Focus), Dave Seidel (Mindreef Inc), AKIRA

SHIMAYA (NTT), David Shoaf (Hewlett-Packard), Yasser Shohoud (Microsoft Corp.), David Smiley (Ascential Software), Seumas Soltysik (IONA Technologies), Joseph Stanko (Plumtree Software Inc.), Andrew Stone (Accenture), Julie Surer (MITRE Corp.), YASUO TAKEMOTO (NTT), Nobuyoshi Tanaka (NEC Corporation), Jorgen Thelin (Microsoft Corp.), Sameer Vaidya (Talking Blocks, Inc.), William Vambenepe (Hewlett-Packard), Claus von Riegen (SAP AG), Rick Weil (Eastman Kodak Company), Scott Werden (WRQ Inc.), Ajamu Wesley (IBM), Ian White (Micro Focus), Dave Wilkinson (Vignette), Mark Wood (Eastman Kodak Company), Prasad Yendluri (webMethods Inc.), and Brandon Zhu (NetManage Inc).

ICS 35.100.05

Price based on 32 pages