Message Body Handling in the Session Initiation Protocol (SIP)

Abstract

   This document specifies how message bodies are handled in SIP.
   Additionally, this document specifies SIP user agent support for MIME
   (Multipurpose Internet Mail Extensions) in message bodies.

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

   Message body handling in SIP was originally specified in [RFC3261],
   which relied on earlier specifications (e.g., MIME) to describe some
   areas. This document contains background material on how bodies are
   handled in SIP and normative material on areas that had not been
   specified before or whose specifications needed to be completed.
   Sections containing background material are clearly identified as
   such by their titles. The material on the normative sections is
   based on experience gained since [RFC3261] was written. Implementers
   need to implement what is specified in [RFC3261] (and its references)
   in addition to what is specified in this document.

2. Terminology

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

   The following abbreviations are used in this document.

      UA: User Agent

      UAC: User Agent Client

      UAS: User Agent Server

      URL: Uniform Resource Locator

3. Message Body Encoding

   This section deals with the encoding of message bodies in SIP.

3.1. Background on Message Body Encoding

   SIP [RFC3261] messages consist of an initial line (request line in
   requests and status line in responses), a set of header fields, and
   an optional message body. The message body is described using header
   fields such as Content-Disposition, Content-Encoding, and Content-
   Type, which provide information on its contents. Figure 1 shows a
   SIP message that carries a body. Some of the header fields are not
   shown for simplicity:

```
INVITE sip:conf-fact@example.com SIP/2.0
Content-Type: application/sdp
Content-Length: 192

v=0
o=alice 2890844526 2890842807 IN IP4 atlanta.example.com
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 20002 RTP/AVP 31
a=rtpmap:31 H261/90000
```

Figure 1: SIP message carrying a body

The message body of a SIP message can be divided into various body
parts. Multipart message bodies are encoded using the MIME
(Multipurpose Internet Mail Extensions) [RFC2045] format. Body parts
are also described using header fields such as Content-Disposition,
Content-Encoding, and Content-Type, which provide information on the
contents of a particular body part. Figure 2 shows a SIP message
that carries two body parts. Some of the header fields are not shown
for simplicity:

```
INVITE sip:conf-fact@example.com SIP/2.0
Content-Type: multipart/mixed;boundary="boundary1"
Content-Length: 619

--boundary1
Content-Type: application/sdp

v=0
o=alice 2890844526 2890842807 IN IP4 atlanta.example.com
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 20002 RTP/AVP 31
a=rtpmap:31 H261/90000

--boundary1
Content-Type: application/resource-lists+xml
Content-Disposition: recipient-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
  <list>
    <entry uri="sip:bill@example.com"/>
    <entry uri="sip:randy@example.net"/>
    <entry uri="sip:joe@example.org"/>
  </list>
</resource-lists>
--boundary1--
```

Figure 2: SIP message carrying a body

SIP uses S/MIME [RFC3851] to protect message bodies. As specified in
[RFC3261], UASs that cannot decrypt a message body or a body part can
use the 493 (Undecipherable) response to report the error.

3.2. UA Behavior to Encode Binary Message Bodies

SIP messages can carry binary message bodies such as legacy
signalling objects [RFC3204]. SIP proxy servers are 8-bit safe.
That is, they are able to handle binary bodies. Therefore, there is
no need to use encodings such as base64 to transport binary bodies in
SIP messages. Consequently, UAs SHOULD use the binary transfer
encoding [RFC4289] for all payloads in SIP, including binary
payloads. The only case where a UA MAY use a different encoding is
when transferring application data between applications that only
handle a different encoding (e.g., base64).

4. 'multipart' Message Bodies

   This section deals with 'multipart' message bodies and their
   handling.

4.1. Background on 'multipart' Message Bodies

   [RFC3261] did not mandate support for 'multipart' message bodies in
   MIME format [RFC2046]. However, since [RFC3261] was written, many
   SIP extensions rely on them.

   The use of 'multipart/mixed' MIME bodies is a useful tool to build
   SIP extensions. An example of such an extension could be the
   inclusion of location information in an INVITE request. Such an
   INVITE request would use the 'multipart/mixed' MIME type [RFC2046] to
   carry two body parts: a session description and a location object.
   An example of an existing extension that uses 'multipart/mixed' to
   send a session description and a legacy-signalling object is defined
   in [RFC3204].

   Another MIME type that is useful to build SIP extensions is
   'multipart/alternative' [RFC2046]. Each body part within a
   'multipart/alternative' carries an alternative version of the same
   information.

   The transition from SDP to new session description protocols could be
   implemented using 'multipart/alternative' bodies. SIP messages
   (e.g., INVITE requests) could carry a 'multipart/alternative' body
   with two body parts: a session description written in SDP and a
   session description written in a newer session description format.
   Legacy recipient UAs would use the session description written in
   SDP. New recipient UAs would use the one written in the newer
   format.

   Nested MIME bodies are yet another useful tool to build and combine
   SIP extensions. Using the extensions in the previous examples, a UA
   that supported a new session description format and that needed to
   include a location object in an INVITE request would include a
   'multipart/mixed' body with two body parts: a location object and a
   'multipart/alternative'. The 'multipart/alternative' body part
   would, in turn, have two body parts: a session description written in
   SDP and a session description written in the newer session
   description format.

4.2. Mandatory Support for 'multipart' Message Bodies

   For all MIME-based extensions to work, the recipient needs to be able
   to decode the multipart bodies. Therefore, SIP UAs MUST support
   parsing 'multipart' MIME bodies, including nested body parts.
   Additionally, UAs MUST support the 'multipart/mixed' and 'multipart/
   alternative' MIME types. Support for other MIME types such as
   'multipart/related' is OPTIONAL.

      Note that, by default, unknown 'multipart' subtypes are treated as
      'multipart/mixed'. Also note that SIP extensions can also include
      'multipart' MIME bodies in responses. That is why both UACs and
      UASs need to support 'multipart' bodies.

   Legacy SIP UAs without support for 'multipart' bodies generate a 415
   (Unsupported Media Type) response when they receive a 'multipart'
   body in a request. A UAC sending a 'multipart' body can receive such
   an error response when communicating with a legacy SIP UA that
   predates this specification.

      It has been observed in the field that a number of legacy SIP UAs
      without support for 'multipart' bodies simply ignored those bodies
      when they were received. These UAs did not return any error
      response. Unsurprisingly, SIP UAs not being able to report this
      type of error have caused serious interoperability problems in the
      past.

4.3. UA Behavior to Generate 'multipart' Message Bodies

   UAs SHOULD avoid unnecessarily nesting body parts because doing so
   would, unnecessarily, make processing the body more laborious for the
   receiver. However, [RFC2046] states that a 'multipart' media type
   with a single body part is useful in some circumstances (e.g., for
   sending non-text media types). In any case, UAs SHOULD NOT nest one
   'multipart/mixed' within another unless there is a need to reference
   the nested one (i.e., using the Content ID of the nested body part).
   Additionally, UAs SHOULD NOT nest one 'multipart/alternative' within
   another.

      Note that UAs receiving unnecessarily nested body parts treat them
      as if they were not nested.

5. 'multipart/mixed' Message Bodies

   This section does not specify any additional behavior regarding how
   to generate and process 'multipart/mixed' bodies. This section is
   simply included for completeness.

6.   'multipart/alternative' Message Bodies

   This section deals with 'multipart/alternative' message bodies and
   their handling.

6.1.   Background on 'multipart/alternative' Message Bodies

   Each body part within a 'multipart/alternative' carries an
   alternative version of the same information. The body parts are
   ordered so that the last one is the richest representation of the
   information. The recipient of a 'multipart/alternative' body chooses
   the last body part it understands.

      Note that within a body part encoded in a given format (i.e., of a
      given content type), there can be optional elements that can
      provide richer information to the recipient in case the recipient
      supports them. For example, in SDP (Session Description Protocol)
      [RFC4566], those optional elements are encoded in 'a' lines.
      These types of optional elements are internal to a body part and
      are not visible at the MIME level. That is, a body part is
      understood if the recipient understands its content type,
      regardless of whether or not the body part's optional elements are
      understood.

      Note as well that each part of a 'multipart/alternative' body
      represents the same data, but the mapping between any two parts is
      not necessarily without information loss. For example,
      information can be lost when translating 'text/html' to 'text/
      plain'. [RFC2046] recommends that each part should have a
      different Content-ID value in the case where the information
      content of the two parts is not identical.

6.2.   UA Behavior to Generate 'multipart/alternative' Message Bodies

   Section 8.2 mandates all the top-level body parts within a
   'multipart/alternative' to have the same disposition type.

   The 'session' and 'early-session' [RFC3959] disposition types require
   that all the body parts of a 'multipart/alternative' body have
   different content types. Consequently, for the 'session' and 'early-
   session' disposition types, UAs MUST NOT place more than one body
   part with a given content type in a 'multipart/alternative' body.
   That is, for 'session' and 'early-session', no body part within a
   'multipart/alternative' can have the same content type as another
   body part within the same 'multipart/alternative'.

6.3. UA Behavior to Process 'multipart/alternative' Message Bodies

   This section does not specify any additional behavior regarding how
   to process 'multipart/alternative' bodies. This section is simply
   included for completeness.

7. 'multipart/related' Message Bodies

   This section deals with 'multipart/related' message bodies and their
   handling.

7.1. Background on 'multipart/related' Message Bodies

   Compound objects in MIME are represented using the 'multipart/
   related' content type [RFC2387]. The body parts within a particular
   'multipart/related' body are all part of a compound object and are
   processed as such. The body part within a 'multipart/related' body
   that needs to be processed first is referred to as the 'root' body
   part. The root body part of a 'multipart/related' body is identified
   by the 'start' parameter, which is a Content-Type header field
   parameter and contains a Content-ID URL pointing to the root body
   part. If the start parameter is not present, the root body part is,
   by default, the first body part of the 'multipart/related'. An
   example of a compound object is a web page that contains images. The
   html body part would be the root. The remaining body parts would
   contain the images. An example of a SIP extension using 'multipart/
   related' is specified in [RFC4662].

7.2. UA Behavior to Generate 'multipart/related' Message Bodies

   This section does not specify any additional behavior regarding how
   to generate 'multipart/related' bodies. This section is simply
   included for completeness.

7.3. UA Behavior to Process 'multipart/related' Message Bodies

   Per [RFC2387], a UA processing a 'multipart/related' body processes
   the body as a compound object ignoring the disposition types of the
   body parts within it. Ignoring the disposition types of the
   individual body parts makes sense in the context in which 'multipart/
   related' was originally specified. For instance, in the example of
   the web page, the implicit disposition type for the images would be
   'inline', since the images are displayed as indicated by the root
   html file. However, in SIP, the disposition types of the individual
   body parts within a 'multipart/related' play an important role and,
   thus, need to be considered by the UA processing the 'multipart/
   related'. Different SIP extensions that use the same disposition
   type for the 'multipart/related' body can be distinguished by the

disposition types of the individual body parts within the 'multipart/
related'. Consequently, SIP UAs processing a 'multipart/related'
body with a given disposition type MUST process the disposition types
of the body parts within it according to the SIP extension making use
the disposition type of the 'multipart/related'.

   Note that UAs that do not understand 'multipart/related' will
   treat 'multipart/related' bodies as 'multipart/mixed' bodies.
   These UAs will not be able to process a given body as a compound
   object. Instead, they will process the body parts according to
   their disposition type as if each body part was independent from
   each other.

8. Disposition Types

   This section deals with disposition types in message bodies.

8.1. Background on Content and Disposition Types in SIP

   The Content-Disposition header field, defined in [RFC2183] and
   extended by [RFC3261], describes how to handle a SIP message's body
   or an individual body part. Examples of disposition types used in
   SIP in the Content-Disposition header field are 'session' and
   'render'.

   [RFC3204] and [RFC3459] define the 'handling' parameter for the
   Content-Disposition header field. This parameter describes how a UAS
   reacts if it receives a message body whose content type or
   disposition type it does not understand. If the parameter has the
   value 'optional', the UAS ignores the message body; if the parameter
   has the value 'required', the UAS returns a 415 (Unsupported Media
   Type) response. The default value for the 'handling' parameter is
   'required'. The following is an example of a Content-Disposition
   header field:

      Content-Disposition: signal; handling=optional

   [RFC3204] identifies two situations where a UAS (User Agent Server)
   needs to reject a request with a body part whose handling is
   required:

   1. if it has an unknown content type.

   2. if it has an unknown disposition type.

   If the UAS did not understand the content type of the body part, the
   UAS can add an Accept header field to its 415 (Unsupported Media
   Type) response listing the content types that the UAS does

understand. Nevertheless, there is no mechanism for a UAS that does
not understand the disposition type of a body part to inform the UAC
about which disposition type was not understood or about the
disposition types that are understood by the UAS.

The reason for not having such a mechanism is that disposition types
are typically supported within a context. Outside that context, a UA
need not support the disposition type. For example, a UA can support
the 'session' disposition type for body parts in INVITE and UPDATE
requests and their responses. However, the same UA would not support
the 'session' disposition type in MESSAGE requests.

In another example, a UA can support the 'render' disposition type
for 'text/plain' and 'text/html' body parts in MESSAGE requests.
Additionally, the UA can support the 'session' disposition type for
'application/sdp' body parts in INVITE and UPDATE requests and their
responses. However, the UA might not support the 'render'
disposition type for 'application/sdp' body parts in MESSAGE
requests, even if, in different contexts, the UA supported all of the
following: the 'render' disposition type, the 'application/sdp'
content type, and the MESSAGE method.

A given context is generally (but not necessarily) defined by a
method, a disposition type, and a content type. Support for a
specific context is usually defined within an extension. For
example, the extension for instant messaging in SIP [RFC3428]
mandates support for the MESSAGE method, the 'render' disposition
type, and the 'text/plain' content type.

   Note that, effectively, content types are also supported within a
   context. Therefore, the use of the Accept header field in a 415
   (Unsupported Media Type) response is not enough to describe in
   which contexts a particular content type is supported.

Therefore, support for a particular disposition type within a given
context is typically signalled by the use of a particular method or
an option-tag in a Supported or a Require header field. When support
for a particular disposition type within a context is mandated,
support for a default content type is also mandated (e.g., a UA that
supports the 'session' disposition type in an INVITE request needs to
support the 'application/sdp' content type).

8.2. UA Behavior to Set the 'handling' Parameter

   As stated earlier, the 'handling' Content-Disposition parameter can
   take two values: 'required' or 'optional'. While it is typically
   easy for a UA to decide which type of handling an individual body
   part requires, setting the 'handling' parameter of 'multipart' bodies
   requires extra considerations.

   If the handling of a 'multipart/mixed' body as a whole is required
   for processing its enclosing body part or message, the UA MUST set
   the 'handling' parameter of the 'multipart/mixed' body to 'required'.
   Otherwise, the UA MUST set it to 'optional'. The 'handling'
   parameters of the top-level body parts within the 'multipart/mixed'
   body are set independently from the 'handling' parameter of the
   'multipart/mixed' body. If the handling of a particular top-level
   body part is required, the UA MUST set the 'handling' parameter of
   that body part 'required'. Otherwise, the UA MUST set it to
   'optional'.

      Per the previous rules, a 'multipart/mixed' body whose handling is
      optional can contain body parts whose handling is required. In
      such case, the receiver is required to process the body parts
      whose handling is required if and only if the receiver decides to
      process the optional 'multipart/mixed' body.

      Also per the previous rules, a 'multipart/mixed' body whose
      handling is required can contain only body parts whose handling is
      optional. In such case, the receiver is required to process the
      body as a whole but, when processing it, the receiver may decide
      (based on its local policy) not to process any of the body parts.

   The 'handling' parameter is a Content-Disposition parameter.
   Therefore, in order to set this parameter, it is necessary to provide
   the 'multipart/mixed' body with a disposition type. Per [RFC3261],
   the default disposition type for 'application/sdp' is 'session' and
   for other bodies is 'render'. UAs SHOULD assign 'multipart/mixed'
   bodies a disposition type of 'render'.

      Note that the fact that 'multipart/mixed' bodies have a default
      disposition type of 'render' does not imply that they will be
      rendered to the user. The way the body parts within the
      'multipart/mixed' are handled depends on the disposition types of
      the individual body parts. The actual disposition type of the
      whole 'multipart/mixed' is irrelevant. The 'render' disposition
      type has been chosen for 'multipart/mixed' bodies simply because
      'render' is the default disposition type in SIP.

If the handling of a 'multipart/alternative' body as a whole is
required for processing its enclosing body part or message, the UA
MUST set the 'handling' parameter of the 'multipart/alternative' body
to 'required'. Otherwise, the UA MUST set it to 'optional'. The UA
SHOULD also set the 'handling' parameter of all the top-level body
part within the 'multipart/alternative' to 'optional'.

   The receiver will process the body parts based on the handling
   parameter of the 'multipart/alternative' body. The receiver will
   ignore the handling parameters of the body parts. That is why
   setting them to 'optional' is at the "SHOULD" level and not at the
   "MUST" level -- their value is irrelevant.

The UA MUST use the same disposition type for the 'multipart/
alternative' body and all its top-level body parts.

If the handling of a 'multipart/related' body as a whole is required
for processing its enclosing body part or message, the UA MUST set
the 'handling' parameter of the 'multipart/related' body to
'required'. Otherwise, the UA MUST set it to 'optional'. The
'handling' parameters of the top-level body parts within the
'multipart/related' body are set independently from the 'handling'
parameter of the 'multipart/related' body. If the handling of a
particular top-level body part is required, the UA MUST set the
'handling' parameter of that body part to 'required'. Otherwise, the
UA MUST set it to 'optional'. If at least one top-level body part
within a 'multipart/related' body has a 'handling' parameter of
'required', the UA SHOULD set the 'handling' parameter of the root
body part to 'required'.

8.3. UA Behavior to Process 'multipart/alternative'

   The receiver of a 'multipart/alternative' body MUST process the body
   based on its handling parameter. The receiver SHOULD ignore the
   handling parameters of the body parts within the 'multipart/
   alternative'.

8.4. UAS Behavior to Report Unsupported Message Bodies

   If a UAS cannot process a request because, in the given context, the
   UAS does not support the content type or the disposition type of a
   body part whose handling is required, the UAS SHOULD return a 415
   (Unsupported Media Type) response even if the UAS supported the
   content type, the disposition type, or both in a different context.

      Consequently, it is possible to receive a 415 (Unsupported Media
      Type) response with an Accept header field containing all the
      content types used in the request.

   If a UAS receives a request with a body part whose disposition type
   is not compatible with the way the body part is supposed to be
   handled according to other parts of the SIP message (e.g., a Refer-To
   header field with a Content-ID URL pointing to a body part whose
   disposition type is 'session'), the UAS SHOULD return a 415
   (Unsupported Media Type) response.

9. Message Body Processing

   This section deals with the processing of message bodies and how that
   processing is influenced by the presence of references to them.

9.1. Background on References to Message Body Parts

   Content-ID URLs allow creating references to body parts. A given
   Content-ID URL [RFC2392], which can appear in a header field or
   within a body part (e.g., in an SDP attribute), points to a
   particular body part. The way to handle that body part is defined by
   the field the Content-ID URL appears. For example, the extension to
   refer to multiple resources in SIP [RFC5368] places a Content-ID URL
   in a Refer-To header field. Such a Content-ID URL points to a body
   part that carries a URI list. In another example, the extension for
   file transfer in SDP [RFC5547] places a Content-ID URL in a 'file-
   icon' SDP attribute. This Content-ID URL points to a body part that
   carries a (typically small) picture.

9.2. UA Behavior to Generate References to Message Bodies

   UAs MUST only include forward references in the SIP messages they
   generate. That is, an element in a SIP message can reference a body
   part only if the body part appears after the element. Consequently,
   a given body part can only be referenced by another body part that
   appears before it or by a header field. Having only forward
   references allows recipients to process body parts as they parse
   them. They do not need to parse the remainder of the message in
   order to process a body part.

      It was considered to only allow (forward) references among body
      parts that belonged to the same 'multipart/related' [RFC2387]
      wrapper. However, it was finally decided that this extra
      constraint was not necessary.

9.3. UA Behavior to Process Message Bodies

   In order to process a message body or a body part, a UA needs to know
   whether a SIP header field or another body part contains a reference
   to the message body or body part (e.g., a Content-ID URL pointing to
   it). If the body part is not referenced in any way (e.g., there are

no header fields or other body parts with a Content-ID URL pointing
to it), the UA processes the body part as indicated by its
disposition type and the context in which the body part was received.

If the SIP message contains a reference to the body part, the UA
processes the body part according to the reference. If the SIP
message contains more than one reference to the body part (e.g., two
header fields contain Content-ID URLs pointing to the body part), the
UA processes the body part as many times as references are.

   Note that, following the rules in [RFC3204], if a UA does not
   understand a body part whose handling is optional, the UA ignores
   it. Also note that the content indirection mechanism in SIP
   [RFC4483] allows UAs to point to external bodies. Therefore, a UA
   receiving a SIP message that uses content indirection could need
   to fetch a body part (e.g., using HTTP [RFC2616]) in order to
   process it.

9.4. The 'by-reference' Disposition Type

   Per the rules in Section 9.3, if a SIP message contains a reference
   to a body part, the UA processes the body part according to the
   reference. Since the reference provides the context in which the
   body part needs to be processed, the disposition type of the body
   part is irrelevant. However, a UA that missed a reference to a body
   part (e.g., because the reference was in a header field the UA did
   not support) would attempt to process the body part according to its
   disposition type alone. To keep this from happening, we define a new
   disposition type for the Content-Disposition header field: by-
   reference.

   A body part whose disposition type is 'by-reference' needs to be
   handled according to a reference to the body part that is located in
   the same SIP message as the body part (given that SIP only allows
   forward references, the reference will appear in the same SIP message
   before the body part). A recipient of a body part whose disposition
   type is 'by-reference' that cannot find any reference to the body
   part (e.g., the reference was in a header field the recipient does
   not support and, thus, did not process) MUST NOT process the body
   part. Consequently, if the handling of the body part was required,
   the UA needs to report an error.

   Note that extensions that predate this specification use
   references to body parts whose disposition type is not 'by-
   reference'. Those extensions use option-tags to make sure the
   recipient understands the whole extension and, thus, cannot miss
   the reference and attempt to process the body part according to
   its disposition type alone.

10. Guidelines to Authors of SIP Extensions

   These guidelines are intended for authors of SIP extensions that
   involve, in some way, message bodies or body parts. These guidelines
   discuss aspects that authors of such extensions need to consider when
   designing them.

   This specification mandates support for 'multipart/mixed' and
   'multipart/alternative'. At present, there are no SIP extensions
   that use different 'multipart' subtypes such as parallel [RFC2046] or
   digest [RFC2046]. If such extensions were to be defined in the
   future, their authors would need to make sure (e.g., by using an
   option-tag or by other means) that entities receiving those
   'multipart' subtypes were able to process them. As stated earlier,
   UAs treat unknown 'multipart' subtypes as 'multipart/mixed'.

   Authors of SIP extensions making use of 'multipart/related' bodies
   have to explicitly address the handling of the disposition types of
   the body parts within the 'multipart/related' body. Authors wishing
   to make use of 'multipart/related' bodies should keep in mind that
   UAs that do not understand 'multipart/related' will treat it as
   'multipart/mixed'. If such treatment by a recipient is not
   acceptable for a particular extension, the authors of such extension
   would need to make sure (e.g., by using an option-tag or by other
   means) that entities receiving the 'multipart/related' body were able
   to correctly process them.

   As stated earlier, SIP extensions can also include 'multipart' MIME
   bodies in responses. Hence, a response can be extremely complex and
   the UAC receiving the response might not be able to process it
   correctly. Because UACs receiving a response cannot report errors to
   the UAS that generated the response (i.e., error responses can only
   be generated for requests), authors of SIP extensions need to make
   sure that requests clearly indicate (e.g., by using an option-tag or
   by other means) the capabilities of the UAC so that UASs can decide
   what to include in their responses.

11. Security Considerations

   This document specifies how SIP entities handle message bodies.
   [RFC3261] discusses what type of information is encoded in SIP
   message bodies and how SIP entities can protect that information. In
   addition to the hop-by-hop security SIP can provide, SIP can also
   secure information in an end-to-end fashion. SIP message bodies can
   be end-to-end encrypted and integrity protected using S/MIME
   [RFC3851], as described in [RFC3261].

12. IANA Considerations

   This document contains two actions that have been completed by IANA.

12.1. Registration of the 'by-reference' Disposition Type

   This document defines a new Content-Disposition header field
   disposition type (by-reference) Section 9.4. This value has been
   registered in the IANA registry for Mail Content Disposition Values
   with the following description:

        by-reference   The body needs to be handled according to a
                       reference to the body that is located in
                       the same SIP message as the body.

12.2. Update of the 'handling' Parameter Registration

   References to this specification, to [RFC3204], and to [RFC3459] have
   been added to the entry for the Content-Disposition 'handling'
   parameter in the Header Field Parameters and Parameter Values
   registry. The following is the resulting entry.

| | Predefined | |
Header Field	Parameter Name	Values	Reference
Content-Disposition	handling	Yes	[RFC3204] [RFC3261]
			[RFC3459] [RFC5621]

13. Acknowledgements

   The ideas in this document were originally discussed with Paul
   Kyzivat. Christer Holmberg, Francois Audet, Dan Wing, Adam Roach,
   Keith Drage, and Dale Worley provided comments on it. Dave Crocker
   performed a thorough review on the whole document.

14. References

14.1. Normative References

   [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
              Extensions (MIME) Part One: Format of Internet Message
              Bodies", RFC 2045, November 1996.

   [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
              Extensions (MIME) Part Two: Media Types", RFC 2046,
              November 1996.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2183] Troost, R., Dorner, S., and K. Moore, "Communicating
              Presentation Information in Internet Messages: The
              Content-Disposition Header Field", RFC 2183, August 1997.

   [RFC2387] Levinson, E., "The MIME Multipart/Related Content-type",
              RFC 2387, August 1998.

   [RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource
              Locators", RFC 2392, August 1998.

   [RFC3204] Zimmerer, E., Peterson, J., Vemuri, A., Ong, L., Audet,
              F., Watson, M., and M. Zonoun, "MIME media types for ISUP
              and QSIG Objects", RFC 3204, December 2001.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              June 2002.

   [RFC3459] Burger, E., "Critical Content Multi-purpose Internet Mail
              Extensions (MIME) Parameter", RFC 3459, January 2003.

   [RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail
              Extensions (S/MIME) Version 3.1 Message Specification",
              RFC 3851, July 2004.

   [RFC3959] Camarillo, G., "The Early Session Disposition Type for the
              Session Initiation Protocol (SIP)", RFC 3959,
              December 2004.

   [RFC4483] Burger, E., "A Mechanism for Content Indirection in
              Session Initiation Protocol (SIP) Messages", RFC 4483,
              May 2006.

14.2. Informative References

   [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
              Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
              Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C.,
              and D. Gurle, "Session Initiation Protocol (SIP) Extension
              for Instant Messaging", RFC 3428, December 2002.

   [RFC4289]   Freed, N. and J. Klensin, "Multipurpose Internet Mail
               Extensions (MIME) Part Four: Registration Procedures",
               BCP 13, RFC 4289, December 2005.

   [RFC4566]   Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
               Description Protocol", RFC 4566, July 2006.

   [RFC4662]   Roach, A., Campbell, B., and J. Rosenberg, "A Session
               Initiation Protocol (SIP) Event Notification Extension for
               Resource Lists", RFC 4662, August 2006.

   [RFC5368]   Camarillo, G., Niemi, A., Isomaki, M., Garcia-Martin, M.,
               and H. Khartabil, "Referring to Multiple Resources in the
               Session Initiation Protocol (SIP)", RFC 5368,
               October 2008.

   [RFC5547]   Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S.,
               and P. Kyzivat, "A Session Description Protocol (SDP)
               Offer/Answer Mechanism to Enable File Transfer", RFC 5547,
               May 2009.

Author's Address

   Gonzalo Camarillo
   Ericsson
   Hirsalantie 11
   Jorvas 02420
   Finland

   EMail: Gonzalo.Camarillo@ericsson.com