SIP Working Group G. Camarillo
Internet-Draft Ericsson
Expires: November 30, 2007 May 29, 2007

Message Body Handling in the Session Initiation Protocol (SIP) draft-camarillo-sip-body-handling-01.txt

Status of this Memo

Copyright Notice

Abstract

This document clarifies how message bodies are handled in SIP. Additionally, it discusses to which degree SIP user agents need to support MIME (Multipurpose Internet Mail Extensions)-encoding of body parts.

Table of Contents

1. Introduction 3
2. Terminology 3
3. Multipart Message Bodies 3
3.1. General Considerations 4
3.2. Body Generation 5
3.3. UAS Behavior 5
4. Message-body and Body-part Disposition 5
4.1. Body Generation 7
4.2. Body Processing 8
4.3. UAS Behavior 8
5. Guidelines to Authors of SIP Extensions 9
6. Security Considerations 10
7. Acknowledgements 10
8. IANA Considerations 10
9. Normative References 10
Author's Address 11
Intellectual Property and Copyright Statements 12

1. Introduction

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.

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.

Section 4 discusses issues related to the disposition of message bodies and body parts in SIP. Section 3 discusses issues related to the handling of multipart message bodies in SIP.

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].

3. 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. Therefore, this specification updates [RFC3261]'s recommendation regarding support for multipart MIME bodies.

It is expected that most SIP UAs will implement extensions that require them to generate 'multipart/mixed' MIME bodies. An example of such an extension would be the inclusion of location information in an INVITE request. Such an INVITE request would use the 'multipart/mixed' MIME type 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 a number of SIP UAs will need to generate is 'multipart/alternative'. 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. This way, 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 may be optional elements that may 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 may be lost when translating 'text/html' to 'text/plain'.

It is expected that the transition from SDP to new session description protocols is implemented using 'multipart/alternative' bodies. INVITE requests would 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 UASs would use the session description written in SDP. New UASs would use the one written in the newer format.

A number of SIP UAs will also need to generate nested MIME bodies. 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.

3.1. General Considerations

For all MIME-based extensions to work, the recipient needs to, of course, be able to decode the multipart bodies. Therefore, SIP UAs SHOULD be able to parse 'multipart' MIME bodies, including nested body parts. In particular, UAs SHOULD support the 'multipart/mixed' and 'multipart/alternative' MIME types. Note that, by default, unknown 'multipart' subtypes are treated as 'multipart/mixed'.

> Note that SIP extensions may also include 'multipart' MIME bodies in responses. That is why both UACs and UASs need to support 'multipart' bodies.

3.2. Body Generation

UAs should avoid unnecessarily nesting body parts. Therefore, UAs SHOULD NOT use a 'multipart' body when there is only one body part. UACs 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). UAs SHOULD NOT nest one 'multipart/alternative' within another.

The body parts within a 'multipart/alternative' MUST all have different content types. That is, no body part within a 'multipart/alternative' can have the same content type as another body part within the same 'multipart/alternative'.

OPEN ISSUE 1: we know that we do not want two SDPs in a 'multipart/alternative', but is this valid generally with any content type? Would it be possible to provide two alternative body parts using the same format and, thus, the same content type but in, say, different languages?

As stated earlier, the mapping between two body parts within a 'multipart/alternative' body may imply information loss. [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.

3.3. UAS Behavior

Section 3.1 recommends that all UAs support 'multipart' bodies. However, if a particular UAS does not support 'multipart' bodies and receives one, the UAS SHOULD return a 415 (Unsupported Media Type) response.

> Note that it is essential that UASs without MIME support are at least able to return an error response when receiving a 'multipart' body. Not being able to signal this type of error could cause serious interoperability problems.

OPEN ISSUE 2: do we need to talk about encrypted body parts?

4. Message-body and Body-part Disposition

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] defines the 'handling' parameter for the Content-Disposition header field. From Section 6 of [RFC3204]:

> "This document also defines a Content Disposition parameter, "handling". The handling parameter, handling-parm, describes how the UAS should react 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 MUST ignore the message body; if it has the value "required", the UAS MUST return 415 (Unsupported Media Type). If the handling parameter is missing, the value "required" is to be assumed."

[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 (User Agent Server) did not understand the content type of the body part, it 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 (User Agent Client) 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 (User Agent) may not support the disposition type. For example, a UA may 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 may support the 'render' disposition type for 'text/plain' and 'text/html' body parts in MESSAGE requests. Additionally, the UA may support the 'session' disposition type for 'application/sdp' body parts in INVITE and UPDATE requests and their responses. However, the UA may not support the 'render' disposition type for 'application/sdp' body parts in MESSAGE requests, even if, in different contexts, the UA supported all 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.

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).

Content-ID URLs (Uniform Resource Locators) are another tool to describe how a body part should be handled. Some extensions use a Content-ID URL [RFC2392], which can appear in a header field or within a body part (e.g., in an SDP attribute), that points to a body part. The way to handle that body part is defined by the field the Content-ID URL appears in and by the disposition type of the body part. For example, the extension to refer to multiple resources in SIP [I-D.ietf-sip-multiple-refer] places a Content-ID URL in a Refer-To header field. Such a Content-ID URL points to a body part whose disposition type is supposed to be 'recipient-list'. In another example, the extension for file transfer in SDP [I-D.ietf-mmusic-file-transfer-mech] places a Content-ID URL in a 'file-icon' SDP attribute. This Content-ID URL points to a body part whose disposition type is supposed to be 'icon'.

4.1. Body Generation

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

If at least one of the body parts within a 'multipart/mixed' body has a 'handling' value of 'required', the UA MUST set the 'handling' parameter of the 'multipart/mixed' body to 'required'. If all the body parts within a 'multipart/mixed' body have a 'handling' value of 'optional', the UA MUST set the 'handling' parameter of the 'multipart/mixed' body to 'optional'.

OPEN ISSUE 3: the handling parameter is a Content-Disposition parameter. Therefore, in order to set the handling parameter, it is necessary to provide the 'multipart' body with a disposition type.

Otherwise, its disposition type would be, by default, render. Which disposition type shall we use for 'multipart/mixed' bodies? Shall we create a new disposition type for this purpose?

> Note that, per [RFC3261], if the Content-Disposition header field is missing, bodies of Content-Type 'application/sdp' imply the disposition 'session', while other content types imply 'render'.

If the handling of a 'multipart/alternative' body is required, the UA MUST set the 'handling' parameter of the 'multipart/alternative' body and to the last body part within the 'multipart/alternative' to 'required'. Additionally, the UAC MUST set the 'handling' parameter of all body parts within the 'multipart/alternative' except the last one to 'optional'. The UA MUST use the same disposition type for the 'multipart/alternative' body and all its body parts.

4.2. Body Processing

UAs process message bodies and body parts in the following way. On receiving a SIP message, if a 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.

OPEN ISSUE 4: this means that a UA would need to parse all body parts to find references between them before being able to fully process them. If we are OK with this, we need to include text here explaining it.

If the SIP message contains a reference to the body part, the UA processes the body part according to the reference and the disposition type of the body part. 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 there are.

Following the rules in [RFC3204], if a UA does not understand a body part whose handling is optional, it ignores it.

4.3. UAS Behavior

If a UAS cannot process a request because, in the given context, it 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.

OPEN ISSUE 5: we could define a new response code (Content or Disposition Type not Supported in this Context) to report known content and disposition types appearing in an unsupported context in order to be more explicit than always using 415. It would avoid receiving a 415 response with an Accept header field containing all the content types used in the request. How useful would this really be?

If a UAS receives a request with a body part whose disposition type is not compatible with the way the body part should 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.

5. 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.

This specification recommends support for 'multipart/mixed' and 'multipart/alternative'. At present, there are no SIP extensions that use different 'multipart' subtypes such as parallel [RFC2046], digest [RFC2046], or related [RFC2387]. 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'.

When a SIP UA receives a header field or an optional body part it does not understand, the UA ignores it. A header field or a body part carrying a reference to another body part (e.g., a Content-ID URL) can influence the way that body part is handled. If a header field or a body part carrying a reference to a body part is not understood and, thus, ignored by its recipient, the body part could be handled in an unintended way. Therefore, authors of SIP extensions that involve references to body parts need to make sure (e.g., by using an option-tag or by other means) that entities processing those extensions understand and process the references.

Additionally, authors of such extensions need to specify the acceptable disposition types of the referenced body part and a default, mandatory to support, content type per disposition type.

As stated earlier, SIP extensions may also include 'multipart' MIME bodies in responses. However, UACs receiving a response cannot

report errors to the UAS that generated the response (i.e., error responses can only be generated for requests). Therefore, 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.

6. Security Considerations

TBD.

7. Acknowledgements

The ideas in this document were discussed with Paul Kyzivat. Christer Holmberg, Francois Audet, and Dan Wing provided comments on this document.

8. IANA Considerations

This document does not contain any IANA actions.

9. 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.

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

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

[I-D.ietf-sip-multiple-refer]
Camarillo, G., "Referring to Multiple Resources in the Session Initiation Protocol (SIP)", draft-ietf-sip-multiple-refer-01 (work in progress), January 2007.

[I-D.ietf-mmusic-file-transfer-mech]
Garcia-Martin, M., "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", draft-ietf-mmusic-file-transfer-mech-00 (work in progress), December 2006.

Author's Address

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).