Network Working Group
Internet Draft

Keith Moore
University of Tennessee
24 October 1993

Multipart/References MIME Content-Type

draft-moore-mime-references-00.txt

1. Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. (The file 1id-abstracts.txt, available via anonymous ftp from nic.ddn.mil, describes the current status of each Internet Draft.) It is not appropriate to use Internet Drafts as reference material or to cite them other than as "work in progress".

2. Abstract

This memo defines a new MIME content-type "multipart/references", which can be used to denote a set of MIME contents, of which any one may reference the others by its MIME content-id. This mechanism may be used by presentation languages that wish to be able to reference MIME objects, or by other applications (file transfer, authentication, delivery reports) which need to supply related information about another MIME object.

3. Introduction

Several new MIME [1] -based applications are being developed which require data that is best represented (within a MIME message) as multiple MIME body parts. Frequently, such applications simply need to "embellish" or provide additional information about one or more other objects, which themselves are naturally represented as MIME body parts. Examples of such applications include the following:

+ File transfer. When using MIME to transfer a file, it is useful to represent the file being transferred as a MIME body part (with appropriate MIME content-type labelling), and provide additional information (ownership, creation date, permissions, icon, or data specific to a particular system) in a separate body part. A MIME mail reader could present the contents of the file being transferred in addition to offering to store the file on the

recipient's file system.

+ Authentication. A separate body part could be used as a certificate of authorship and integrity check, using protocols such as PEM [2,3,4,5].

+ Delivery and receipt notifications. A delivery status notification (or receipt notification) may include both the original message, and a separate body part indicating why the delivery of the message failed. The mechanism defined in this memo could be used to illustrate the relationship between the two.

+ Hypertext systems. It may be desirable to transmit a set of hypertext documents which reference one another, in a single MIME message.

+ Presentation languages. Since MIME says little about how the objects it carries are to be presented, this extension may be used by presentation languages that wish to reference other MIME objects.

Although MIME [5] defines a content-id header which can be used for this purpose, it provides no mechanism by which one body part can reference another. Neither does it provide a mechanism to indicate whether a particular body part should be presented by a mail reader, stored in a file (as in an attachment), or whether the body part is intended to be referenced by another body part.

This memo thus defines:

+ a "multipart/references" content-type to be used to indicate a set of body parts that may reference one another.

+ a new "internal" access-type for the message/external-body content-type, which instructs the mail reader to present another body part contained within the same multipart/references content. (This is intended to be used within multipart/alternative, as a fallback for when better presentation languages are not supported.)

4. The Multipart/References Content-Type

The Multipart/References Content-Type is defined as follows:

```
MIME type name: multipart
MIME subtype name: references
Required parameters: start
Optional parameters: none
Encoding considerations: No content-transfer-encoding may be used.
Security considerations: none
```

NOTE IN DRAFT: this content-type needs a better name. Suggestions are welcome!

The "start" parameter contains the content-id of one of the immediate children of the multipart/references body part being defined (without the enclosing '<' and '>'). It indicates to the mail reader which of the components of the multipart/references should be presented.

When presenting a multipart/references body part, the mail reader first scans all of the components contained within. Then it presents the component whose content-id is indicated by the "start" parameter. The "start" component may then reference other body parts contained within the same, or an enclosing, multipart/references body part. If an external program is responsible for presenting the "start" body part, the mail reader will run it in an environment from which it can obtain the other body parts (and associated headers) by using their content-id.

Any component of a multipart/references body part (not just the "start" component) may reference any other component within the same (or an enclosing) multipart/references body part. (The module charged with presenting the "start" component may cause other components of the multipart/references body part to be presented, which themselves reference other body parts.)

Other than storing the information associated with each component of a multipart/references body part to allow for later retrival, and undoing any content-transfer-encoding applied to the component, no processing is performed on any of the components, before presenting the "start" component.

NOTE: This includes any "composite" content-types such as multipart, message/external-body, or message/rfc822. If such objects are referenced by other components, they will obtain the original headers and/or (decoded) body comprising the multipart, message/external-body, or message/rfc822 body part. The individual components or sub-components of a composite CANNOT be directly referenced by another body part, unless it first parses the components.

A component of a multipart/references body part need not include a content-id header. However, without such a header, there is no way for it to either be the "distinguished" component, or to be refernced by another component. Such components will only be presented by a mail reader that does not implement this specification.

5. The "internal" Access-Type for Message/External-Body

A new access-type is defined for the MIME Message/External-Body content-type. The "internal" access-type indicates that the actual body of this particular message/external-body body part, should be obtained

from another body part contained within an enclosing multipart/references body part. The mandantory parameter for this access-type is:

CONTENT-ID -- the content-id of the body part being referenced

The value associated with the content-id parameter should NOT include the enclosing '<' and '>' required by the content-id body part header.

NOTE IN DRAFT: alternatives for the name "internal" are also solicited!

When the internal access-type is used, the headers that appear in the body of the message/external-body part will normally be similar to the body part headers that appear with the referenced body part. However, the headers need not be the identical, since it may be desirable to present the object slightly differently under different conditions. In any case, when a message/external-body body part is presented, the headers appearing in the body of the message/external-body body part are used, rather than those of the referenced body part.

The "internal" access type is designed to allow "fallback" access to one or more components of a multipart/references body part. For example, a hypothetical application/foo body part might reference other body parts A, B, and C. If the recipient's mail reader doesn't support application/foo, the sender might want her to see A, B, and C anyway. So the message could be constructed as follows. (In this example, A, B, C, and D are the content-ids of the components of the multipart/references body part.)

```
multipart/references ; start=D {
    A: [body-part-A]
    B: [body-part-B]
    C: [body-part-C]
    D: multipart/alternative {
     application/foo (which references A, B, and C internally)
     multipart/mixed
         message/external-body; access-type=internal; content-id=A
         message/external-body; access-type=internal; content-id=B
         message/external-body; access-type=internal; content-id=C
    }
}
```

6. Security considerations

Security considerations are not discussed in this memo.

7. Acknowledgements

This proposal is an amalgam resulting from ideas presented in several

other approaches to this problem, including Harald Tveit Alvestrand's "multipart/related", Dave Crocker's "multipart/header-set", and Rens Troost's "content-disposition" documents. (There are probably others

that I don't recall at the moment.) Discussion on the ietf-822 and info-metamail mailing lists has helped to define the requirements for this proposal.

8. References

[1] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, September 23 1993.
[2] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures". RFC 1421, February 10 1993.
[3] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422, February 10 1993.
[4] D. Balenson, "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers", RFC 1423, February 10 1993.
[5] B. Kaliski, "Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services", RFC 1424, Febryary 10, 1993.

9. Author's address

Keith Moore
University of Tennessee
107 Ayres Hall
Knoxville, TN 37996-1301
USA

email: moore@cs.utk.edu

Appendix. Implementation suggestion

This proposal requires that the "modules" responsible for presenting a body part be able to reference other body parts. The mechanisms by which this might be done vary considerably from one system to another (e.g. logical names on VMS, shared file descriptors on UNIX). Because dissimilar systems often share code nowadays, there is some advantage in having a system-independent way to provide such access. Here is a suggestion for how this might be done in a relatively painless manner.

When a mail reader encounters a multipart/references body part, it proceeds to scan each of the components. Each component's headers are stored in a temporary file, and the body part (after undoing any content-transfer-encoding) is stored in a separate temporary file. The names of the files are based on the content-id of the body part being stored.

Since content-IDs may contain special characters not allowed in a file name, or may be longer than the length of a file name, they may be mapped into legal file names as follows:

1. **Remove the '<' and '>' enclosing the content-id.**
2. **Apply the MD5 digest algorithm (see RFC 1321) to the result.**
3. **Separate the resulting 128-bit digest into four groups of 32 bits each, and exclusive-or together all four groups, producing a 32-bit** result.
4. **Encode the resulting 32-bit quantity as 8 hexadecimal, upper case characters.**
5. **If necessary, truncate the resulting character string so that it fits within the limits of the local file system (leaving room for a four** character filename suffix).
6. **Append the string ".HDR" to produce the filename to contain the header, and ".BDY" for the filename to contain the body part.**
7. **The calling program should remove the temporary files after the sub-program completes its work.**

The author hopes this method will be sufficient for any modern file system. File name collisions are presumed to be unlikely, but paranoid implementors might want to modify the filename suffix should collisions occur.

An application programmer's interface should be provided to handle the details of translating file names and manipulating the resulting files. Such an interface could also be extended "underneath" to provide more efficient access where the operating system supports it, e.g. by sharing memory between processes.