Network Working Group   Jacob Palme
Internet Draft   Stockholm
draft-palme-mhtml-info-01.txt   University/KTH
                                             December 2001

Category-to-be:
Informational
Expires: June 2001

Sending HTML in MIME, an informational supplement to the RFC: MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)

Status of this Memo

1.   Abstract

The memo "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)" [RFC 2557] specifies how to send packaged aggregate HTML objects in MIME format. This memo is an accompanying informational document, intended to be an aid to developers. This document is not an Internet standard.

Issues discussed are implementation methods, caching strategies, problems with rewriting of URIs, making messages suitable both for mailers which can and which cannot handle Multipart/related and handling recipients which do not have full Internet connectivity.

2. Table of Contents

Mailing List Information

Further discussion on this document should be done through the mailing list MHTML@SEGATE.SUNET.SE.

```
To subscribe to this list, send a message to
    LISTSERV@SEGATE.SUNET.SE
which contains the text
SUB MHTML <your name (not your email address)>
```

Archives of this list are available by anonymous ftp from
    FTP://SEGATE.SUNET.SE/lists/MHTML/
The archives are also available by email. Send a message to
LISTSERV@SEGATE.SUNET.SE with the text "INDEX MHTML" to get
a list of the archive files, and then a new message "GET
<file name>" to retrieve the archive files.

Comments on less important details may also be sent to the
editor, Jacob Palme <jpalme@dsv.su.se>.

More information may also be available at URL:
HTTP://dsv.su.se/jpalme/ietf/mhtml.html

3. Introduction

[MHTML] specifies how to send packaged aggregate HTML
objects in MIME multipart format. This memo is an
accompanying informational document, intended to be an aid
to developers. This document is not an Internet standard.

The latest revised version of this document can be find in
plain text
and HTML format at
http://dsv.su.se/jpalme/ietf/mhtml.html#info.

4. Implementation Methods

The [MHTML] standard has been intentionally written to be
implementable both in cases where a HTML document viewer
(web browser) and a program receiving MIME objects, such as
an email program, are combined, and when they are separate
programs. Implementation is of course easier if the
document viewer is combined with the MIME receiving client.

Below are described different implementation methods. Real
implementations may sometimes combine ideas from more than
one of the different methods described below.

Note: Some document viewers can take a whole document of
"Content-Type: message" or "Content-Type: multipart" as one
single file to be displayed. When such viewers are known to
be used, the problems described below become much easier to
handle, just submit the whole combined MIME message as a
single file to the viewer.

4.1 Method 1: Combining Viewer And MIME Receiving Program

This is the architecturally simplest approach. A web-browser with a built in MIME receiving program (such as an email program) will be able to use its own document viewer capabilities to display HTML-formatted messages. Since it is the same program, that program will more easily be able to connect a URL in the HTML text to a body part in the message.

4.2 Method 2: Rewriting The HTML

```
     +----------+                          +--------+
     | Document |                          | Mail   |
     | viewer   |                          | client |
     +-------+--+                          +-+------+
             |                               |
          +--+-------------------------------+--+
          | +----------+  +--+  +--+           |
          | | Start    |  |  |  |  | Related   |    Figure 1
          | | HTML     |  |  |  |  | body part |
          | | document |  |  |  |  | parts     |
          | +----------+  +--+  +--+           |
          +------------------------------------+
```

If the document viewer is separate from the MIME receiving client, the MIME client might turn over the HTML body part to the document viewer and ask it to display it (Figure 1). One way of doing this is to store the HTML body part in a file, and ask the document viewer to display this file. If multipart/related is used, this can be implemented by storing all the body parts within the multipart/related in an otherwise empty folder/directory.

The mail client may have to rewrite the HTML, replacing URI-s with (possibly relative) URL-s which the Document viewer can resolve as file names in the same directory/folder where the HTML document itself is stored when turning it over to the Document viewer. Problems with such rewriting of URIs is discussed in section 5 below.

4.3 Method 3: Using A Translation Table

```
     +----------+                         +--------+
     | Document |                         | Mail   |
     | viewer   |                         | client |
     +-------+--+                         +-+------+
             |                              |
          +--+------------------------------+-+
          | +--------+  +--+  +--+            |
          | | Trans-
```

```
  |  |  |  |  | Related    |        Figure 2
          | | lation |  |  |  |  | body part  |
          | | table  |  |  |  |  | parts      |
          | +--------+  +--+  +--+            |
          +-----------------------------------+
```

An alternative to rewriting the HTML file before turning it over to the Document viewer may be to use a translation table, in case the Document viewer has the capability to use such a table to rewrite URL-s on the fly while displaying the document (Figure 2). This requires that the Document viewer is capable of receiving CID: URL-s and resolving them using this translation table in the same way as for other URL-s.

4.4 **Method 4: Using A Proxy HTTP Server To Retrieve** Referenced Body Parts

```
     +--------+       +-----------+       +--------+
     | Proxy  |       | Data base |       | Mail   |
     | web    |-------| of cached |-------| server |
     | server |       | objects   |       |        |
     +----+---+       +-----------+       +----+---+
          |                                    |
     +----+-----+                         +----+---+
     | Document |                         | Mail   |
     | viewer   |                         | client |
     +-------+--+                         +-+------+
             |                              |
          +--+-----------------------------+-+
          |         Start HTML object         |  Figure 3
          +-----------------------------------+
```

Yet another method is to use a proxy web server, to which the document viewer requests are sent, and which will then use the cached body parts instead of normal web retrieval from the network (Figure 3). If the Document viewer is set to use this proxy server for all URL-s, including CID URL-s, no rewriting of the HTML will be necessary.

4.5 **Method 5: Putting The Mail Client Into A Proxy HTTP** Server

```
      +--------+--------+
      | Proxy  | Mail   |
      |  HTTP  | client |
      | server |        |
      +--------+--------+
               |
         HTTP protocol              Figure 4
               |
          +----+-----+
          | Document |
          | Viewer   |
          +----------+
```

A mail client can also be included in an HTTP server (Figure 4). The user will then not have to install any mail client software in his personal computer; all the mail functionality is mapped on HTTP and HTML elements.

4.6 Other Methods

The mail client and the document viewer can of course communicate in other ways, such as using inter-process communication.

4.7 Combined Methods

Several of the methods described above can also be combined. The mailer might for example display simpler HTML documents itself, but automatically or manually transfer the HTML documents to a separate HTML viewer for more complex documents.

A common practice in HTML viewers is to simply ignore all markups which the viewer does not understand. This practice, if implemented in a mailer with limited HTML viewing capabilities, might mean that the user is shown a very incomplete message without any warning that information is missing. In this case, it is better to give the user some kind of warning, combined with a command to view the letter with a separate HTML viewer, or turn the document over automatically to a separate viewer when the document contains markup which the mailer cannot render itself.

4.8 Communication Between Document Viewer And Mail Client

Many document viewers (web browsers) have API-s to allow other programs to communicate with them. There is however no accepted real or de-facto standard for such API-s, which means that a mail program which relies on such API-s will only be able to use those document viewers, whose API they support.

Note however, that most of the methods described above can be implemented with a very minimal such API. The only API function needed is to be able to tell a document viewer, when it is started, to open a particular file. And this API function is a standardized part of the operating system on most platforms. In particular, method 1 and 3 above uses the functionality that a relative URL is resolved with the location of the base document as base. This means that if the base document is a file, relative URL-s will be resolved as FILE URL-s in the same directory/folder where the HTML document itself is placed.

There is a need for buttons in the Web page which the user can use to get back to the mail program again after reading the mail with the document viewer. A common technique to achieve this is to define a new MIME data type for this button. The document viewer is then configured to transfer control to the mail client when the user pushes this button; i.e. downloads a file of this new MIME type.

5. Problems with Rewriting URIs when Copying HTML Documents

Sending of HTML-formatted messages is based on the assumption that an HTML documents, together with in-line objects like images, applets and frames, can be copied into a MIME message. Such copying may require rewriting of URIs containing references between the different message parts. The MHTML standard [MHTML] has been carefully prepared to allow existing web pages to be copied without such rewriting, through the use of the Content-Location MIME content heading field.

There is however a problem if the source HTML document contains relative URIs in parameters to objects and applets, such as in the example below:

```
From: foo1@bar.net
To: foo2@bar.net
Subject: A simple example
Mime-Version: 1.0
Content-Type: multipart/related;
boundary="boundary-example-1";
                  type=Text/HTML
Content-Base: "http://www.ietf.cnri.reston.va.us"

--boundary-example 1
Content-Type: Text/HTML; charset=US-ASCII

   ... text of the HTML document...
<OBJECT
    CLASSID = "clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
    <PARAM NAME="imageurl" VALUE="image.gif">
</OBJECT>
...etc...

--boundary-example-1
```

```
Content-Location: "image.gif"
Content-Type: IMAGE/GIF
Content-Transfer-Encoding: BASE64

R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk
5
..etc...

--boundary-example-1--

From: foo1@bar.net
To: foo2@bar.net
Subject: A simple example
Mime-Version: 1.0
Content-Type: multipart/related;
boundary="boundary-example-1";
                  type=Text/HTML
Content-Base: "http://www.ietf.cnri.reston.va.us"

--boundary-example 1
Content-Type: Text/HTML; charset=US-ASCII

   ... text of the HTML document...
<OBJECT
    CLASSID = "clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
    <PARAM NAME="imageurl" VALUE="image.gif">
</OBJECT>
...etc...

--boundary-example-1
Content-Location: "image.gif"
Content-Type: IMAGE/GIF
Content-Transfer-Encoding: BASE64

R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk
5
..etc...

--boundary-example-1--
```

Only the object might know that the imageurl parameter is a relative URI. It's nearly impossible for the HTML parser to understand that the parameter is a relative URI. Simply searching for "image.gif" is not robust, as the string "image.gif" may be used elsewhere. URIs in scripts can also have similar problems.

One might envisage even more difficult cases, an applet might take a parameter "subject" and another parameter "range" and when subject="auto" and range="1-5" it could

compute, and try to use auto1.gif, auto2.gif ... auto5.gif
as relative URLs.

Some implementation methods described in section 4 above,
for example method 2 described in section 4.2, may require
rewriting of the URIs in the HTML document.

There is no perfect solution to this problem.

One way of alleviating the problem is to produce the
original document using only absolute URIs, preferably of
the CID type, since they are more easily identifiable.

Another way of alleviating the problem is to make all URIs
and Content-Locations into simple relative URIs containing
file names only (without paths, preferably using a file
name format common to most platforms, i.e. 1-6 ascii
letters or digits, a period, and 1-3 extension ascii
letters or digits). An implementation using method 2
described in section 4.2 above can then just store the
parts as files in an empty directory on the recipient
computer with the Content-Locations as file names. It can
then turn the start HTML file over to a document viewer,
and need not rewrite the URIs at all. This simple variant
of use of the MHTML standard is probably most robust, and
those implementors who can control the production of the
HTML documents to be sent are thus recommended to use this
variant.

6. Caching of Body Parts

Suppose a message contains body parts with the
Content-Location header as defined in [MHTML]. A receiving
agent might then put this body part into a web cache, with
the URI in the Content-Location as its name, so that later
retrievals of this URI use the cached body parts. There is
however no guarantee that such a cached item is correct.
Such caching is thus not recommended for use in other ways
than for resolution of links within one particular MIME
message.

The MHTML standard does not cover links between different
messages, but if you want to implement this, use of Content-
ID and/or Message-ID, rather than Content-Location, is
recommended.

If incoming messages are stored in a store where messages
can be automatically deleted (purged), purging of body
parts should not occur before purging of the whole message,
to which they belong.

If an incoming message contains a body part which is linked via Content-Location, then no HTTP lookup should be performed to check if the body part is recent. The message should thus still contain the old HTML document, even if the HTTP-available document has been revised. (Example: "Here is the weather map of October 29, 1997"). Exception from this is:

(a) If the linked document is not enclosed in the message, but referred
     to via Content-Type: message/external-body, then the latest version
     should be shown using ordinary HTTP caching conventions.

(b) If a new message is sent with a Supersedes reference to the old
     message, the old message should still show the old version of all
     the body parts, but it might be wise to inform the user that a
     superseding message is available.

7. "Save as" Command

Many HTML viewers have a "Save as" command to save an HTML document in a local file. Usually, this command has two variants, "Save as text" which converts the HTML document to plain text before saving it, and "Save as source" which saves the HTML document as an HTML-formatted document.

These two variants may not be enough in the case of MHTML documents. There is a third option, which might be named "Save as aggregate". This option would save the HTML plus all related parts in a file with the Content-Type: Multipart/related. The file would thus begin with the heading of the Multipart/related body part.

There are two variants of this: Saving the document as it looked like when you got it, or saving the document including all inline body parts, even those you had to retrieve from the Internet when showing the message to the user. The second format is of special value, because it provides an archiving format of the full document, allowing the user to view it in the future as it looked like at one particular time, even though web content may change in the future.

Finally, a user may also want to save the e-mail or http heading fields of an incoming message. This is sometimes

the same as "Save as aggregate", but may include additional
body parts before or outside of the mulitpart/related
aggregate.

To indicate whether such a saved document was received by e-
mail or http, it might be saved with an additional
surrounding body part of content-type message/rfc822 or
message/http.

Example, suppose you receive by e-mail the following
message:

```
    MAIL FROM:<alice@bar.net>
    RCPT TO:<bob@foo.net>
    DATA
    From: Alice <alice@bar.net>
    To: Bob <bob2@foo.net>
    Date: 23 Jan 1998 10:51
    Subject: A simple example
    Mime-Version: 1.0
    Content-Type: multipart/related; boundary="boundary-example-1";
                  type="text/html"; start=<foo3@foo1@bar.net>

    --boundary-example-1
       Content-Type: text/html;charset=US-ASCII
       Content-ID: <foo3@foo1@bar.net>

       Here is the IETF logo with white background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo.gif"
        ALT="IETF logo with white background">
       And here is the IETF logo with transparent background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo2e.gif"
       <ALT="IETF logo with transparent background">

    --boundary-example-1
        Content-Location: ietflogo.gif
        Content-Base: http://www.ietf.cnri.reston.va.us/images/
       Content-Type: IMAGE/GIF
       Content-Transfer-Encoding: BASE64


       R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5

       NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHVwbGljYXRpb24gcHJvaGliaXRlZC4A
       etc...

    --boundary-example-1--
```

Saving the above message as text might give the following
file:

```
    From: Alice <alice@bar.net>
    To: Bob <bob2@foo.net>
    Date: 23 Jan 1998 10:51
    Subject: A simple example

       Here is the IETF logo with white background:
       IETF logo with white background
       And here is the IETF logo with transparent
background:
       IETF logo with transparent background
```

Saving the same text as html source might give the
following file:

```
       Here is the IETF logo with white background:
       <IMG
SRC="http://www.ietf.cnri.reston.va.us/images/ietflogo.gif"
        ALT="IETF logo with white background">
       And here is the IETF logo with transparent background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo2e.gif"
       <ALT="IETF logo with transparent background">
```

Saving the same text as aggregate might give the following
file

```
    From: Alice <alice@bar.net>
    To: Bob <bob2@foo.net>
    Date: 23 Jan 1998 10:51
    Subject: A simple example
    Mime-Version: 1.0
    Content-Type: multipart/related; boundary="boundary-example-1";
                  type="text/html"; start=<foo3@foo1@bar.net>

    --boundary-example-1
       Content-Type: text/html;charset=US-ASCII
       Content-ID: <foo3@foo1@bar.net>

       Here is the IETF logo with white background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo.gif"
        ALT="IETF logo with white background">
       And here is the IETF logo with transparent background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo2e.gif"
       <ALT="IETF logo with transparent background">

    --boundary-example-1
```

```
         Content-Location: ietflogo.gif
         Content-Base: http://www.ietf.cnri.reston.va.us/images/
        Content-Type: IMAGE/GIF
        Content-Transfer-Encoding: BASE64


        R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5
        NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHVwbGljYXRpb24gcHJvaGliaXRlZC4A
        etc...

     --boundary-example-1--
```

Saving the same text as archiving aggregate might give the
following file (where the missing body part is fetched
through http and added to the saved file):

```
     From: Alice <alice@bar.net>
     To: Bob <bob2@foo.net>
     Date: 23 Jan 1998 10:51
     Subject: A simple example
     Mime-Version: 1.0
     Content-Type: multipart/related; boundary="boundary-example-1";
                   type="text/html"; start=<foo3@foo1@bar.net>

     --boundary-example-1
        Content-Type: text/html;charset=US-ASCII
        Content-ID: <foo3@foo1@bar.net>

        Here is the IETF logo with white background:
        <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
         ietflogo.gif"
         ALT="IETF logo with white background">
        And here is the IETF logo with transparent background:
        <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
         ietflogo2e.gif"
        <ALT="IETF logo with transparent background">

     --boundary-example-1
         Content-Location: ietflogo.gif
         Content-Base: http://www.ietf.cnri.reston.va.us/images/
        Content-Type: IMAGE/GIF
        Content-Transfer-Encoding: BASE64


        R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5
        NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHVwbGljYXRpb24gcHJvaGliaXRlZC4A
        etc...

     --boundary-example-1
       Content-Location: ietflogo2e.gif
```

```
      Content-Base: http://www.ietf.cnri.reston.va.us/images/
      Content-Type: IMAGE/GIF
      Content-Transfer-Encoding: BASE64


      R0lGODlhGAGgANX/ACkpKTExMTk5OUJCQkpKSlJSUlpaWmNjY2tra3Nzc3t7e4
      SEhIyMjJSUlJycnKWlpa2trbW1tcDAwM7Ozv/eQnNzjHNzlGtrjGNjhFpae1pa
      etc...

    --boundary-example-1--
```

Saving the same message as message might give the following
file:

```
    from:<alice@bar.net>
    To:<bob@foo.net>
    Mime-Version: 1.0
    Content-Type: Message/rfc822;
          boundary="boundary-example-2"

    --boundary-example-2
    From: Alice <alice@bar.net>
    To: Bob <bob2@foo.net>
    Date: 23 Jan 1998 10:51
    Subject: A simple example
    Mime-Version: 1.0
    Content-Type: multipart/related;
boundary="boundary-example-1";
                  type="text/html"; start=<foo3@foo1@bar.net>

    --boundary-example-1
       Content-Type: text/html;charset=US-ASCII
       Content-ID: <foo3@foo1@bar.net>

       Here is the IETF logo with white background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo.gif"
        ALT="IETF logo with white background">
       And here is the IETF logo with transparent background:
       <IMG SRC="http://www.ietf.cnri.reston.va.us/images/
        ietflogo2e.gif"
       <ALT="IETF logo with transparent background">
    --boundary-example-1
        Content-Location: ietflogo.gif
        Content-Base: http://www.ietf.cnri.reston.va.us/images/
       Content-Type: IMAGE/GIF
       Content-Transfer-Encoding: BASE64

       R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5
       NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHVwbGljYXRpb24gcHJvaGliaXRlZC4A
       etc...

    --boundary-example-1--
    --boundary-example-2--
```

8. Recipients which cannot Handle the Multipart/related Content-Type

A message sent according to the specifications in [MHTML] may have recipients, whose mailers cannot handle the Multipart/related Content-Type in the way specified in [MHTML].

According to [MIME1] a mailer which encounters an unknown subtype to Multipart, should handle this as Multipart/mixed.

To improve this, Multipart/alternative can be used as discussed in section 9 of this memo.

Content-Disposition, as specified in [CONDISP] and in [MHTML], section 10, can also be used as an aid to mailers which do not understand Multipart/related.

Captions on images, which are included in the HTML text, might for non-HTML-capable recipients be found in the Content-Description header [CONDISP]. Do not assume, however, that HTML-capable user agents will display the Content-Description header, they may assume that this information is included in the HTML text instead.

9. Use of the Content-Type: Multipart/alternative

If the message is sent to recipients, all of which may not have mailers capable of handling the Text/HTML content-type, then the "Content-Type: Multipart/Alternative" [MIME1] can be used in two ways:

9.1 Multipart/alternative inside Multipart/related

The Multipart/alternative is put inside the "Content-Type Multipart/related", body parts can be specified with "Content-Type: Text/plain" as the first choice, and "Content-Type: Text/HTML" as the second choice.

Example:

```
    Content-Type: Multipart/related;
boundary="boundary-example-1";
```

```
                type=MULTIPART/ALTERNATIVE

       --boundary-example 1
       Content-Type: MULTIPART/ALTERNATIVE
       Boundary: boundary-example-2

          --boundary-example-2
          Content-Type: Text/plain

          ... plain text version of the document for recipients
          whose mailers cannot handle Text/HTML ...

          --boundary-example-2
          Content-Type: Text/HTML; charset=US-ASCII
          Content-ID: content-id-example@example.host

          ... text of the HTML document ...

          --boundary-example-2--
       --boundary-example-1
       Content-Type: Image/GIF

       ... a body part, to which the HTML document has a link  ...
       --boundary-example-1--
```

Note that the type parameter of Multipart/related in this case should be Multipart/alternative and not Text/HTML.

9.2 Multipart/alternative outside Multipart/related

The multipart/alternative is put outside the Multipart/Related, with Multipart/Related as one alternative and Multipart/Mixed as the other alternative. Note however that the [MHTML] does not recommend links from inside Multipart/Related to objects outside of the Multipart/Related, so putting inline images outside the Multipart/Related is not suitable. Instead, such inline images may have to repeated in both branches of the multipart/alternative with this method.

Example:

```
    Content-Type: MULTIPART/ALTERNATIVE
    Boundary: boundary-example-1

    --boundary-example-1
       Content-Type: Multipart/mixed; boundary="boundary-example-3"

       --boundary-example-3
```

```
          Content-Type: Text/plain; charset=US-ASCII

          ... plain text version of the message for recipients
          whose mailers cannot handle Text/HTML ...

       --boundary-example-3
       Content-Type: Image/GIF

          ... A picture associated with the plain text message  ...
       --boundary-example-3--

    --boundary-example-1
       Content-Type: Multipart/related; boundary="boundary-example-1";
                     type=Text/HTML

       --boundary-example 2
          Content-Type: Text/HTML; charset=US-ASCII
          Content-ID: content-id-example@example.host

          ... text of the HTML document ...

       --boundary-example-2
       Content-Type: Image/GIF

          ... a body part, to which the HTML document has a link  ...
       --boundary-example-2--
    --boundary-example-1--
```

9.3 Comparing the Two Methods

When choosing between these two methods of employing multipart/alternative, note the following:

(1) Clients which do not support Multipart/related, and which thus will interpret it as Multipart/mixed, will with choice 9.1 display the inline objects. Thus, a recipient whose mailer can handle image/gif but not multipart/related will still be shown the images, they will not be suppressed by being inside a suppressed branch of the Multipart/alternative.

(2) Choice 9.2 will not show inline images in the Multipart/Related, unless this information is repeated in both branches of the Multipart/Alternative.

A general warning: Some mailers do not support "Content-Type: Multipart/alternative", and may then interpret it as Multipart/mixed, even though support of multipart/alternative is required for MIME conformance.

9.4 Reducing the Download Time

If a message is sent as multipart/alternative, this would normally mean that the mail client downloads both variants, and then shows only one of the to the user. This will thus increase the download time. A way of avoiding this problem is to use the FETCH command of IMAP, which allows a client to download only certain body parts from a multipart message.

10. Writing Readable HTML

An alternative to use of multipart/alternative is to produce HTML code which is is easy to read as it is. This alternative only works if you restrict the use of HTML features, for example if you have a special program to generate the HTML text.

Below is an example of such a readable HTML text:

```
<p>   This is an example of HTML code, which is written
       so as to be readable for people who read it as
       plain text.

<p>   Here is the second paragraph, which contains a
          <a href="cid:456*foo@bar.net"> link
</a>  to a separate body part.
<p>   Here is an embedded picture:

<p>      <img src="cid:123*foo@bar.net" width="13" height="13">

<p>   End of this HTML-formatted message.
```

11. Textual Alternatives to HTML Forms

One important usage of HTML in e-mail is to send forms, which the recipients fill in and return. It is then problematic how to handle recipients whose mailers do not support HTML. One way is to use textual encoding of the forms. This encoding is done so that the user action needed to send in the form is made simple also for those who have only textual e-mail systems. Important is that the textual users are not forced to write complex commands in special command languages. Instead, the form should be written so that the user need only make simple changes to the form before sending it back, like deleting or adding single characters.

Below is an example which shows how this can be done. The main principle is that every line beginning with ";" is an explanation for the reader, and every line beginning with "!" is a text, which the user can convert into a command by just deleting the "!" in front of the line.

The users will thus have to learn a very simple rule of filling in forms: Just delete the "!" in front of your selections.

Technically, the recipient of a filled-in textual form should regard all lines beginning with ";" or "!" as comment, and interpret all other lines as commands.

11.1 Form in HTML Format

```
<FORM action="mailto:meeting-scheduling@ietf.org"
method="POST">

<P>Which meeting date do you prefer?

<P>1 December 1997 <SELECT NAME="19971201">
    <OPTION>Very good
    <OPTION>Good
    <OPTION>Acceptable
    <OPTION>Bad
    <OPTION>Very bad
</SELECT>

<P>7 December 1997 <SELECT NAME="19971207">
    <OPTION>Very good
    <OPTION>Good
    <OPTION>Acceptable
    <OPTION>Bad
    <OPTION>Very bad
</SELECT>

<P>14 December 1997 <SELECT NAME="19971214">
    <OPTION>Very good
    <OPTION>Good
    <OPTION>Acceptable
    <OPTION>Bad
    <OPTION>Very bad
</SELECT>

<P>21 December 1997 <SELECT NAME="19971221">
    <OPTION>Very good
    <OPTION>Good
    <OPTION>Acceptable
    <OPTION>Bad
    <OPTION>Very bad
```

```
</SELECT>

<P>Who should be the chairman?

<P><INPUT TYPE="radio" NAME="chairman" VALUE="Mary">Mary

<P><INPUT TYPE="radio" NAME="chairman" VALUE="John">John

<P>Do you want simultaneous translation during the meeting?

<P><INPUT TYPE="checkbox" NAME="translation"
VALUE="English">To and
from English

<P><INPUT TYPE="checkbox" NAME="translation"
VALUE="French">To and
from French

<P><INPUT TYPE="checkbox" NAME="translation"
VALUE="Japanese">To and
from Japanese

<P>Please propose issues to discuss during the meeting:

<P><TEXTAREA NAME="issues" ROWS=7 COLS=66></TEXTAREA>

<P><INPUT TYPE="submit" NAME="Submit"
VALUE="Submit"><INPUT TYPE="reset" VALUE="Reset">
```

11.2 The same Form In Textual Format

```
; This is a computer-generated form. Please fill it in and return it
; to meeting-scheduler@ietf.org. To fill in the form, just copy its
; text into your reply and remove the exclamation mark (!) in front
; of your choices.

; If your mailer adds ">" or "> " in front of lines, you can keep
; these or remove them as you prefer.

Question 1: Which meeting date do you prefer?

Option 1.1: 1 December 1997
! Very good
! Good
! Acceptable
! Bad
! Very bad

Option 1.2: 7 December 1997
! Very good
! Good
! Acceptable
! Bad
! Very bad

Option 1.3: 14 December 1997
! Very good
! Good
! Acceptable
! Bad
! Very bad

Option 1.4: 21 December 1997
! Very good
! Good
! Acceptable
! Bad
! Very bad

Question 2: Who should be the chairman?
! Mary
! John

Question 3: Do you want simultaneous translation during the
meeting?

Option 3.1: To and from English
! Yes
! No

Option 3.2: To and from French
! Yes
! No

Option 3.3: To and from Japanese
! Yes
! No

Question 4: Please propose issues to discuss during the
meeting.
Write your proposal on the empty lines below.






-- End of Question 4
```

12. Recipient may not have Full Internet Connectivity

The recipient of a message sent by email may not always have full Internet connectivity. The recipient may be behind a gateway or firewall which prohibits or restricts Internet connectivity.

This means that the recipient may not be able to resolve URI-s in an email message, unless the referred-to documents are included in the email message itself. Thus, it is often suitable to include in an email message all documents which are referred to (directly or indirectly) by URI-s in the message. This may of course not always be possible, in some cases the set of referred-to documents (directly or indirectly) may be the whole WWW document space, i.e. millions of documents. A choice must then be made how much to include. Of course, it is most important to include all inline objects, i.e. objects linked by such hyperlinks as IMG, etc., which specify that the linked objects are to be shown to the user immediately.

In the case of ACTION elements in HTML forms, by making these ACTION elements of the "mailto:" URL type, rather than the "http:" URL type, you will enable also recipients without full Internet connectivity to fill in and send in your forms. The HTML specification [HTML2] allows default action when no ACTION element is included, but this default action may not be suitable when sending the HTML document via email. Thus, it is better to always put an explicit ACTION element into HTML forms sent by email.

A disadvantage with the "mailto:" URL as ACTION, however, is that this may not work if the user has not specified his e-mail address in the preferences of this HTML viewer. This is common for multi-user workstations.

Including URLs, which have to resolved over the Internet, typically referring to 1x1 pixel transparent images, is a known method for spammers to get information about the recipient of e-mail. It is possible that some mailers will restrict or disallow this in order to stop this method of information-gathering by spammers.

Encoding of Non-Ascii Characters

```
          Displayed text                      Displayed text
                |                                   ^
                V                                   |
          +-------------+                     +-----------------+
          | HTML editor |                     | HTML viewer     |
          |             |                     | or Web browser  |
          +-------------+                     +-----------------+
              |                                       ^
              V                                       |
          HTML markup                             HTML markup
              |                                       ^
              V                                       |
   +---------+ +---------------+       +-------------+ +---------------+
   | MIME    | | MIME content- |       | MIME        | | MIME content- |
   | encap-  | | transfer-     |       | heading     | | transfer-     |
   | sulator | | encoder       |       | interpreter | | decoder       |
   +---------+ +---------------+       +-------------+ +---------------+
     |              |                            ^              ^
     V              V         +-----------+      |              |
MIME heading + MIME content->| Transport |->MIME heading + MIME content
                              +-----------+
```

Figure 5

Definitions (see Figure 5):

Displayed text	A visual representation of the intended text.
HTML markup	A sequence of characters formatted according to the HTML specification [HTML2].
MIME content	A sequence of octets physically forwarded via email, may use MIME content-transfer-encoding as specified in [MIME1].
HTML editor	Software used to produce HTML markup.
MIME content-transfer-encoder	Software used to encode non-US-ASCII charactersr as specified in [MIME1].
MIME content-transfer-decoder	Software used to decode non-US-ASCII characters as specified in [MIME1].
MIME heading interpreter	Software used to interpret the information in MIME headings.
HTML viewer	Software used to display HTML documents to recipients.

Some implementations may have a choice of whether to represent non-ascii characters at the HTML layer (using "&" entity references or numeric character references as defined in [HTML2] section 3.2.1) or at the MIME layer (using Content-Transfer-Encoding as defined in [MIME1] section 5).

In choosing between these two representation methods, note the following effects:

(1) Modifying HTML markup may disrupt security content integrity checksums. If the checksums are computed between the HTML editor and the MIME encapsulator, then making the encoding in the MIME encapsulator will not break the checksums.

(2) The choice of modifying HTML markup may be more suitable for recipients whose mailers do not support MIME.

(3) Using MIME Content-Transfer-Encoding may be more suitable for recipients who have MIME-compliant mailers but do pass the text over to a document viewer (web browser).

14. Conversion from HTTP to MIME

Information received or retrieved using HTTP cannot always be sent unchanged as email using the "Content-Type: Text/HTML", because of the restrictions which MIME places on the format of "Content-Type: Text/HTML". The same problem may occur for documents retrieved via HTTP, which are in other textual formats than HTML. In particular, note the following:

(a) Content-encodings allowed in HTTP, but not allowed in MIME, must be removed.

(b) HTTP allows line breaks as bare CRs or bare LFs or something else, while MIME only allows line breaks as CRLF in subtypes of the Text content-type.

(c) HTTP allows character sets like Unicode-1-1, which do not represent line breaks as CRLFs, such text may have to be rewritten to character sets like Unicode-1-1-UTF-7 in which line breaks are represented as CRLFs.

A good overview of the differences, with regard to the use of "Content-Type: Text", between MIME and HTTP, can be found in [HTTP] appendix C.

If you want to provide web documents, which can be sent through e-mail without modification (which might break integrity checksums), then you SHOULD provide them up in the canonical form, with line breaks as CRLF, and avoid lines longer than 76 characters/line.

If you want to send HTTP unchanged via email, you might
consider using the "Content-Type: Message/HTTP" instead of
the "Content-Type: Text/HTML". Note that with this Content-
Type, the whole object, as sent through HTTP, can be
encoded as a single object with, for example, BASE64
encoding. After decoding of the BASE64, the resulting
object can have HTTP peculiar formats, like single LF or
single CR between lines. However, some mailers may not be
capable of handling the Message/HTTP Content-Type.

Example, the binary part of the following message

```
    Content-Type: message/http
    Content-Transfer-Encoding: base64
```

SFRUUC8xLjEgMjAwIE9LDURhdGU6IFNhdCwgMTQgRmViIDE5OTggMTM6MDM
6MzggR01U

DVNlcnZlcjogQXBhY2hlLzEuMi40DUxhc3QtTW9kaWZpZWQ6IFdlZCwgMjM
gSnVsIDE5
    ... ...

might, when the base64 encoding above is decoded, yield:

```
    HTTP/1.1 200 OK
    Date: Sat, 14 Feb 1998 13:03:38 GMT
    ETag: "43788-124-33d658c5"
    Content-Length: 292
    Accept-Ranges: bytes
    Content-Type: text/html

    ... <HTML data with only LF between lines> ...
```

15. Default Font Size

Many HTML editors and viewers allow the user to specify the
size of the default font (<FONT SIZE=3> or <FONT SIZE="+0">
according to personal wishes, for example 10 pt or 12 pt or
14 **pt depending on eye sight and screen distance. This**
setting should *not* cause a change in the FONT SIZE= value
in the generated HTML which is produced and sent. The
reason for this is that otherwise users may inadvertently
send whole letters with the text in <FONT SIZE=1> or <FONT
SIZE=2>, which may be easy to read for the sender but
difficult to read for some recipients.

Similarly, a user choice of default FONT, to for example

GENEVA or ARIAL, should not cause <FONT FACE=GENEVA> or <FONT FACE=ARIAL> to be sent. User who wish to send e-mail with <FONT SIZE=2> or <FONT FACE=GENEVA> must explicitly specify this, for example using a FONT command in their HTML editor or e-mail text editor.

16. Copyright and Disclaimer

17. Acknowledgments

Harald Tveit Alvestrand, Richard Baker, Dave Crocker, Martin J. Duerst, Roy Fielding, Lewis Geer, Al Gilman, Paul Hoffman, Alexander Hopmann, Mark K. Joseph, Greg Herlihy, Valdis Kletnieks, Daniel LaLiberte, Ed Levinson, Jay Levitt, Albert Lunde, Larry Masinter, Keith Moore, Gavin Nicol, Pete Resnick, Jon Smirl, Einar Stefferud, Jamie Zawinski and several other people have helped us with preparing this memo. I alone take responsibility for any errors which may still be in the memo.

18. References

Temporary note: This list contains some references to Internet drafts. It is anticipated that these Internet drafts will become RFC-s before this memo. The references will then in this memo be changed to refer to the corresponding RFC instead. This list also includes some RFC-s which are not up to date, and which will be replaced by new memos presently in ietf draft status.

Ref.	Author, title
[CONDISP]	R. Troost, S. Dorner: "Communicating Presentation Information in Internet Messages: The Content- Disposition Header", RFC 1806, June 1995.
[HOSTS]	R. Braden (editor): "Requirements for Internet Hosts -- Application and Support", STD-3, RFC 1123, October 1989.
[HTML2]	T. Berners-Lee, D. Connolly: "Hypertext Markup Language - 2.0", RFC 1866, November 1995.
[HTTP]	J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: 2616 Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, RFC 2616, June 1999.

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

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

[MIME1] N. Freed & N. Borenstein: "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 2045, November 1996.

[MIME2] N. Freed & N. Borenstein: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types". RFC 2046, November 1996.

[NEWS] M.R. Horton, R. Adams: "Standard for interchange of USENET messages", RFC 1036, December 1987.

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

[RELURL] R. Fielding: "Relative Uniform Resource Locators", RFC 1808, June 1995.

[MSGFMT] P. Resnick: "Internet Message Format" STD 11, RFC 2822, April 2001.

[SMTP] J. Klensin: "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[URL] T. Berners-Lee, L. Masinter, M. McCahill: "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[URLBODY] N. Freed and Keith Moore: "Definition of the URL MIME External-Body Access-Type", RFC 2017, October 1996.

19. Author's Address

Jacob Palme
Stockholm University and KTH
Electrum 230
S-164 40 Kista, Sweden

Phone: +46-8-16 16 67
Fax: +46-8-783 08 29
Email: jpalme@dsv.su.se

Working group chairman:

Einar Stefferud <stef@nma.com>