Network Working Group
Request for Comments: 3391
Category: Informational

R. Herriot

December 2002

The MIME Application/Vnd.pwg-multiplexed Content-Type

Status of this Memo

Copyright Notice

IESG Note

The IESG believes use of this media type is only appropriate in situations where the producer is fully aware of the capabilities and limitations of the consumer. In particular, this mechanism is very dependent on the producer knowing when the consumer will need a particular component of a multipart object. But consumers potentially work in many different ways and different consumers may need different things at different times. This mechanism provides no means for a producer to determine the needs of a particular consumer and how they are to be accommodated.

Alternative mechanisms, such as a protocol based on BEEP which is capable of bidirectional communication between the producer and consumer, should be considered when the capabilities of the consumer are not known by the producer.

Abstract

The Application/Vnd.pwg-multiplexed content-type, like the Multipart/Related content-type, provides a mechanism for representing objects that consist of multiple components. An Application/Vnd.pwg-multiplexed entity contains a sequence of chunks. Each chunk contains a MIME message or a part of a MIME message. Each MIME message represents a component of the compound object, just as a body part of a Multipart/Related entity represents a component. With a Multipart/Related entity, a body part and its reference in some other body part may be separated by many octets. With an Application/Vnd.pwg-multiplexed entity, a message and its reference in some other message can be made quite close by chunking the message containing the reference. For example, if a long message contains

references to images and the producer does not know of the need for each image until it generates the reference, then Application/Vnd.pwg-multiplexed allows the consumer to process the reference to the image and the image before it consumes the entire long message. This ability is important in printing and scanning applications. This document defines the Application/Vnd.pwg-multiplexed content-type. It also provides examples of its use.

Table of Contents

1. Introduction

The simple MIME content-types, such as "text/plain" provide a mechanism for representing a simple object, such as a text document. The Multipart/Related [RFC2387] content-type provides a mechanism for representing a compound object, such as a text document with two gif images.

A compound object consists of multiple components. One such component is the root component, which contains references to other components of the compound object. These components may, in turn, contain references to other components of the compound object. For example, a compound object could consist of a root html text component and two gif image components -- each referenced by the root component.

A compound object and a component are both abstractions. For transmission over the wire or writing to storage, each needs a representation. A "Multipart/Related entity" is one possible representation of a compound object, and a "body part" is one possible representation of a component.

However, the Multipart/Related content-type is not a good solution for applications that require each component to be close to its corresponding reference in the root component. This document defines a new MIME content-type Application/Vnd.pwg-multiplexed that provides a better solution for some applications. The Application/Vnd.pwg-multiplexed content-type, like the Multipart/Related content-type, provides a common mechanism for representing a compound object. A Multipart/Related entity consists of a sequence of body parts separated by boundary strings. Each body part represents a component of the compound object. An Application/Vnd.pwg-multiplexed entity consists of a sequence of chunks, each of whose length is specified in the chunk header. Each chunk contains a message or a part of a message. Each message represents a component of the compound object. Chunks from different messages can be interleaved. HTTP is the typical transport for an Application/Vnd.pwg-multiplexed entity over the wire. An Application/Vnd.pwg-multiplexed entity could be stored in a Microsoft HTML (message/rfc822) file whose suffix is .mht.

The following paragraphs contain three examples of applications. For each application, there is a discussion of its solution with the Application/Vnd.pwg-multiplexed content-type, the Multipart/Related content-type and BEEP [RFC3080].

Example 1: a printing application. A Producer creates a print stream that consists of a very long series of page descriptions, each of which references one or more images. The root component is the long series of page descriptions. An image may be referenced from multiple pages descriptions, and there is a mechanism to indicate when there are no additional references to an image (i.e., the image is out of scope). The Producer does not know what images to include with a page until it generates that page. The Consumer is presumed to have enough storage to hold all in-scope images and enough of the root component to process at least one page. The Producer doesn’t need any knowledge of the Consumer’s storage capabilities in order to create an entity that the Consumer can successfully process. However, the Producer needs to be prudent about the number of images that are in-scope at any time. Of course, a malicious Producer may try to exceed the storage capabilities of the Consumer, and the Consumer must guard against such entities (see section 6). Here are ways to represent this compound object.

With the Application/Vnd.pwg-multiplexed content-type, each image is a message and the root component is a message. The Producer breaks the root component message into chunks with each image message occurring shortly before its first reference. When the Consumer encounters a reference, it can assume that it has already received the referenced image in an earlier chunk.

With the Multipart/Related content-type, each image must either precede or follow the root component.

If images follow the root component, the Consumer must read all remaining pages of the root component before it can print the first page that references such images. The Consumer must wait to print such a page until it has received the entire root component, and the Consumer may not have the space to hold the remaining pages.

If images precede the root component, the Producer must determine and send all such images before it sends the root component. The Consumer must, in the best case, wait some additional time before it receives the first page of the root component. In the worse case, the Consumer may not have enough storage for all the images.

The Multipart/Related solution is not a good solution because of the wait time and because, in some cases, the Consumer may not have sufficient storage for all of the images.

With BEEP, the images and root component can be sent in separate channels. The Producer can push each image when it encounters the first reference or the Consumer can request it when it encounters the first reference. The over-the-wire stream of octets is similar to an Application/Vnd.pwg-multiplexed entity. However, there is a substantial difference in behavior for a printing application. With the Application/Vnd.pwg-multiplexed content-type, the Producer puts each image message before its first reference so that when the Consumer encounters a reference, the image is guaranteed to be present on the printer. With BEEP, if the Consumer pulls the image, the Consumer has to wait while the image comes over the network. If the Producer pushes the image, BEEP may put the image message after its first reference and the Consumer may still have to wait for the image. A high-speed printer should not have to risk waiting for images; otherwise it cannot run at full speed.

Example 2: a scanning (fax-like) application. The Producer is a scanner, which scans pages and sends them along with a vnd.pwg-xhtml-print+xml root component that contains references to each page

image. Each page is referenced exactly once in the root-component. The Consumer is a printer that consumes vnd.pwg-xhtml-print+xml entities and their attachments. That is, the Consumer is not limited to print jobs that come from scanners. A Producer and Consumer are each presumed to have enough storage to hold a few page images and most if not all of the root component. The Producer doesn't need any additional knowledge of the Consumer's storage capabilities in order to create an entity that the Consumer can successfully process. Of course, a malicious Producer may try to exceed the storage capabilities of the Consumer and the Consumer must guard against such entities (see section 6). Here are ways to represent this compound object.

With the Application/Vnd.pwg-multiplexed content-type, each page image is a message and the root component is a message. The Producer breaks the root component message into chunks with each image message just before or just after its reference.

With the Multipart/Related content-type, the images cannot precede the root component because the Consumer might not have enough space to store them until the root component arrived. In this case, the printer could fail to print the job correctly and the Producer might not know. Therefore the images must follow the root component, and the Producer must scan all pages before it can send the first page. At the very least, this solution delays the printing of the pages until all have been scanned. In the worst case, the Producer does not have sufficient memory to buffer the images, and the job fails.

With BEEP, the issues are the same as in the previous example, except that speed is not as important in this case. So BEEP is a viable alternative for this example.

Example 3: a printing application. A Producer creates a print stream that consists of a series of pages, each of which references zero or more images. Each image is referenced exactly once. The Producer does not know what images to include with a page until it generates that page, and the Producer doesn't know the layout details; the Consumer handles layout. The Producer has enough storage to send the root component and all images. However, it may not have enough storage to hold the entire root component or all octets of any of the images. The Consumer is presumed to have enough storage to render the root component and to render each image. It may not have enough storage to hold the entire root component or all octets of any of the images. The Producer doesn't determine the Consumer's storage capabilities. Rather it arranges the components so that the Consumer is mostly likely to succeed. Of course, a malicious Producer may try

to exceed the storage capabilities of the Consumer, and the Consumer must guard against such entities (see section 6). Here are ways to represent this compound object.

With the Application/Vnd.pwg-multiplexed content-type, each image is a message and the root component is a message. The Producer breaks the root component message into chunks with each image message just after its reference. The references appear first so that the Consumer knows the location of each image before it processes the image. This strategy minimizes storage needs for Producer and Consumer and provides a good strategy in case of failure. Here are the cases to consider.

a) When the document consists of vertically aligned blocks where each block contains either lines of text or a single image, the sequence of chunks is the same as the sequence of printable blocks, thus minimizing Consumer buffering needs.

b) When a block can contain N side-by-side images, the Consumer must buffer N-1 images unless the Producer interleaves the images. If the Producer doesn't interleave the images, and the Consumer runs out of storage before it has received N-1, images, it can print what it has and print the remaining images below; not what the Producer intended, but better than nothing. If the Producer interleaves images, and the Consumer runs out of storage before it has received the bands of N images, the Consumer would print portions of images interleaved with portions of other images. So, a Producer should not interleave images.

c) When a block contains text and image side-by-side (i.e., run-around text), there are additional buffering requirements. When the Consumer processes the text that follows the reference, it will place some of it next to the image (run-around text) and will place the remaining text after the image. The Producer doesn't know where the run-around ends, and thus doesn't know where to end the text chunk and start the image chunk. If the Producer ends the text too soon, then the Consumer either has to process the entire image (if it has enough storage) in order to get the remaining run-around text, or it ends the run-around text prematurely. If the Producer ends the text too late, then the Consumer may have to store too much text and possibly put the image later than the Producer requested. Because text data requires significantly less storage than image data, a good strategy for Producer is to err on the side of sending too much rather than too little text before the image data.

d) When a block contains text and multiple side-by-side images, the problem becomes a combination of items b) and c) above.

The Application/Vnd.pwg-multiplexed content-type can be made to work in this example, but a Consumer must have failure strategies and the result may not be quite what the producer intended. With the Multipart/Related content-type, the images cannot precede the root component because the Consumer might not have enough space to store them until the root component arrived. Also, the images cannot follow the root component because the Consumer might not have enough storage for the root component before the first image arrives. So the Multipart/Related content-type is not an acceptable solution for this example.

With BEEP, the Producer can send the root component on channel 1 and the Consumer can request images on even numbered channels when it encounters a reference. This solution allows more flexibility than the Application/Vnd.pwg-multiplexed content-type. If there are side-by-side images and/or run-around text, the Consumer can request bands of each image or run-around text over separate channels.

In all of these examples, the Application/Vnd.pwg-multiplexed content-type provides a much better solution than Multipart/Related. However, it is evenly matched with BEEP. For applications where speed is important and ordering of the chunks is important in order to avoid printing delays, the Application/Vnd.pwg-multiplexed content-type is best. For applications, where the Consumer needs more control over the ordering of received octets, BEEP is best.

2. Terminology

This document uses some of the MIME terms that are defined in [RFC2045]. The following are the terms used in this document:

Entity: the headers and the content. In this document, the term "entity" describes all the octets that represent a compound object.

Message: an entity as in [RFC2045]. In this document, the term "message" describes all octets that represent one component of a compound object. That is, it has MIME headers and content.

Body Part: an entity inside a multipart. That is, a body part is the headers and content (i.e., octets) between the multipart boundary strings not including the CRLF at the beginning and end. This document never uses "entity" to mean "body part".

Headers: the initial lines of an entity, message or body part. An empty line (i.e., two adjacent CRLFs) terminates the headers. Sometimes the term "MIME header" is used instead of just "header".

Content: the part of an entity, message or body part that follows the headers (i.e., follows the two adjacent CRLFs). The content of a body part ends at the octet preceding the CRLF before the multipart boundary string. The content of a message ends at the octets specified by the length field in the Chunk Header.

This document uses the following additional terms.

Chunk: a chunk of data, consisting of a chunk header, a chunk payload and a CRLF.

Chunk Header: the first line of a chunk. The line consists of the "CHK" keyword, the message number, the length and the continuation indicator, each separated by a single space character (ASCII 32). A CRLF terminates the line. Each message in an Application/Vnd.pwg-multiplexed entity has a message number that normally differs from the message numbers of all other messages in the Application/Vnd.pwg-multiplexed entity. The message number 0 is reserved for final Chunk Header in the Application/Vnd.pwg-multiplexed entity.

Chunk Payload: the octets between the Chunk Header and the Chunk Header of the next chunk. The length field in the header's length field specifies the number of octets in the Chunk Payload. The Chunk Payload is either a complete message or a part of a message. The continuation field in the header specifies whether the chunk is the last chunk of the message.

CRLF: the sequence of octets corresponding to the two US-ASCII characters CR (decimal value 13) and LF (decimal value 10) which, taken together, in this order, denote a line break. A CRLF terminates each chunk in order to provide visual separation from the next chunk header.

Consumer: the software that receives and processes a MIME entity, e.g., software in a printer or software that reads a file.

Producer: the software that creates and sends a MIME entity, e.g., software in a scanner or software that writes a file.

3. Details

The Application/Vnd.pwg-multiplexed content-type, like Multipart/Related, is intended to represent a compound object consisting of several inter-related components. This document does not specify the representation of these relationships, but [RFC2557] contains examples of Multipart/Related entities that use the Content-ID and Content-Location headers to identify body parts and URLs (including the "cid" URL) to reference body parts. It is expected that Application/Vnd.pwg-multiplexed entities would use the patterns described in [RFC2557].

For an Application/Vnd.pwg-multiplexed entity, there is one parameter for the Content-Type header. It is a "type" parameter, and it is like the "type" parameter for the Multipart/Related content-type. The value of the "type" parameter must be the content-type of the root message and it effectively specifies the type of the compound object.

An Application/Vnd.pwg-multiplexed entity contains a sequence of chunks. Each chunk consists of a chunk header, a chunk payload and a CRLF.

- The chunk header consists of a "CHK" keyword followed by the message number, the chunk payload length, whether the chunk is the last chunk of a message and, finally, a CRLF. The length field removes the need for boundary strings that Multipart uses. (See section 3.1 for the syntax of a chunk header).

- The chunk payload is a sequence of octets that is either a complete message or a part of a message.

- The CRLF provides visual separation from the following chunk.

Each message represents a component of the compound object, and a message is intended to have exactly the same representation, octet for octet, as a body part of a Multipart/Related entity that represents the same component. When a message is split across multiple chunks, the chunks need not be contiguous.

The contents of an Application/Vnd.pwg-multiplexed entity have the following properties:

1) The first chunk contains a complete or partial message that (in either case) represents the root component of the compound object.

2) Additional chunks contain messages or partial messages that represent some component of the compound object.

3) The final chunk's header contains a message number of 0, a length of 0 and a last-chunk-of-message mark (i.e., the chunk header line is "CHK 0 0 LAST"). The final chunk contains no chunk payload.

4) A message can be broken into multiple parts and each break can occur anywhere within the message. Each part of the message is zero or more bytes in length and each part of the message is the contents of its own chunk. The order of the chunks within the Application/Vnd.pwg-multiplexed entity must be the same as the order of the parts within the message.

5) A message represents a component of a compound object, and it is intended that it have exactly the same representation, octet for octet, as a body part of a Multipart/Related entity that represents the same component. In particular, the message may contain a Content-Type header to specify the content-type of the message content. Also, the message may contain a Content-ID header and/or Content-Location header to identify a message that is referenced from within another message. If a message contains no Content-Type header, then the message has an implicit content-type of "text/plain; charset=us-ascii", cf. [RFC2045].

See section 4 for a discussion displaying an Application/Vnd.pwg-multiplexed entity.

3.1 Syntax of Application/Vnd.pwg-multiplexed Contents

The ABNF [RFC2234] for the contents of an Application/Vnd.pwg-multiplexed entity is:

```
contents = *chunk finalChunk
chunk      = header payload CRLF
header     = "CHK" SP messageNumber SP length SP isMore CRLF
messageNumber   = 1..2147483647
length   = 0..2147483647
isMore       = "MORE" / "LAST"
payload    = *OCTET
finalChunk = finalHeader CRLF
finalHeader  = "CHK" SP "0" SP "0" SP "LAST" CRLF
```

The messageNumber field specifies the message that the chunk is associated with. See the end of this section for more details.

The length field specifies the number of octets in the chunk payload (represented in ABNF as "payload"). The first octet of the chunk payload is the one immediately following the LF (i.e., final octet) of the chunk header. The last octet of the chunk payload is the one immediately preceding the two octets CRLF that end the chunk.

The isMore field has a value of "LAST" for the last chunk of a message and "MORE" for all other chunks of a message.

Normally each message in an Application/Vnd.pwg-multiplexed entity has a unique message number, and a message consists of the concatenation of all the octets from the one or more chunks with the same message number. The isMore field of the chunk header of the last chunk of each message must have a value of "LAST" and the isMore field of the chunk header of all other chunks must have a value of "MORE".

Two or more messages may have the same message number, though such reuse of message numbers is not recommended. The chunks with the same message number represent a sequence of one or more messages where the isMore field of the chunk header of the last chunk of each message has a value of "LAST". All chunks whose isMore field of the chunk header has the value of "MORE" belong to the same message as the next chunk (in sequence) whose isMore field of the chunk header has the value of "LAST". In other words, if two messages have the same message number, the last chunk of the first message must occur before the first chunk of the second message.

The behavior of the Consumer is undefined if the final Chunk (i.e., the Chunk whose chunk header is "CHK 0 0 LAST") occurs before the last chunk of every message occurs.

Two adjacent chunks usually have different message numbers. However, they may have the same message number. If two adjacent chunks have the same message number, the two chunks could be combined into a single chunk, but they need not be combined.

The number of octets in a chunk payload may be zero, and an Application/Vnd.pwg-multiplexed entity may contain any number of chunks with zero octets of chunk payload. For example, the last chunk of each message may contain zero octets for programming convenience. As another example, suppose that a particular compound object format requires that referenced messages occur before the root message. This document requires that the first chunk of an Application/Vnd.pwg-multiplexed entity contain the root message or a

part of it. So, the first chunk contains a chunk payload of zero octets with the first octet of the root message in the second chunk. That is, all of the message headers of the root message are in the second chunk. As an extreme but unlikely example, it would be possible to have a message broken into ten chunks with zero octet chunk payloads in all chunks except for chunks 4 and 7.

3.2 Parameters for Application/Vnd.pwg-multiplexed

This section defines additional parameters for Application/Vnd.pwg-multiplexed.

3.2.1 The "type" Parameter

The type parameter must be specified. Its value is the content-type of the "root" message. It permits a Consumer to determine the content-type without reference to the enclosed message. If the value of the type parameter differs from the content-type of the root message, the Consumer's behavior is undefined.

3.2.2 Syntax

The syntax for "parameter" is:

```
parameter   := "type"  "=" type "/" subtype ; cf. [RFC2045]
```

4. Handling Application/Vnd.pwg-multiplexed Entities

The application that handles the Application/Vnd.pwg-multiplexed entity has the responsibility for displaying the entity. However, Application/Vnd.pwg-multiplexed messages may contain Content-Disposition headers that provide suggestions for the display and storage of a message, and in some cases the application may pay attention to such headers.

As a reminder, Content-Disposition headers [RFC1806] allow the sender to suggest presentation styles for MIME messages. There are two presentation styles, 'inline' and 'attachment'. Content-Disposition headers have a parameter for specifying a suggested file name for storage.

There are three cases to consider for handling Application/Vnd.pwg-multiplexed entities:

a) The Consumer recognizes Application/Vnd.pwg-multiplexed and the content-type of the root. The Consumer determines the presentation style for the compound object; it handles the display of the components of the compound object in the context

of the compound object. In this case, the Content-Disposition header information is redundant or even misleading, and the Consumer shall ignore them for purposes of display. The Consumer may use the suggested file name if the entity is stored.

b) The Consumer recognizes Application/Vnd.pwg-multiplexed, but not the content-type of the root. The Consumer will give the user the choice of suppressing the entire Application/Vnd.pwg-multiplexed entity or treating the Application/Vnd.pwg-multiplexed entity as a Multipart/Mixed entity where each message is a body part of the Multipart/Mixed entity. In this case (where the entity is not suppressed), the Consumer may find the Content-Disposition information useful for displaying each body part of the resulting Multipart/Mixed entity. If a body part has no Content-Disposition header, the Consumer should display the body part as an attachment.

c) The Consumer does not recognize Application/Vnd.pwg-multiplexed. The Consumer treats the Application/Vnd.pwg-multiplexed entity as opaque and can do nothing with it.

5. Examples

This section contains five examples. Each example is a different representation of the same compound object. The compound object has four components: an XHTML text component and three image components. The images are encoded in binary. The string "<<binary data>>" and "<<part of binary data>>" in each example represents all or part of the binary data of each image. Two of the images are potentially side by side and the third image is displayed later in the document. All of the images are identified by Content-Id and two of the images are also identified by a Content-Location. One of the images references the Content-Location.

The first example shows a Multipart/Related representation of the compound object in order to provide a representation that the reader is familiar with. The remaining examples show Application/Vnd.pwg-multiplexed representations of the same compound object. In the second example, each chunk contains a whole message. In the third example, the XHTML message is split across 3 chunks, and these chunks are interleaved among the three image messages. In the fourth example, the XHTML message is split across 4 chunks, and the two side-by-side images are each split across two chunks. The XHTML chunks are interleaved among the image chunks. In the fifth example, there are chunks with empty payloads and adjacent chunks with the same message number.

The last example may seem to address useless cases, but sometimes it is easier to write software if these cases are allowed. For example, when a buffer fills, it may be easiest to write a chunk and not worry if the previous chunk had the same message number. Likewise, it may be easiest to end a message with an empty chunk. Finally, the Application/Vnd.pwg-multiplexed content-type requires that the first chunk be part of the root message. Sometimes, it is more convenient for the Producer if the root message starts after the occurrence of some attachments. Since a chunk can be empty, the first chunk of the root message can be empty, i.e., it doesn't even contain any headers. Then the first chunk contains a part of the root message, but the Producer doesn't generate any octets for that chunk.

Each body part of the Multipart/Related entity and each message of the Application/Vnd.pwg-multiplexed entity contain a content-disposition, which the Consumer uses according to the rules in section 4. Note the location of the content-disposition headers in the examples.

5.1 Example With Multipart/Related

In this example, the compound object is represented as a Multipart/Related entity so that the reader can compare it with the Application/Vnd.pwg-multiplexed entities.

```
Content-Type: multipart/related; boundary="boundary-example";
              type="text/xhtml+xml"

--boundary-example
Content-ID: <49568.44343xxx@foo.com>
Content-Type: application/vnd.pwg-xhtml-print+xml
Content-Disposition: inline

<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
   <body>
      <p>some text
         <img src="cid:49568.45876xxx@foo.com"/>
         <img src="http://foo.com/images/image2.gif"/>
         some more text after the images
      </p>
      <p>some more text without images
      </p>
      <p>some more text
         <img src="cid:49568.47333xxx@foo.com"/>
      </p>
```

```
        <p>some final text
        </p>
      </body>
   </html>
   --boundary-example
   Content-ID: <49568.45876xxx@foo.com>
   Content-Location: http://foo.com/images/image1.gif
   Content-Type: image/gif
   Content-Disposition: attachment

   <<binary data>>
   --boundary-example
   Content-ID: <49568.46000xxx@foo.com>
   Content-Location: http://foo.com/images/image2.gif
   Content-Type: image/gif
   Content-Disposition: attachment

   <<binary data>>
   --boundary-example
   Content-ID: <49568.47333xxx@foo.com>
   Content-Type: image/gif
   Content-Disposition: attachment

   <<binary data>>
   --boundary-example--
```

5.2 Examples with Application/Vnd.pwg-multiplexed

The four examples in this section show Application/Vnd.pwg-multiplexed representations of the same compound object. Note that each CRLF is represented by a visual line break.

5.2.1 Example Where Each Chunk Has a Complete Message

In this example, the compound object is represented as an Application/Vnd.pwg-multiplexed entity. Each chunk contains an entire message, i.e., none of the messages are split across multiple chunks. Each message in this example is identical to the corresponding body part in the preceding Multipart/Relate example.

```
   Content-Type: application/vnd.pwg-multiplexed;
                 type="application/vnd.pwg-xhtml-print+xml"

   CHK 1 550 LAST
   Content-ID: <49568.44343xxx@foo.com>
   Content-Type: application/vnd.pwg-xhtml-print+xml
   Content-Disposition: inline
```

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
   <body>
      <p>some text
         <img src="cid:49568.45876xxx@foo.com"/>
         <img src="http://foo.com/images/image2.gif"/>
         some more text after the images
      </p>
      <p>some more text without images
      </p>
      <p>some more text
         <img src="cid:49568.47333xxx@foo.com"/>
      </p>
      <p>some final text
      </p>
   </body>
</html>

CHK 2 6346 LAST
Content-ID: <49568.45876xxx@foo.com>
Content-Location: http://foo.com/images/image1.gif
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 3 6401 LAST
Content-ID: <49568.46000xxx@foo.com>
Content-Location: http://foo.com/images/image2.gif
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 4 7603 LAST
Content-ID: <49568.47333xxx@foo.com>
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 0 0 LAST
```

5.2.2 Example of Chunking the Root Message

In this example, the compound object is represented as an Application/Vnd.pwg-multiplexed entity. The message containing the XHTML component is split into 3 pieces so that the reference to an image is as close as possible to the beginning of the chunk. The chunk containing the referenced image message occurs just before the chunk with the reference. This minimizes the distance between reference and referenced message.

Note that there are other possible arrangements (see the third example in section 5.2.3). For example, a sender could split the XHTML message so that the reference to an image is as close as possible to the end of the chunk. Then the chunk containing the referenced image message should occur just after the chunk with the reference. The sender could mix this strategy with the one used in this example.

```
Content-Type: application/vnd.pwg-multiplexed;
              type=" application/vnd.pwg-xhtml-print+xml"

CHK 1 267 MORE
Content-ID: <49568.44343xxx@foo.com>
Content-Type: application/vnd.pwg-xhtml-print+xml
Content-Disposition: inline

<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
   <body>
      <p>some text

CHK 2 6346 LAST
Content-ID: <49568.45876xxx@foo.com>
Content-Location: http://foo.com/images/image1.gif
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 3 6401 LAST
Content-ID: <49568.46000xxx@foo.com>
Content-Location: http://foo.com/images/image2.gif
Content-Type: image/gif
Content-Disposition: attachment
```

```
<<binary data>>
CHK 1 166 MORE
         <img src="cid:49568.45876xxx@foo.com"/>
         <img src="http://foo.com/images/image2.gif"/>
         some more text after the images
      </p>
      <p>some more text without images
      </p>
      <p>some more text

CHK 4 7603 LAST
Content-ID: <49568.47333xxx@foo.com>
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 1 80 LAST
         <img src="cid:49568.47333xxx@foo.com"/>
      </p>
      <p>some final text
      </p>
   </body>
</html>

CHK 0 0 LAST
```

5.2.3 Example of Chunking the Several Messages

In this example, the compound object is represented as an Application/Vnd.pwg-multiplexed entity. The message containing the XHTML component is split into 4 pieces so that the reference to an image is as close as possible to either the beginning or the end of the chunk. The references to the first and second images closely follow the referenced images. The reference to the third image closely precedes the referenced image. This minimizes the distance between reference and referenced message. In addition, the first two image messages are split into two chunks each.

```
Content-Type: application/vnd.pwg-multiplexed;
              type=" application/vnd.pwg-xhtml-print+xml"

CHK 1 303 MORE
Content-ID: <49568.44343xxx@foo.com>
Content-Type: application/vnd.pwg-xhtml-print+xml
Content-Disposition: inline
```

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
   <body>
      <p>some text

CHK 2 184 MORE
Content-ID: <49568.45876xxx@foo.com>
Content-Location: http://foo.com/images/image1.gif
Content-Type: image/gif
Content-Disposition: attachment

<<part of binary data>>
CHK 3 200 MORE
Content-ID: <49568.46000xxx@foo.com>
Content-Location: http://foo.com/images/image2.gif
Content-Type: image/gif
Content-Disposition: attachment

<<part of binary data>>
CHK 1 78 MORE
         <img src="cid:49568.45876xxx@foo.com"/>
         <img src="http://foo.com/images/image2.gif"/>

CHK 2 6162 LAST
<<part of binary data>>
CHK 3 6201 LAST
<<part of binary data>>
CHK 1 127 MORE
         some more text after the images
      </p>
      <p>some more text without images
      </p>
      <p>some more text
         <img src="cid:49568.47333xxx@foo.com"/>

CHK 4 7603 LAST
Content-ID: <49568.47333xxx@foo.com>
Content-Type: image/gif
Content-Disposition: attachment
```

```
<<binary data>>
CHK 1 41 LAST
      </p>
      <p>some final text
      </p>
   </body>
</html>

CHK 0 0 LAST
```

5.2.4 Example of Chunks with Empty Payloads

This example is identical to the previous one, except that some chunks have a chunk payload of zero octets. The root message starts with a chunk whose payload is empty and every message ends with a chunk whose payload is empty. This example also shows two adjacent chunks that are from the same message. These two chunks could be coalesced into a single chunk, but they might be kept separate for programming convenience.

```
Content-Type: application/vnd.pwg-multiplexed;
              type=" application/vnd.pwg-xhtml-print+xml"

CHK 1 0 MORE

CHK 2 184 MORE
Content-ID: <49568.45876xxx@foo.com>
Content-Location: http://foo.com/images/image1.gif
Content-Type: image/gif
Content-Disposition: attachment

<<part of binary data>>
CHK 3 200 MORE
Content-ID: <49568.46000xxx@foo.com>
Content-Location: http://foo.com/images/image2.gif
Content-Type: image/gif
Content-Disposition: attachment

<<part of binary data>>
CHK 1 303 MORE
Content-ID: <49568.44343xxx@foo.com>
Content-Type: application/vnd.pwg-xhtml-print+xml
Content-Disposition: inline
```

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">
   <body>
      <p>some text

CHK 2 6162 MORE
<<part of binary data>>
CHK 3 6201 MORE
<<part of binary data>>
CHK 2 0 LAST

CHK 3 0 LAST

CHK 1 78 MORE
         <img src="cid:49568.45876xxx@foo.com"/>
         <img src="http://foo.com/images/image2.gif"/>

CHK 4 7603 MORE
Content-ID: <49568.47333xxx@foo.com>
Content-Type: image/gif
Content-Disposition: attachment

<<binary data>>
CHK 4 0 LAST

CHK 1 127 MORE
         some more text after the images
      </p>
      <p>some more text without images
      </p>
      <p>some more text
         <img src="cid:49568.47333xxx@foo.com"/>

CHK 1 41 MORE
      </p>
      <p>some final text
      </p>
   </body>
</html>

CHK 1 0 LAST

CHK 0 0 LAST
```

6. Security Considerations

There are security considerations that pertain to each message of an Application/Vnd.pwg-multiplexed entity. Those security considerations are described by the document that defines the content-type of the message. They are not addressed in this document.

There are also security considerations that pertain to the Application/Vnd.pwg-multiplexed entity as a whole. A Producer that is buggy or malicious may send an Application/Vnd.pwg-multiplexed entity that could cause a Consumer to request more storage than it has, even if it has a large amount of storage. A Consumer must be able to deal gracefully with the following scenarios and combinations of them:

- The chunks of one or more messages are separated by a very large number of octets. In the extreme case some or all of the messages don't terminate, i.e., they don't contain a closing chunk.
- A very large number of messages are started and interleaved before their final chunk occurs.
- A message contains one or more references to other messages that never occur or don't occur for a large number of octets.
- A very large number of referenced messages occur before the Consumer knows that it can discard them.

7. Registration Information for Application/Vnd.pwg-multiplexed

The following form is copied from RFC 1590, Appendix A.

To: iana@iana.org

Subject: Registration of new Media Type application/Vnd.pwg-multiplexed
Media Type name: Application
Media subtype name: Vendor Tree - vnd.pwg-multiplexed
Required parameters: Type, a media type/subtype.
Optional parameters: No optional parameters
Encoding considerations: Each message of an Application/Vnd.pwg-multiplexed entity can be encoded in any manner allowed by the Content-Type of the message. However, using the reasoning of Multipart, the Application/Vnd.pwg-multiplexed entity cannot be encoded. Otherwise, a message would be

encoded twice, once at the message level and once at the Application/Vnd.pwg-multiplexed level.

Security considerations: See section 6 (Security Considerations) of RFC 3391.

Published specification: RFC 3391.

Person & email address to contact for further information:

Robert Herriot
706 Colorado Ave.
Palo Alto, CA 94303
USA
Phone: 1-650-327-4466
Fax: 1-650-327-4466
EMail: bob@herriot.com

8. Acknowledgments

The author gratefully acknowledges the contributions of: Ugo Corda, Dave Crocker, Melinda Sue Grant, Graham Klyne, Carl-Uno Manros, Larry Masinter, Ira McDonald, Chris Newman, Henrik Frystyk Nielsen and Dale R. Worley. In particular, Chris Newman provided invaluable help.

9. References

[RFC1806] Troost, R. and S. Dorner, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC 1806, June 1995.

[RFC1873] Levinson, E. and J. Clark, "Message/External-Body Content-ID Access Type", RFC 1873, December 1995.
Levinson, E., "Message/External-Body Content-ID Access Type", Work in Progress.

[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.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for SyntaxSpecifications: ABNF", RFC 2234, November 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.

[RFC2557] Palme, J., "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML", RFC 2557, March 1999.

[RFC2822] Resnick, P., Editor, "Internet Message Format", RFC 2822, April 2001.

[RFC3080] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

10. Author's Address

Robert Herriot
706 Colorado Ave.
Palo Alto, CA 94303
USA

Phone: 1-650-327-4466
Fax: 1-650-327-4466
EMail: bob@herriot.com

11. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.