The MIME Application/Multiplexed Content-type

draft-herriot-application-multiplexed-03

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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 draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

The Application/Multiplexed content-type, like the Multipart/Related content-type, provides a mechanism for representing objects that consist of multiple components. An Application/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 an Application/Multiplexed entity, a body part and its reference in some other body part may be separated by many octets -- more octets than a memory-constrained device can deal with. With an Application/Multiplexed entity, a chunk can contain part of a message. This allows a message and its reference in some other message to be made quite close -- close enough for a memory-constrained device to deal with. This document defines the Application/Multiplexed content-type. It also provides examples of its use.
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 a 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 components -- each referenced by the root component.

A compound object and a component are both abstractions. For transmission over the wire, 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.

Within a Multipart/Related entity, the number of octets separating a body part from its reference in some other body part is arbitrarily long. For a memory constrained Receiving User Agent, there is a need to keep the number of octets between a body part and its reference below some maximum number. This requirement implies the need to define some unit that represents a part of a component and to be able to interleave these units among those from other components.

This document defines a new MIME content-type Application/Multiplexed that provides a simple solution to this problem. There are compound objects that a Receiving User Agent can successfully receive as an Application/Multiplexed entity but not as a Multipart/Related entity because of memory constraints. It is recognized that are compound objects that a Receiving User Agent can successfully receive via a protocol but not as a Application/Multiplexed entity. However, there are circumstances where it is preferable to transmit a compound object via a stream of octets rather than via a protocol.

The Application/Multiplexed content-type, like the Multipart/Related content-type, provides a common mechanism for representing a compound object. A Multipart/Related entity consists of sequence of body parts separated by boundary strings, and each body part represents a component of the compound object. An Application/Multiplexed entity contains 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.

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.

Receiving User Agent: the software that receives and processes a MIME entity.

This document uses the following additional terms.

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

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 blank. A CR LF terminates the line. Each message in an application/multiplexed entity has a message number that normally differs from the message numbers of all other messages in the application/multiplexed entity. The message number 0 is reserved for final Chunk Header in the application/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 octets. 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.

CR LF: A CR LF terminates each chunk in order to provide visual separation from the next chunk header.
3 Details

The Application/Multiplexed content-type, like Multipart/Related, is intended to represent a compound object consisting of several interrelated 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/Multiplexed entities would use the patterns described in [RFC2557].

For an Application/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/Multiplexed entity contains a sequence of chunks. Each chunk consists of a chunk header, a chunk payload and a CR LF.

- 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 CR LF. 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 CR LF 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/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
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/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.

See section 4 for a discussion displaying an Application/Multiplexed entity.

3.1 Syntax of Application/Multiplexed Contents

The ABNF [RFC2234] for the contents of an Application/Multiplexed entity is:

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

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 CR LF 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/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 Receiving User Agent 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 has occurred.

Two adjacent chunks usually have different message numbers, but 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/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/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/Multiplexed

This section defines additional parameters for Application/Multiplexed.
3.2.1 The "type" Parameter

The type parameter must be specified and its value is the content-type of the "root" message. It permits a MIME Receiving User Agent 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 Receiving User Agent’s behavior is undefined.

3.2.2 Syntax

The syntax for the "parameter" of the Multipart/Interleaved content-type (in addition to the parameters specified for multipart) is:

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

4 Handling Application/Multiplexed Entities

The application that handles the Application/Multiplexed entity has the responsibility for displaying the entity. However, Application/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/Multiplexed entities:

a) The Receiving User Agent recognizes Application/Multiplexed and the content-type of the root. The Receiving User Agent 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 Receiving User Agent shall ignore them for purposes of display. The Receiving User Agent may use the suggested file name if the entity is stored.

b) The Receiving User Agent recognizes Application/Multiplexed, but not the content-type of the root. The Receiving User Agent will give the user the choice of suppressing the entire Application/Multiplexed entity or treating the
Application/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 Receiving User Agent 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 Receiving User Agent should display the body part as an attachment.

c) The Receiving User Agent does not recognize Application/Multiplexed. The Receiving User Agent treats the Application/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/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.

Each body part of the Multipart/Related entity and each message of the Application/Multiplexed entity contain a content-disposition, which the Receiving User Agent 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/Multiplexed entities.

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

--boundary-example
Content-ID: <49568.44343xxx@foo.com>
Content-Type: text/xhtml+xml; charset="us-ascii"
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/Multiplexed

The four examples in this section show Application/Multiplexed representations of the same compound object. Note that each CR LF 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/Multiplexed entity. Each chunk contains an entire message, i.e. none of the messages is 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/multiplexed;
   type=text/xhtml+xml

CHK 1 550 LAST
Content-ID: <49568.44343xxx@foo.com>
Content-Type: text/xhtml+xml;charset="us-ascii"
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>
5.2.2 Example of Chunking the Root Message

In this example, the compound object is represented as an Application/Multiplexed entity. The message that represents the XHTML (root) component is split across multiple chunks. The messages that contain the image components are not split across multiple chunks.

In this example, the compound object is represented as an Application/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-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

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

<<binary data>>

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/Multiplexed. The message that represents the XHTML (root) component is split across multiple chunks. The messages that contain the image components are not split across multiple chunks.

In this example, the compound object is represented as an Application/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. In the former case, the chunk containing the referenced image message occurs just before the chunk with the reference (see the first two images in this example). In the later case, the chunk containing the referenced image message occurs just after the chunk with the reference (see the third image in this example). 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/multiplexed;
   type=text/xhtml+xml

<?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

<<part of binary data>>

<image src="cid:49568.45876xxx@foo.com"/>
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.

```xml
<image 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
        <image 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
Content-ID: <49568.46000xxx@foo.com>
Content-Location: http://foo.com/images/image2.gif
Content-Type: image/gif
Content-Disposition: attachment

Content-ID: <49568.44343xxx@foo.com>
Content-Type: text/xhtml+xml;charset="us-ascii"
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</p>
Receiving User Agent Requirements

See section 4 for details about how a Receiving User Agent processes an Application/Multiplexed entity.

6.1 Storing Application/Multiplexed Entities

The Application/Multiplexed content-type will be used for objects that have internal linkages between the messages. When the objects are stored the linkages may require processing by the application or its receiving user agent.

6.2 Recursion

MIME is a recursive structure. Hence, it is possible for an Application/Multiplexed entity to contain other Application/Multiplexed entities.

6.3 Configuration Considerations

It is suggested that MUAs that use configuration mechanisms of [RFC1524]. For an example, refer to Application/Multiplexed as Application/Multiplexed/<type>, where <type> is the value of the "type" parameter.

7 Security Considerations

Security considerations relevant to Application/Multiplexed are identical to those of the underlying content-type.

8 Registration Information for Application/Multiplexed

The following form is copied from RFC 1590, Appendix A.
To: IANA@isi.edu

Subject: Registration of new Media Type content-type/subtype
Media Type name: Application
Media subtype name: Multiplexed
Required parameters: Type, a media type/subtype.
Optional parameters: No optional parameters
Encoding considerations: Application/Multiplexed content-types cannot have encodings. But the individual messages do have encodings.
Security considerations: Depends solely on the referenced type.
Published specification: RFC-REL (this document).
Person & email address to contact for further information:

Robert Herriot
Xerox Corp.
3400 Hillview Ave, Bldg #1
Palo Alto, CA 94304
USA
Phone: 1-650-813-7696
Email: rherriot@pahv.xerox.com

9 Acknowledgments

The author gratefully acknowledges the contributions of: Ugo Corda, Dave Crocker, 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.

10 References


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

11 Author’s Address

Robert Herriot  
Xerox Corp.  
3400 Hillview Ave, Bldg #1  
Palo Alto, CA 94304  
USA  
Phone: 1-650-813-7696  
Email: rherriot@pahv.xerox.com

12 Full Copyright Statement

Copyright © The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative work that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.
The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.