The MIME Application/BatchBeep Content-type

draft-herriot-application-batchbeep-00

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted. (If this document becomes part of an IETF working group activity, then it will be brought into full compliance with 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.

This Internet-Draft will expire on September 16, 2000.

Copyright Notice

Copyright © The Internet Society (2000). All Rights Reserved.

Abstract

The Application/BatchBeep content-type, like the Multipart/Related content-type, provides a mechanism for representing objects that consist of multiple components. An Application/BatchBeep entity contains the wire representation of the client-to-server part of a BEEP (Blocks Extensible Exchange Protocol) session, which consists of sequence of frames. Each frame contains a message or a part of a message. Each message (other than channel 0 messages) 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 -- more octets than a memory-constrained device can deal with. With an Application/BatchBeep entity, a frame 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/BatchBeep content-type. It also provides examples of its use.

Table of Contents

1 Introduction....................................................3
2 Terminology....................................................3
3 Details........................................................4
   3.1 Parameters for Application/BatchBeep........................7
       3.1.1 The "type" Parameter ..................................7
       3.1.2 Syntax ................................................7
4 Handling Application/BatchBeep Entities.........................7
5 Examples........................................................8
   5.1 Example With Multipart/Related.............................9
   5.2 Example Where Each Frame Has a Complete Message..........10
   5.3 Example of Chunking the Root Body Part...................11
   5.4 Example of Chunking the Several Body Parts...............13
6 Receiving User Agent Requirements................................14
   6.1 Data Requirements.........................................14
   6.2 Storing Application/BatchBeep Entities.....................15
   6.3 Recursion..................................................15
   6.4 Configuration Considerations................................15
7 Security Considerations.........................................15
8 Registration Information for Application/BatchBeep...............15
9 Acknowledgments................................................16
10 References.....................................................16
11 Author’s Address................................................17
12 Full Copyright Statement....................................17
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/BatchBeep that solves this problem.

The Application/BatchBeep 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/BatchBeep entity contains the wire representation of the client-to-server part of a BEEP (Blocks Extensible Exchange Protocol) session, which consists of sequence of frames, each of whose length is specified in the frame header. Each frame contains a message or a part of a message. Each message (other than channel 0 messages) represents a component of the compound object. Frames 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 "entity" is used to describe all the octets used to represent a compound object.

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 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 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.

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

This document uses the following terms from BEEP [RFC3080].

Message: an entity as in [RFC2045]. That is, it has MIME headers and content.

Frame: a chunk of data, consisting of a frame header, a frame payload and a frame trailer.

Frame Header: the first line of a frame. It has information, such as the keyword (e.g. "MSG"), the channel number, the message number, the continuation indicator, the sequence number and the length. This document uses the term "frame header" where [RFC3080] would use the term "header", and it uses the term "header" to mean MIME ([RFC2045]) header.

Frame Payload: the octets between the frame header and frame trailer. The length field in the header’s length field specifies the octets. The frame payload is either a complete message or a part of a message.

Frame Trailer: the last line of a frame. It has only the keyword "END". For symmetry with "frame header", this document uses the term "frame trailer" where [RFC3080] would use the term "trailer".

3 Details

The Application/BatchBeep 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/BatchBeep entities would use the patterns described in 
[RFC2557].

For an Application/BatchBeep 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 body part 
and it effectively specifies the type of the compound object.

An Application/BatchBeep entity contains the wire representation of 
the client-to-server part of a BEEP (Blocks Extensible Exchange 
Protocol) session, which consists of a sequence of frames. Each frame 
consists of a frame header, a frame payload and a frame trailer.

- The frame header is a subset of the frame header of [RFC3080]. 
  Only the "MSG" keyword is allowed. As a reminder, the frame 
  header specifies the channel number, the message number, the 
  frame payload length and whether the frame is the last of a 
  message. The length field removes the need for boundary strings 
  that Multipart uses.

- The frame payload is either a message or a part of a message.

- The frame trailer is a line with keyword "END".

Each message (other than channel 0 messages) 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 frames, the frames need not be 
contiguous.

Channel 0 contains control messages. The first message is a greeting 
message. The second message is a start message which initiates 
channel 1. The first message on channel 1 represents the root 
component. There are additional start messages if the 
Application/BatchBeep content uses other channels. The additional 
channels are all odd numbered, cf. [RFC3080]. The last message is a 
close message on channel 0 that closes channel 0. This message 
signifies the end of the Application/BatchBeep entity.

The "RPY", "ERR" and "ANS" messages that a server would send to a 
client in a BEEP session are not a part of the Application/BatchBeep 
entity.
The contents of an Application/BatchBeep entity have the following properties:

1) An Application/BatchBeep entity contains the wire representation of a BEEP (Blocks Extensible Exchange Protocol) session.

2) The first frame contains a greeting message on channel 0.

3) The second frame contains a start message on channel 0 that opens channel 1.

4) If additional channels are used, there must be a start message on channel 0 before the channel is used

5) The final frame contains a close message on channel 0 that closes channel 0. There are no frames after this message.

6) There may be close messages that close other channels when they are no longer needed. Otherwise, such channels are implicitly closed when channel 0 is closed.

7) Every frame header has the keyword "MSG". Any other keyword is an error.

8) The first message on channel 1 represents the root component of the compound object.

9) An additional message on channel 1 or a message on another odd-numbered channels represents some other component of the compound object.

10) A message can be broken into multiple parts and each break can occur anywhere within the message. Each part of the message is the contents of its own frame. The order of the frames within the Application/BatchBeep entity must be the same as the order of the parts within the message.

11) According to [RFC 3080] a frame of a message can be interleaved with the frames of other messages on other channels, but the frames of two messages on the same channel cannot be interleaved.

12) A message represents a component of a compound object, and it is intended that it be exactly the same representation, octet for octet, as a body part of a Multipart/Related entity that represents the same component. In particular, the body part may contain a Content-Type header to specify the content-type of the message content. Also, the body part may contain a Content-
ID header to identify a message that is referenced from within another message.

See section 3.1 for a discussion displaying a Application/BatchBeep entity.

3.1 Parameters for Application/BatchBeep

This section defines additional parameters for Application/BatchBeep.

3.1.1 The "type" Parameter

The type parameter must be specified and its value is the content-type of the "root" body part. It permits a MIME Receiving User Agent to determine the content-type without reference to the enclosed body part. If the value of the type parameter differs from the content-type of the root body part, the Receiving User Agent’s behavior is undefined.

3.1.2 Syntax

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

\[
\text{parameter} \ := \ "\text{type} \ " \ = \ "\text{type} \ / \ " \ \text{subtype} \ ; \ cf. \ [\text{RFC2045}]\
\]

4 Handling Application/BatchBeep Entities

The application that handles the Application/BatchBeep entity has the responsibility for displaying the entity. However, Application/BatchBeep body parts may contain Content-Disposition headers that provide suggestions for the display and storage of a body part, 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 body parts. 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/BatchBeep entities:

a) The Receiving User Agent recognizes Application/BatchBeep 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/BatchBeep, but not the content-type of the root. The Receiving User Agent will give the user the choice of suppressing the entire Application/BatchBeep entity or treating the Application/BatchBeep 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/BatchBeep. The Receiving User Agent treats the Application/BatchBeep entity as opaque and can do nothing with it.

5 Examples

This section contains four 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. One image is encoded in base64, and the other two images are encoded in binary (for sake of example). Two of the images are potentially side by side, and the third image is displayed later in the document.

The first example shows a Multipart/Related representation of the compound object. The remaining examples show Application/BatchBeep representations of the same compound object. In the second example, each frame contains a whole message. In the third example, the XHTML message is split across 3 frames, and these frames are interleaved among the three image messages. In the fourth example, the XHTML message is split across 4 frames, and the two side-by-side images are each split across two frames. The XHTML frames are interleaved among the image frames.

Each body part contains a content-disposition, which the Receiving User Agent uses according to the rules in section 3.1. 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/Batch 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="cid:49568.46000xxx@foo.com"/>
some more text after the images
</p>
<p>some more text without images</p>
<p>some more text</p>
<img src="cid:49568.47333xxx@foo.com"/>
</p>
<p>some final text</p>
</body>
</html>
--boundary-example
Content-ID: <49568.45876xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: base64
Content-Disposition: attachment

R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHVwbGljYXRpb24gcHJvaGliaXRlZC4Aetc...
--boundary-example
Content-ID: <49568.46000xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-Disposition: attachment

<binary data>
--boundary-example
5.2 Example Where Each Frame Has a Complete Message

In this example, the compound object is represented as an Application/BatchBeep entity. Each frame contains an entire message, i.e. none of the messages is split across multiple frames.

```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
    <img src="cid:49568.45876xxx@foo.com"/>
    <img src="cid:49568.46000xxx@foo.com"/>
    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>
END
```

MSG 1 2 . 551 6346
Content-ID: <49568.45876xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: base64
Content-Disposition: attachment

R01GOD1hGAGgAPEAAPP:///ZRaCgoAAAAACH+PUNvcHlyaWdodCAoQykqMTk5
NSBJRVRGLiBvbmF1dGhvcml6ZWQg2HZVwbGljYXRpb24gcHJvaGliaXRlZC4A

Herriot                      Expires: September 23, 2001
5.3 Example of Chunking the Root Body Part

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

In this example, the compound object is represented as an Application/BatchBeep 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 frame. The frame containing the referenced image message occurs just before the frame 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 5.4). 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 frame. Then the frame containing the referenced image message should occur just after the frame with the reference. The sender could mix this strategy with the one use in this example.

Content-Type: application/batchbeep;
type="text/xhtml+xml"

MSG 1 1 * 1 267
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>
</body>
</html>

MSG 3 1 . 1 6346
Content-ID: <49568.45876xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: base64
Content-Disposition: attachment
R0lGODlhGAGgAPEAAP/////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5NSBJRVRGLiBVbmF1dGhvcml6ZWQgZHwGljYXRpb24gcHJvaGliaXRlZC4Aetc...

MSG 3 2 . 6347 6401
Content-ID: <49568.46000xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-Disposition: attachment

<binary data>

MSG 1 1 * 268 166
  <img src="cid:49568.45876xxx@foo.com"/>
  <img src="cid:49568.46000xxx@foo.com"/>
  some more text after the images
</p>
</body>
</html>

MSG 3 3 . 12748 7603
Content-ID: <49568.47333xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-Disposition: attachment

<binary data>

MSG 1 1 . 434 80
  <img src="cid:49568.47333xxx@foo.com"/>
</p>
</body>
</html>

Herriot
Expires: September 23, 2001 [page 12]
5.4 Example of Chunking the Several Body Parts

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

In this example, the compound object is represented as an Application/BatchBeep 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 frame. In the former case, the frame containing the referenced image message occurs just before the frame with the reference (see the first two images in this example). In the later case, the frame containing the referenced image message occurs just after the frame 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 frames each.

Content-Type: application/batchbeep;
type=text/xhtml+xml

MSG 1 1 * 1 303
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
END

MSG 3 1 * 1 184
Content-ID: <49568.45876xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: base64
Content-Disposition: attachment
R0lGODlhGAGgAPEAAP//////ZRaCgoAAAACH+PUNvcHlyaWdodCAoQykgMTk5
END

MSG 5 1 * 1 200
Content-ID: <49568.46000xxx@foo.com>
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-Disposition: attachment

<first part of the binary data>
END
6 Receiving User Agent Requirements

See section 3.1 for details about how a Receiving User Agent processes a Application/BatchBeep entity.

6.1 Data Requirements

ISSUE: what should be in this section:

MIME-capable mail user agents (MUAs) are required to provide the application:
a) the contents of the MIME entities and the entity Content-\
headers,

b) the parameters of the Application/BatchBeep Content-type header,

c) the correspondence between each body part’s local file name, that
body part’s header data, and, if present, the body part’s content-\
ID

6.2 Storing Application/BatchBeep Entities

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

6.3 Recursion

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

6.4 Configuration Considerations

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

7 Security Considerations

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

8 Registration Information for Application/BatchBeep

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: BatchBeep
Required parameters: Type, a media type/subtype.
Optional parameters: No optional parameters
Encoding considerations: Application/BatchBeep 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, Chris Newman and Henrik Frystyk Nielsen. 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 (1998). 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.