              Returning Values from Forms: multipart/form-data

Status of this Memo

Copyright Notice

1. Abstract

   This specification defines an Internet Media Type, multipart/form-
   data, which can be used by a wide variety of applications and
   transported by a wide variety of protocols as a way of returning a
   set of values as the result of a user filling out a form.

2. Introduction

   In many applications, it is possible for a user to be presented with
   a form. The user will fill out the form, including information that
   is typed, generated by user input, or included from files that the
   user has selected. When the form is filled out, the data from the
   form is sent from the user to the receiving application.

   The definition of MultiPart/Form-Data is derived from one of those
   applications, originally set out in [RFC1867] and subsequently
   incorporated into [HTML40], where forms are expressed in HTML, and in
   which the form values are sent via HTTP or electronic mail. This
   representation is widely implemented in numerous web browsers and web
   servers.

   However, multipart/form-data can be used for forms that are presented
   using representations other than HTML (spreadsheets, Portable
   Document Format, etc), and for transport using other means than
   electronic mail or HTTP. This document defines the representation of
   form values independently of the application for which it is used.

3. Definition of multipart/form-data

   The media-type multipart/form-data follows the rules of all multipart
   MIME data streams as outlined in [RFC 2046]. In forms, there are a
   series of fields to be supplied by the user who fills out the form.
   Each field has a name. Within a given form, the names are unique.

   "multipart/form-data" contains a series of parts. Each part is
   expected to contain a content-disposition header [RFC 2183] where the
   disposition type is "form-data", and where the disposition contains
   an (additional) parameter of "name", where the value of that
   parameter is the original field name in the form. For example, a part
   might contain a header:

        Content-Disposition: form-data; name="user"

   with the value corresponding to the entry of the "user" field.

   Field names originally in non-ASCII character sets may be encoded
   within the value of the "name" parameter using the standard method
   described in RFC 2047.

   As with all multipart MIME types, each part has an optional
   "Content-Type", which defaults to text/plain. If the contents of a
   file are returned via filling out a form, then the file input is
   identified as the appropriate media type, if known, or
   "application/octet-stream". If multiple files are to be returned as
   the result of a single form entry, they should be represented as a
   "multipart/mixed" part embedded within the "multipart/form-data".

   Each part may be encoded and the "content-transfer-encoding" header
   supplied if the value of that part does not conform to the default
   encoding.

4. Use of multipart/form-data

4.1 Boundary

   As with other multipart types, a boundary is selected that does not
   occur in any of the data. Each field of the form is sent, in the
   order defined by the sending appliction and form, as a part of the
   multipart stream. Each part identifies the INPUT name within the
   original form. Each part should be labelled with an appropriate
   content-type if the media type is known (e.g., inferred from the file
   extension or operating system typing information) or as
   "application/octet-stream".

4.2 Sets of files

   If the value of a form field is a set of files rather than a single
   file, that value can be transferred together using the
   "multipart/mixed" format.

4.3 Encoding

   While the HTTP protocol can transport arbitrary binary data, the
   default for mail transport is the 7BIT encoding. The value supplied
   for a part may need to be encoded and the "content-transfer-encoding"
   header supplied if the value does not conform to the default
   encoding. [See section 5 of RFC 2046 for more details.]

4.4 Other attributes

   Forms may request file inputs from the user; the form software may
   include the file name and other file attributes, as specified in [RFC
   2184].

   The original local file name may be supplied as well, either as a
   "filename" parameter either of the "content-disposition: form-data"
   header or, in the case of multiple files, in a "content-disposition:
   file" header of the subpart. The sending application MAY supply a
   file name; if the file name of the sender's operating system is not
   in US-ASCII, the file name might be approximated, or encoded using
   the method of RFC 2231.

   This is a convenience for those cases where the files supplied by the
   form might contain references to each other, e.g., a TeX file and its
   .sty auxiliary style description.

4.5 Charset of text in form data

   Each part of a multipart/form-data is supposed to have a content-
   type. In the case where a field element is text, the charset
   parameter for the text indicates the character encoding used.

   For example, a form with a text field in which a user typed 'Joe owes
   <eu>100' where <eu> is the Euro symbol might have form data returned
   as:

     --AaB03x
     content-disposition: form-data; name="field1"
     content-type: text/plain;charset=windows-1250
     content-transfer-encoding: quoted-printable

```
   Joe owes =80100.
   --AaB03x
```

5. Operability considerations

5.1 Compression, encryption

   Some of the data in forms may be compressed or encrypted, using other
   MIME mechanisms. This is a function of the application that is
   generating the form-data.

5.2 Other data encodings rather than multipart

   Various people have suggested using new mime top-level type
   "aggregate", e.g., aggregate/mixed or a content-transfer-encoding of
   "packet" to express indeterminate-length binary data, rather than
   relying on the multipart-style boundaries. While this would be
   useful, the "multipart" mechanisms are well established, simple to
   implement on both the sending client and receiving server, and as
   efficient as other methods of dealing with multiple combinations of
   binary data.

   The multipart/form-data encoding has a high overhead and performance
   impact if there are many fields with short values. However, in
   practice, for the forms in use, for example, in HTML, the average
   overhead is not significant.

5.3 Remote files with third-party transfer

   In some scenarios, the user operating the form software might want to
   specify a URL for remote data rather than a local file. In this case,
   is there a way to allow the browser to send to the client a pointer
   to the external data rather than the entire contents? This capability
   could be implemented, for example, by having the client send to the
   server data of type "message/external-body" with "access-type" set
   to, say, "uri", and the URL of the remote data in the body of the
   message.

5.4 Non-ASCII field names

   Note that MIME headers are generally required to consist only of 7-
   bit data in the US-ASCII character set. Hence field names should be
   encoded according to the method in RFC 2047 if they contain
   characters outside of that set.

5.5 Ordered fields and duplicated field names

   The relationship of the ordering of fields within a form and the
   ordering of returned values within "multipart/form-data" is not
   defined by this specification, nor is the handling of the case where
   a form has multiple fields with the same name. While HTML-based forms
   may send back results in the order received, and intermediaries
   should not reorder the results, there are some systems which might
   not define a natural order for form fields.

5.6 Interoperability with web applications

   Many web applications use the "application/x-url-encoded" method for
   returning data from forms. This format is quite compact, e.g.:

   name=Xavier+Xantico&verdict=Yes&colour=Blue&happy=sad&Utf%F6r=Send

   however, there is no opportunity to label the enclosed data with
   content type, apply a charset, or use other encoding mechanisms.

   Many form-interpreting programs (primarily web browsers) now implement
   and generate multipart/form-data, but an existing application might
   need to optionally support both the application/x-url-encoded format
   as well.

5.7 Correlating form data with the original form

   This specification provides no specific mechanism by which
   multipart/form-data can be associated with the form that caused it to
   be transmitted. This separation is intentional; many different forms
   might be used for transmitting the same data. In practice,
   applications may supply a specific form processing resource (in HTML,
   the ACTION attribute in a FORM tag) for each different form.
   Alternatively, data about the form might be encoded in a "hidden
   field" (a field which is part of the form but which has a fixed value
   to be transmitted back to the form-data processor.)

6. Security Considerations

   The data format described in this document introduces no new security
   considerations outside of those introduced by the protocols that use
   it and of the component elements. It is important when interpreting
   content-disposition to not overwrite files in the recipients address
   space inadvertently.

   User applications that request form information from users must be
   careful not to cause a user to send information to the requestor or a
   third party unwillingly or unwittingly. For example, a form might

request 'spam' information to be sent to an unintended third party,
or private information to be sent to someone that the user might not
actually intend. While this is primarily an issue for the
representation and interpretation of forms themselves, rather than
the data representation of the result of form transmission, the
transportation of private information must be done in a way that does
not expose it to unwanted prying.

With the introduction of form-data that can reasonably send back the
content of files from user's file space, the possibility that a user
might be sent an automated script that fills out a form and then
sends the user's local file to another address arises. Thus,
additional caution is required when executing automated scripting
where form-data might include user's files.

7. Author's Address

Larry Masinter
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Fax: +1 650 812 4333
EMail: masinter@parc.xerox.com

Appendix A. Media type registration for multipart/form-data

    Media Type name:
      multipart

    Media subtype name:
      form-data

    Required parameters:
      none

    Optional parameters:
      none

    Encoding considerations:
      No additional considerations other than as for other multipart
      types.

    Security Considerations
      Applications which receive forms and process them must be careful
      not to supply data back to the requesting form processing site that
      was not intended to be sent by the recipient. This is a
      consideration for any application that generates a multipart/form-
      data.

      The multipart/form-data type introduces no new security
      considerations for recipients beyond what might occur with any of
      the enclosed parts.

References

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

    [RFC 2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions)
               Part Three: Message Header Extensions for Non-ASCII Text",
               RFC 2047, November 1996.

    [RFC 2231] Freed, N., and K. Moore, "MIME Parameter Value and Encoded
               Word Extensions: Character Sets, Languages, and
               Continuations", RFC 2231, November 1997.

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

    [RFC 1867] Nebel, E., and L. Masinter, "Form-based File Upload in
               HTML", RFC 1867, November 1995.

    [RFC 2183] Troost, R., Dorner, S., and K. Moore, "Communicating
               Presentation Information in Internet Messages: The
               Content-Disposition Header Field", RFC 2183, August 1997.

    [RFC 2184] Freed, N., and K. Moore, "MIME Parameter Value and Encoded
               Word Extensions: Character Sets, Languages, and
               Continuations", RFC 2184, August 1997.

    [HTML40]   D. Raggett, A. Le Hors, I. Jacobs. "HTML 4.0
               Specification", World Wide Web Consortium Technical Report
               "REC-html40", December, 1997. <http://www.w3.org/TR/REC-
               html40/>

Full Copyright Statement