Network Working Group
Internet-Draft
Obsoletes: 2388 (if approved)
Intended status: Standards Track
Expires: March 22, 2014

L. Masinter
Adobe

September 20, 2013

Returning Values from Forms: multipart/form-data
draft-masinter-multipart-form-data-01

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. It replaces RFC 2388.

NOTE

The Currently, XML source for this Internet Draft may be found in [1], as well as an issue tracker.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on March 22, 2014.

Table of Contents

1. 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 [HTML3.2] and [HTML4], 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.

2. Definition of multipart/form-data

The media-type multipart/form-data generally follows the model of multipart MIME data streams as described in [RFC2046] Section 5.1.

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 SHOULD be unique. After a form has been "filled out" and "submitted" (processes defined by the form), the result is a set of values for each field, the form-data.

"multipart/form-data" contains a series of parts. Each part MUST contain a Content-Disposition header [RFC2183] where the disposition type is "form-data", and where the disposition contains an (additional) parameter of "name"; the value of the 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.

2.1. Boundary

As with other multipart types, the parts are delimited with a boundary, selected such that it does not occur in any of the data. Each field of the form is sent, in the order defined by the sending application and form, as a part of the multipart stream. The boundary is supplied as a "boundary" parameter to the multipart/form-data type, e.g.,

```
multipart/form-data;boundary="-AaB03x"
```

2.2. filename attribute

For form data that represents the content of a local file, the original local file name may be supplied as as well, as a "filename" parameter of the Content-Distribution header. If the file name of the sender's operating system is not in US-ASCII, the file name MAY be approximated. 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.

2.3. Multiple files for one form field

If the value of a form field is a set of files rather than a single

file, that value MUST be transmitted by supplying each in a separate part, but all with the same "name" parameter.

2.4. Content-Type

Each part has an (optional) "Content-Type", which defaults to "text/plain". If the contents of a file are to be sent, the file data is labeled with an appropriate media type, if known, or "application/octet-stream".

2.5. The charset parameter

In the case where a field value is text the charset parameter for the Content-Type: text/plain may be used to indicate the character encoding used in that part. 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
```

2.6. The _charset_ field

Forms have the convention that the value of a form entry with entry name "_charset_" and type "hidden" is automatically set to the name of the form-charset. In this case, the value of the default charset of each text/plain part without a charset parameter is the supplied value.

2.7. Content-Transfer-Encoding

When used in transports which do not allow arbitrary binary data, each part that cannot be represented within the transport SHOULD be encoded and the "content-transfer-encoding" header supplied in that part. For example, some email transports use a 7BIT encoding. [See section 5 of [RFC2046] for more details.] (When transferred via HTTP, Content-Transfer-Encoding the form-data values SHOULD NOT be used.)

2.8. Other Content- headers

The multipart/form-data media type does not support any MIME headers in the parts other than Content-Type, Content-Disposition, and (when appropriate), Content-Transfer-Encoding.

3. Operability considerations

3.1. Non-ASCII field names and values

MIME headers in multipart/form-data are required to consist only of

7-bit data in the US-ASCII character set. While [RFC2388] suggested that non-ASCII field names should be encoded according to the method in [RFC2047] if they contain characters outside of US-ASCII, practice varies.

This specification makes three recommendations for three different states of workflow.

3.1.1. Avoid creating forms with non-ASCII field names

For broadest interoperability with existing deployed software, those creating forms SHOULD avoid non-ASCII field names. This should not be a burden, because in general the field names are not visible to users.

3.1.2. Interpreting forms and creating form-data

Some applications of this specification will supply a character encoding to be used for creation of the multipart/form-data result. In particular, [HTML5] uses

- the value of an accept-charset attribute of the <form> element, if there is one,
- the character encoding of the document containing the form, if it is US-ASCII compatible,
- otherwise UTF-8.

Call this the form-charset. For each character in the entry's name that cannot be expressed using the form-charset, replace the character by a string consisting of an ampersand character (&), a "#" character, one or more ASCII digits representing the Unicode code point of the character in base ten, and finally a semicolon character (;). The result is a US-ASCII surrogate.

multipart/form-data parts which do not have a Content-Type header and which are not the result of supplying a local file MUST be transformed by the same algorithm.

3.1.3. Parsing and interpreting form data

While this specification provides guidance for creation of multipart/form-data, interpreters of multipart/form-data should be aware of the variety of implementations. Currently, deployed browsers differ as to how they encode multipart/form-data. For this reason the matching of form elements to form-data parts may rely on a fuzzier match. In particular, some form interpreters might have followed the advice of [RFC2388] and used the [RFC2047] "encoded-word" method of encoding non-ASCII values:

```
encoded-word = "=?" charset "?" encoding "?" encoded-text "?="
```

Generally, interpreting multipart/form-data will require knowing the charset used in form encoding, in cases where the _charset_ field value or a charset parameter of a text/plain Content-Type header is not supplied.

3.2. Ordered fields and duplicated field names

Form processors given forms with a well-defined ordering SHOULD send back results in the order received and preserve duplicate field names, in order. Intermediaries MUST NOT reorder the results.(Note that there are some forms which do not define a natural order of appearance.)

3.3. 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 (primarly 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.

3.4. 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.)

4. Security Considerations

It is important when interpreting the filename of the Content-Disposition header 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.

5. Media type registration for multipart/form-data

Media Type name: multipart

Media subtype name: form-data

Required parameters: boundary

Optional parameters: none

Encoding considerations: multipart/form-data is frequently used to send arbitrary binary data. For use in transports that restrict the encoding to 7BIT or 8BIT, each part is encoded separately.

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. See Section 4 of this document.

6. References

6.1. Normative References

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

[RFC1867] Nebel, E. and L. Masinter, "Form-based File Upload in HTML", RFC 1867, November 1995.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC2183] Troost, R., Dorner, S. and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

[RFC2184] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2184, August 1997.

[RFC2231] Freed, N. and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, November 1997.

6.2. Informative References

[HTML3.2] Raggett, D., "HTML 3.2 Reference Specification", World Wide Web Consortium Recommendation REC-html32-19970114, January 1997, <http://www.w3.org/TR/REC-html32-19970114>.

[HTML4] Raggett, D., Hors, A. and I. Jacobs, "HTML 4.0 Recommendation", World Wide Web Consortium REC-html40-971218, December 1997, <http://www.w3.org/TR/REC-html40-971218>.

[HTML5] Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, E. and S. Pfeiffer, "HTML5", September 2013, <http://www.w3.org/html/wg/drafts/html/CR/>.

[RFC2388] Masinter, L., "Returning Values from Forms: multipart/form-data", RFC 2388, August 1998.

Appendix A. Changes from RFC 2388

Now, multiple files submitted as part of a single <input type=file multiple> element will result in each file having its own field; the "sets of files" feature ("multipart/mixed") in 2388 is not used.

The _charset_ convention and use of an explicit encoding charset is documented.

Explicit advice about non-ASCII field names is given, the use of RFC 2047 is deprecated (not used for encoding, but form interpreters should be aware.).

The relationship of the ordering of fields within a form and the ordering of returned values within "multipart/form-data" was not

defined before, nor was the handling of the case where a form has multiple fields with the same name.

More prescriptive about order and duplicates.

Remove obsolete discussion of alternatives.

Appendix B. Alternatives

There are numerous alternative ways in which form data can be encoded; many are listed in [RFC2388] under "Other data encodings rather than multipart". 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.

Author's Address

Larry Masinter
Adobe

Email: masinter@adobe.com
URI: http://larry.masinter.net