Abstract

By default, message header field parameters in Hypertext Transfer Protocol (HTTP) messages cannot carry characters outside the ISO-8859-1 character set. RFC 2231 defines an encoding mechanism for use in Multipurpose Internet Mail Extensions (MIME) headers. This document specifies an encoding suitable for use in HTTP header fields that is compatible with a profile of the encoding defined in RFC 2231.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5987.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

By default, message header field parameters in HTTP ([RFC2616]) messages cannot carry characters outside the ISO-8859-1 character set ([ISO-8859-1]). RFC 2231 ([RFC2231]) defines an encoding mechanism for use in MIME headers. This document specifies an encoding suitable for use in HTTP header fields that is compatible with a profile of the encoding defined in RFC 2231.

Note: in the remainder of this document, RFC 2231 is only referenced for the purpose of explaining the choice of features that were adopted; they are therefore purely informative.

Note: this encoding does not apply to message payloads transmitted over HTTP, such as when using the media type "multipart/form-data" ([RFC2388]).

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This specification uses the ABNF (Augmented Backus-Naur Form) notation defined in [RFC5234]. The following core rules are included by reference, as defined in [RFC5234], Appendix B.1: ALPHA (letters), DIGIT (decimal 0-9), HEXDIG (hexadecimal 0-9/A-F/a-f), and LWSP (linear whitespace).
Note that this specification uses the term "character set" for consistency with other IETF specifications such as RFC 2277 (see [RFC2277], Section 3). A more accurate term would be "character encoding" (a mapping of code points to octet sequences).

3. Comparison to RFC 2231 and Definition of the Encoding

RFC 2231 defines several extensions to MIME. The sections below discuss if and how they apply to HTTP header fields.

In short:

- Parameter Continuations aren’t needed (Section 3.1),
- Character Set and Language Information are useful, therefore a simple subset is specified (Section 3.2), and
- Language Specifications in Encoded Words aren’t needed (Section 3.3).

3.1. Parameter Continuations

Section 3 of [RFC2231] defines a mechanism that deals with the length limitations that apply to MIME headers. These limitations do not apply to HTTP ([RFC2616], Section 19.4.7).

Thus, parameter continuations are not part of the encoding defined by this specification.

3.2. Parameter Value Character Set and Language Information

Section 4 of [RFC2231] specifies how to embed language information into parameter values, and also how to encode non-ASCII characters, dealing with restrictions both in MIME and HTTP header parameters.

However, RFC 2231 does not specify a mandatory-to-implement character set, making it hard for senders to decide which character set to use. Thus, recipients implementing this specification MUST support the character sets "ISO-8859-1" [ISO-8859-1] and "UTF-8" [RFC3629].

Furthermore, RFC 2231 allows the character set information to be left out. The encoding defined by this specification does not allow that.

3.2.1. Definition

The syntax for parameters is defined in Section 3.6 of [RFC2616] (with RFC 2616 implied LWS translated to RFC 5234 LWS):
In order to include character set and language information, this specification modifies the RFC 2616 grammar to be:

```
 parameter     = reg-parameter / ext-parameter
 reg-parameter = parmname LWSP "=" LWSP value
 ext-parameter = parmname "*" LWSP "=" LWSP ext-value
 parmname      = 1*attr-char
 ext-value     = charset "/'" [ language ] "/'" value-chars
                 ; like RFC 2231’s <extended-initial-value>
                 ; (see [RFC2231], Section 7)
 charset       = "UTF-8" / "ISO-8859-1" / mime-charset
 mime-charset  = 1*mime-charsetc
 mime-charsetc = ALPHA / DIGIT
                / "!*" / "%!" / "%#" / "%$" / "%" / "%&"
                / "%+" / "%-" / "%=" / "%" / "%" / "%"
                / "%{" / "%}" / "%" ; as <mime-charset> in Section 2.3 of [RFC2978]
                ; except that the single quote is not included
                ; SHOULD be registered in the IANA charset registry
 language      = <Language-Tag, defined in [RFC5646], Section 2.1>
 value-chars   = *( pct-encoded / attr-char )
 pct-encoded   = "%" HEXDIG HEXDIG
                 ; see [RFC3986], Section 2.1
 attr-char     = ALPHA / DIGIT
                / "!*" / "%!" / "%#" / "%$" / "%" / "%&"
                / "%+" / "%-" / "%=" / "%" / "%" / "%"
                ; token except { "*" / "/" / "%" }
Thus, a parameter is either a regular parameter (reg-parameter), as previously defined in Section 3.6 of [RFC2616], or an extended parameter (ext-parameter).

Extended parameters are those where the left-hand side of the assignment ends with an asterisk character.

The value part of an extended parameter (ext-value) is a token that consists of three parts: the REQUIRED character set name (charset), the OPTIONAL language information (language), and a character sequence representing the actual value (value-chars), separated by single quote characters. Note that both character set names and language tags are restricted to the US-ASCII character set, and are matched case-insensitively (see [RFC2978], Section 2.3 and [RFC5646], Section 2.1.1).

Inside the value part, characters not contained in attr-char are encoded into an octet sequence using the specified character set. That octet sequence is then percent-encoded as specified in Section 2.1 of [RFC3986].

Producers MUST use either the "UTF-8" ([RFC3629]) or the "ISO-8859-1" ([ISO-8859-1]) character set. Extension character sets (mime-charset) are reserved for future use.

Note: recipients should be prepared to handle encoding errors, such as malformed or incomplete percent escape sequences, or non-decodable octet sequences, in a robust manner. This specification does not mandate any specific behavior, for instance, the following strategies are all acceptable:

* ignoring the parameter,
* stripping a non-decodable octet sequence,
* substituting a non-decodable octet sequence by a replacement character, such as the Unicode character U+FFFD (Replacement Character).

Note: the RFC 2616 token production ([RFC2616], Section 2.2) differs from the production used in RFC 2231 (imported from Section 5.1 of [RFC2045]) in that curly braces ("{" and "}") are excluded. Thus, these two characters are excluded from the attr-char production as well.
Note: the <mime-charset> ABNF defined here differs from the one in Section 2.3 of [RFC2978] in that it does not allow the single quote character (see also RFC Errata ID 1912 [Err1912]). In practice, no character set names using that character have been registered at the time of this writing.

3.2.2. Examples

Non-extended notation, using "token":

foo: bar; title=Economy

Non-extended notation, using "quoted-string":

foo: bar; title="US-$ rates"

Extended notation, using the Unicode character U+00A3 (POUND SIGN):

foo: bar; title*=iso-8859-1'en'\%A3\%20rates

Note: the Unicode pound sign character U+00A3 was encoded into the single octet A3 using the ISO-8859-1 character encoding, then percent-encoded. Also, note that the space character was encoded as %20, as it is not contained in attr-char.

Extended notation, using the Unicode characters U+00A3 (POUND SIGN) and U+20AC (EURO SIGN):

foo: bar; title*=UTF-8''\%C2\xA3\%20and\%20\%E2\%82\%AC\%20rates

Note: the Unicode pound sign character U+00A3 was encoded into the octet sequence C2 A3 using the UTF-8 character encoding, then percent-encoded. Likewise, the Unicode euro sign character U+20AC was encoded into the octet sequence E2 82 AC, then percent-encoded. Also note that HEXDIG allows both lowercase and uppercase characters, so recipients must understand both, and that the language information is optional, while the character set is not.

3.3. Language Specification in Encoded Words

Section 5 of [RFC2231] extends the encoding defined in [RFC2047] to also support language specification in encoded words. Although the HTTP/1.1 specification does refer to RFC 2047 ([RFC2616], Section 2.2), it’s not clear to which header field exactly it applies, and whether it is implemented in practice (see <http://tools.ietf.org/wg/httpbis/trac/ticket/111> for details).

Thus, this specification does not include this feature.
4. Guidelines for Usage in HTTP Header Field Definitions

Specifications of HTTP header fields that use the extensions defined in Section 3.2 ought to clearly state that. A simple way to achieve this is to normatively reference this specification, and to include the ext-value production into the ABNF for that header field.

For instance:

```
foo-header  = "foo" LWSP ":" LWSP title-param
  title-param = "title" LWSP "=" LWSP value
               / "title*" LWSP "=" LWSP ext-value
ext-value   = <see RFC 5987, Section 3.2>
```

Note: The Parameter Value Continuation feature defined in Section 3 of [RFC2231] makes it impossible to have multiple instances of extended parameters with identical parmname components, as the processing of continuations would become ambiguous. Thus, specifications using this extension are advised to disallow this case for compatibility with RFC 2231.

4.1. When to Use the Extension

Section 4.2 of [RFC2277] requires that protocol elements containing human-readable text are able to carry language information. Thus, the ext-value production ought to be always used when the parameter value is of textual nature and its language is known.

Furthermore, the extension ought to also be used whenever the parameter value needs to carry characters not present in the US-ASCII ([USASCII]) character set (note that it would be unacceptable to define a new parameter that would be restricted to a subset of the Unicode character set).

4.2. Error Handling

Header field specifications need to define whether multiple instances of parameters with identical parmname components are allowed, and how they should be processed. This specification suggests that a parameter using the extended syntax takes precedence. This would allow producers to use both formats without breaking recipients that do not understand the extended syntax yet.

Example:

```
foo: bar; title="EURO exchange rates";
title*=utf-8''%e2%82%ac%20exchange%20rates
```
In this case, the sender provides an ASCII version of the title for legacy recipients, but also includes an internationalized version for recipients understanding this specification -- the latter obviously ought to prefer the new syntax over the old one.

Note: at the time of this writing, many implementations failed to ignore the form they do not understand, or prioritize the ASCII form although the extended syntax was present.

5. Security Considerations

The format described in this document makes it possible to transport non-ASCII characters, and thus enables character "spoofing" scenarios, in which a displayed value appears to be something other than it is.

Furthermore, there are known attack scenarios relating to decoding UTF-8.

See Section 10 of [RFC3629] for more information on both topics.

In addition, the extension specified in this document makes it possible to transport multiple language variants for a single parameter, and such use might allow spoofing attacks, where different language versions of the same parameter are not equivalent. Whether this attack is useful as an attack depends on the parameter specified.

6. Acknowledgements

Thanks to Martin Duerst and Frank Ellermann for help figuring out ABNF details, to Graham Klyne and Alexey Melnikov for general review, to Chris Newman for pointing out an RFC 2231 incompatibility, and to Benjamin Carlyle and Roar Lauritzsen for implementer’s feedback.

7. References

7.1. Normative References


Informative References


Author’s Address

Julian F. Reschke
greenbytes GmbH
Hafenweg 16
Muenster, NW  48155
Germany

EMail: julian.reschke@greenbytes.de
URI:  http://greenbytes.de/tech/webdav/