HTTPbis Working Group
Internet-Draft
Obsoletes: 2616 (if approved)
Intended status: Standards Track
Expires: January 17, 2013

R. Fielding, Ed.
Adobe
Y. Lafon, Ed.
W3C
J. Reschke, Ed.
greenbytes

July 16, 2012

HTTP/1.1, part 5: Range Requests
draft-ietf-httpbis-p5-range-20

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypertext information systems. This document defines range requests and the rules for constructing and combining responses to those requests.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <http://lists.w3.org/Archives/Public/ietf-http-wg/>.

The current issues list is at <http://tools.ietf.org/wg/httpbis/trac/report/3> and related documents (including fancy diffs) can be found at <http://tools.ietf.org/wg/httpbis/>.

The changes in this draft are summarized in Appendix E.1.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 4
   1.1. Conformance and Error Handling 4
   1.2. Syntax Notation 5
2. Range Units 5
   2.1. Range Specifier Registry 6
3. Status Code Definitions 6
   3.1. 206 Partial Content 6
   3.2. 416 Requested Range Not Satisfiable 7
4. Responses to a Range Request 7
   4.1. Response to a Single and Multiple Ranges Request 7
   4.2. Combining Ranges 8
5. Header Field Definitions 9
   5.1. Accept-Ranges 9
   5.2. Content-Range 10
   5.3. If-Range 11
   5.4. Range 12
      5.4.1. Byte Ranges 12
      5.4.2. Range Retrieval Requests 14
6. IANA Considerations 15
   6.1. Status Code Registration 15
   6.2. Header Field Registration 15
   6.3. Range Specifier Registration 16
7. Security Considerations 16
   7.1. Overlapping Ranges 16
8. Acknowledgments 16
9. References 16
   9.1. Normative References 16
   9.2. Informative References 17
Appendix A. Internet Media Type multipart/byteranges 18
Appendix B. Changes from RFC 2616 20
Appendix C. Imported ABNF 20
Appendix D. Collected ABNF 21
Appendix E. Change Log (to be removed by RFC Editor before publication) 22
   E.1. Since draft-ietf-httpbis-p5-range-19 22
Index 22

1. Introduction

HTTP clients often encounter interrupted data transfers as a result of canceled requests or dropped connections. When a client has stored a partial representation, it is desirable to request the remainder of that representation in a subsequent request rather than transfer the entire representation. There are also a number of Web applications that benefit from being able to request only a subset of a larger representation, such as a single page of a very large document or only part of an image to be rendered by a device with limited local storage.

This document defines HTTP/1.1 range requests, partial responses, and the multipart/byteranges media type. The protocol for range requests is an OPTIONAL feature of HTTP, designed so resources or recipients that do not implement this feature can respond as if it is a normal GET request without impacting interoperability. Partial responses are indicated by a distinct status code to not be mistaken for full responses by intermediate caches that might not implement the feature.

Although the HTTP range request mechanism is designed to allow for extensible range types, this specification only defines requests for byte ranges.

1.1. Conformance and Error Handling

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 targets conformance criteria according to the role of a participant in HTTP communication. Hence, HTTP requirements are placed on senders, recipients, clients, servers, user agents, intermediaries, origin servers, proxies, gateways, or caches, depending on what behavior is being constrained by the requirement. See Section 2 of [Part1] for definitions of these terms.

The verb "generate" is used instead of "send" where a requirement differentiates between creating a protocol element and merely forwarding a received element downstream.

An implementation is considered conformant if it complies with all of the requirements associated with the roles it partakes in HTTP. Note that SHOULD-level requirements are relevant here, unless one of the documented exceptions is applicable.

This document also uses ABNF to define valid protocol elements

(Section 1.2). In addition to the prose requirements placed upon them, senders MUST NOT generate protocol elements that do not match the grammar defined by the ABNF rules for those protocol elements that are applicable to the sender's role. If a received protocol element is processed, the recipient MUST be able to parse any value that would match the ABNF rules for that protocol element, excluding only those rules not applicable to the recipient's role.

Unless noted otherwise, a recipient MAY attempt to recover a usable protocol element from an invalid construct. HTTP does not define specific error handling mechanisms except when they have a direct impact on security, since different applications of the protocol require different error handling strategies. For example, a Web browser might wish to transparently recover from a response where the Location header field doesn't parse according to the ABNF, whereas a systems control client might consider any form of error recovery to be dangerous.

1.2. Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with the list rule extension defined in Section 1.2 of [Part1]. Appendix C describes rules imported from other documents. Appendix D shows the collected ABNF with the list rule expanded.

2. Range Units

HTTP/1.1 allows a client to request that only part (a range) of the representation be included within the response. HTTP/1.1 uses range units in the Range (Section 5.4) and Content-Range (Section 5.2) header fields. A representation can be broken down into subranges according to various structural units.

```
  range-unit       = bytes-unit / other-range-unit
  bytes-unit       = "bytes"
  other-range-unit = token
```

HTTP/1.1 has been designed to allow implementations of applications that do not depend on knowledge of ranges. The only range unit defined by HTTP/1.1 is "bytes". Additional specifiers can be defined as described in Section 2.1.

If a range unit is not understood in a request, a server MUST ignore the whole Range header field (Section 5.4). If a range unit is not understood in a response, an intermediary SHOULD pass the response to the client; a client MUST fail.

2.1. Range Specifier Registry

The HTTP Range Specifier Registry defines the name space for the range specifier names.

Registrations MUST include the following fields:

- o Name
- o Description
- o Pointer to specification text

Values to be added to this name space require IETF Review (see [RFC5226], Section 4.1).

The registry itself is maintained at <http://www.iana.org/assignments/http-range-specifiers>.

3. Status Code Definitions

3.1. 206 Partial Content

The server has fulfilled the partial GET request for the resource. The request MUST have included a Range header field (Section 5.4) indicating the desired range, and MAY have included an If-Range header field (Section 5.3) to make the request conditional.

The response MUST include the following header fields:

- o Either a Content-Range header field (Section 5.2) indicating the range included with this response, or a multipart/byteranges Content-Type including Content-Range fields for each part. If a Content-Length header field is present in the response, its value MUST match the actual number of octets transmitted in the message body.
- o Date
- o Cache-Control, ETag, Expires, Content-Location and/or Vary, if the header field would have been sent in a 200 (OK) response to the same request

If a 206 is sent in response to a request with an If-Range header field, it SHOULD NOT include other representation header fields. Otherwise, the response MUST include all of the representation header fields that would have been returned with a 200 (OK) response to the same request.

Caches MAY use a heuristic (see Section 4.1.2 of [Part6]) to determine freshness for 206 responses.

3.2. 416 Requested Range Not Satisfiable

A server SHOULD return a response with this status code if a request included a Range header field (Section 5.4), and none of the ranges-specifier values in this field overlap the current extent of the selected resource, and the request did not include an If-Range header field (Section 5.3). (For byte-ranges, this means that the first-byte-pos of all of the byte-range-spec values were greater than the current length of the selected resource.)

When this status code is returned for a byte-range request, the response SHOULD include a Content-Range header field specifying the current length of the representation (see Section 5.2). This response MUST NOT use the multipart/byteranges content-type. For example,

```
HTTP/1.1 416 Requested Range Not Satisfiable
Date: Mon, 20 Jan 2012 15:41:54 GMT
Content-Range: bytes */47022
Content-Type: image/gif
```

Note: Clients cannot depend on servers to send a 416 (Requested Range Not Satisfiable) response instead of a 200 (OK) response for an unsatisfiable Range header field, since not all servers implement this header field.

4. Responses to a Range Request

4.1. Response to a Single and Multiple Ranges Request

When an HTTP message includes the content of a single range (for example, a response to a request for a single range, or to a request for a set of ranges that overlap without any holes), this content is transmitted with a Content-Range header field, and a Content-Length header field showing the number of bytes actually transferred. For example,

```
HTTP/1.1 206 Partial Content
Date: Wed, 15 Nov 1995 06:25:24 GMT
Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT
Content-Range: bytes 21010-47021/47022
Content-Length: 26012
Content-Type: image/gif
```

When an HTTP message includes the content of multiple ranges (for

example, a response to a request for multiple non-overlapping ranges), these are transmitted as a multipart message. The multipart media type used for this purpose is "multipart/byteranges" as defined in Appendix A.

A server MAY combine requested ranges when those ranges are overlapping (see Section 7.1).

A response to a request for a single range MUST NOT be sent using the multipart/byteranges media type. A response to a request for multiple ranges, whose result is a single range, MAY be sent as a multipart/byteranges media type with one part. A client that cannot decode a multipart/byteranges message MUST NOT ask for multiple ranges in a single request.

When a client asks for multiple ranges in one request, the server SHOULD return them in the order that they appeared in the request.

4.2. Combining Ranges

A response might transfer only a subrange of a representation if the connection closed prematurely or if the request used one or more Range specifications. After several such transfers, a client might have received several ranges of the same representation. These ranges can only be safely combined if they all have in common the same strong validator, where "strong validator" is defined to be either an entity-tag that is not marked as weak (Section 2.3 of [Part4]) or, if no entity-tag is provided, a Last-Modified value that is strong in the sense defined by Section 2.2.2 of [Part4].

When a client receives an incomplete 200 (OK) or 206 (Partial Content) response and already has one or more stored responses for the same method and effective request URI, all of the stored responses with the same strong validator MAY be combined with the partial content in this new response. If none of the stored responses contain the same strong validator, then this new response corresponds to a new representation and MUST NOT be combined with the existing stored responses.

If the new response is an incomplete 200 (OK) response, then the header fields of that new response are used for any combined response and replace those of the matching stored responses.

If the new response is a 206 (Partial Content) response and at least one of the matching stored responses is a 200 (OK), then the combined response header fields consist of the most recent 200 response's header fields. If all of the matching stored responses are 206 responses, then the stored response with the most header fields is

used as the source of header fields for the combined response, except that the client MUST use other header fields provided in the new response, aside from Content-Range, to replace all instances of the corresponding header fields in the stored response.

The combined response message body consists of the union of partial content ranges in the new response and each of the selected responses. If the union consists of the entire range of the representation, then the combined response MUST be recorded as a complete 200 (OK) response with a Content-Length header field that reflects the complete length. Otherwise, the combined response(s) MUST include a Content-Range header field describing the included range(s) and be recorded as incomplete. If the union consists of a discontinuous range of the representation, then the client MAY store it as either a multipart range response or as multiple 206 responses with one continuous range each.

5. Header Field Definitions

This section defines the syntax and semantics of HTTP/1.1 header fields related to range requests and partial responses.

5.1. Accept-Ranges

The "Accept-Ranges" header field allows a resource to indicate its acceptance of range requests.

```
  Accept-Ranges     = acceptable-ranges
  acceptable-ranges = 1#range-unit / "none"
```

Origin servers that accept byte-range requests MAY send

```
  Accept-Ranges: bytes
```

but are not required to do so. Clients MAY generate range requests without having received this header field for the resource involved. Range units are defined in Section 2.

Servers that do not accept any kind of range request for a resource MAY send

```
  Accept-Ranges: none
```

to advise the client not to attempt a range request.

5.2. Content-Range

The "Content-Range" header field is sent with a partial representation to specify where in the full representation the payload body is intended to be applied.

Range units are defined in Section 2.

```
  Content-Range           = byte-content-range-spec
                          / other-content-range-spec

  byte-content-range-spec = bytes-unit SP
                            byte-range-resp-spec "/"
                            ( instance-length / "*" )

  byte-range-resp-spec    = (first-byte-pos "-" last-byte-pos)
                          / "*"

  instance-length         = 1*DIGIT

  other-content-range-spec = other-range-unit SP
                             other-range-resp-spec
  other-range-resp-spec    = *CHAR
```

The header field SHOULD indicate the total length of the full representation, unless this length is unknown or difficult to determine. The asterisk "*" character means that the instance-length is unknown at the time when the response was generated.

Unlike byte-ranges-specifier values (see Section 5.4.1), a byte-range-resp-spec MUST only specify one range, and MUST contain absolute byte positions for both the first and last byte of the range.

A byte-content-range-spec with a byte-range-resp-spec whose last-byte-pos value is less than its first-byte-pos value, or whose instance-length value is less than or equal to its last-byte-pos value, is invalid. The recipient of an invalid byte-content-range-spec MUST ignore it and any content transferred along with it.

In the case of a byte range request: A server sending a response with status code 416 (Requested Range Not Satisfiable) SHOULD include a Content-Range field with a byte-range-resp-spec of "*". The instance-length specifies the current length of the selected resource. A response with status code 206 (Partial Content) MUST NOT include a Content-Range field with a byte-range-resp-spec of "*".

The "Content-Range" header field has no meaning for status codes that

do not explicitly describe its semantic. Currently, only status codes 206 (Partial Content) and 416 (Requested Range Not Satisfiable) describe the meaning of this header field.

Examples of byte-content-range-spec values, assuming that the representation contains a total of 1234 bytes:

- The first 500 bytes:

```
bytes 0-499/1234
```

- The second 500 bytes:

```
bytes 500-999/1234
```

- All except for the first 500 bytes:

```
bytes 500-1233/1234
```

- The last 500 bytes:

```
bytes 734-1233/1234
```

If the server ignores a byte-range-spec (for example if it is syntactically invalid, or if it might be seen as a denial-of-service attack), the server SHOULD treat the request as if the invalid Range header field did not exist. (Normally, this means return a 200 (OK) response containing the full representation).

5.3. If-Range

If a client has a partial copy of a representation and wishes to have an up-to-date copy of the entire representation, it could use the Range header field with a conditional GET (using either or both of If-Unmodified-Since and If-Match.) However, if the condition fails because the representation has been modified, the client would then have to make a second request to obtain the entire current representation.

The "If-Range" header field allows a client to "short-circuit" the second request. Informally, its meaning is "if the representation is unchanged, send me the part(s) that I am missing; otherwise, send me the entire new representation".

```
  If-Range = entity-tag / HTTP-date
```

Clients MUST NOT use an entity-tag marked as weak in an If-Range field value and MUST NOT use a Last-Modified date in an If-Range

field value unless it has no entity-tag for the representation and the Last-Modified date it does have for the representation is strong in the sense defined by Section 2.2.2 of [Part4].

A server that evaluates a conditional range request that is applicable to one of its representations MUST evaluate the condition as false if the entity-tag used as a validator is marked as weak or, when an HTTP-date is used as the validator, if the date value is not strong in the sense defined by Section 2.2.2 of [Part4]. (A server can distinguish between a valid HTTP-date and any form of entity-tag by examining the first two characters.)

The If-Range header field SHOULD only be sent by clients together with a Range header field. The If-Range header field MUST be ignored if it is received in a request that does not include a Range header field. The If-Range header field MUST be ignored by a server that does not support the sub-range operation.

If the validator given in the If-Range header field matches the current validator for the selected representation of the target resource, then the server SHOULD send the specified sub-range of the representation using a 206 (Partial Content) response. If the validator does not match, then the server SHOULD send the entire representation using a 200 (OK) response.

5.4. Range

5.4.1. Byte Ranges

Since all HTTP representations are transferred as sequences of bytes, the concept of a byte range is meaningful for any HTTP representation. (However, not all clients and servers need to support byte-range operations.)

Byte range specifications in HTTP apply to the sequence of bytes in the representation body (not necessarily the same as the message body).

A byte range operation MAY specify a single range of bytes, or a set of ranges within a single representation.

```
  byte-ranges-specifier = bytes-unit "=" byte-range-set
  byte-range-set  = 1#( byte-range-spec / suffix-byte-range-spec )
  byte-range-spec = first-byte-pos "-" [ last-byte-pos ]
  first-byte-pos  = 1*DIGIT
  last-byte-pos   = 1*DIGIT
```

The first-byte-pos value in a byte-range-spec gives the byte-offset

of the first byte in a range. The last-byte-pos value gives the byte-offset of the last byte in the range; that is, the byte positions specified are inclusive. Byte offsets start at zero.

If the last-byte-pos value is present, it MUST be greater than or equal to the first-byte-pos in that byte-range-spec, or the byte-range-spec is syntactically invalid. The recipient of a byte-range-set that includes one or more syntactically invalid byte-range-spec values MUST ignore the header field that includes that byte-range-set.

If the last-byte-pos value is absent, or if the value is greater than or equal to the current length of the representation body, last-byte-pos is taken to be equal to one less than the current length of the representation in bytes.

By its choice of last-byte-pos, a client can limit the number of bytes retrieved without knowing the size of the representation.

```
  suffix-byte-range-spec = "-" suffix-length
  suffix-length = 1*DIGIT
```

A suffix-byte-range-spec is used to specify the suffix of the representation body, of a length given by the suffix-length value. (That is, this form specifies the last N bytes of a representation.) If the representation is shorter than the specified suffix-length, the entire representation is used.

If a syntactically valid byte-range-set includes at least one byte-range-spec whose first-byte-pos is less than the current length of the representation, or at least one suffix-byte-range-spec with a non-zero suffix-length, then the byte-range-set is satisfiable. Otherwise, the byte-range-set is unsatisfiable. If the byte-range-set is unsatisfiable, the server SHOULD return a response with a 416 (Requested Range Not Satisfiable) status code. Otherwise, the server SHOULD return a response with a 206 (Partial Content) status code containing the satisfiable ranges of the representation.

In the byte range syntax, first-byte-pos, last-byte-pos, and suffix-length are expressed as decimal number of octets. Since there is no predefined limit to the length of an HTTP payload, recipients SHOULD anticipate potentially large decimal numerals and prevent parsing errors due to integer conversion overflows.

Examples of byte-ranges-specifier values (assuming a representation of length 10000):

- The first 500 bytes (byte offsets 0-499, inclusive):

```
bytes=0-499
```

- The second 500 bytes (byte offsets 500-999, inclusive):

```
bytes=500-999
```

- The final 500 bytes (byte offsets 9500-9999, inclusive):

```
bytes=-500
```

  Or:

```
bytes=9500-
```

- The first and last bytes only (bytes 0 and 9999):

```
bytes=0-0,-1
```

- Several legal but not canonical specifications of the second 500 bytes (byte offsets 500-999, inclusive):

```
bytes=500-600,601-999
bytes=500-700,601-999
```

5.4.2. Range Retrieval Requests

The "Range" header field defines the GET method (conditional or not) to request one or more sub-ranges of the response representation body, instead of the entire representation body.

```
Range = byte-ranges-specifier / other-ranges-specifier
other-ranges-specifier = other-range-unit "=" other-range-set
other-range-set = 1*CHAR
```

A server MAY ignore the Range header field. However, origin servers and intermediate caches ought to support byte ranges when possible, since Range supports efficient recovery from partially failed transfers, and supports efficient partial retrieval of large representations.

If the server supports the Range header field and the specified range or ranges are appropriate for the representation:

- The presence of a Range header field in an unconditional GET modifies what is returned if the GET is otherwise successful. In other words, the response carries a status code of 206 (Partial

Content) instead of 200 (OK).

- o The presence of a Range header field in a conditional GET (a request using one or both of If-Modified-Since and If-None-Match, or one or both of If-Unmodified-Since and If-Match) modifies what is returned if the GET is otherwise successful and the condition is true. It does not affect the 304 (Not Modified) response returned if the conditional is false.

In some cases, it might be more appropriate to use the If-Range header field (see Section 5.3) in addition to the Range header field.

If a proxy that supports ranges receives a Range request, forwards the request to an inbound server, and receives an entire representation in reply, it MAY only return the requested range to its client.

6. IANA Considerations

6.1. Status Code Registration

The HTTP Status Code Registry located at <http://www.iana.org/assignments/http-status-codes> shall be updated with the registrations below:

Value	Description	Reference
206	Partial Content	Section 3.1
416	Requested Range Not Satisfiable	Section 3.2

6.2. Header Field Registration

The Message Header Field Registry located at <http://www.iana.org/assignments/message-headers/message-header-index.html> shall be updated with the permanent registrations below (see [RFC3864]):

Header Field Name	Protocol	Status	Reference
Accept-Ranges	http	standard	Section 5.1
Content-Range	http	standard	Section 5.2
If-Range	http	standard	Section 5.3
Range	http	standard	Section 5.4

The change controller is: "IETF (iesg@ietf.org) - Internet

Engineering Task Force".

6.3. Range Specifier Registration

The registration procedure for HTTP Range Specifiers is defined by Section 2.1 of this document.

The HTTP Range Specifier Registry shall be created at <http://www.iana.org/assignments/http-range-specifiers> and be populated with the registrations below:

Range Specifier Name	Description	Reference
bytes	a range of octets	Section 2
none	reserved as keyword, indicating no ranges are supported	Section 5.1

The change controller is: "IETF (iesg@ietf.org) - Internet Engineering Task Force".

7. Security Considerations

This section is meant to inform application developers, information providers, and users of the security limitations in HTTP/1.1 as described by this document. The discussion does not include definitive solutions to the problems revealed, though it does make some suggestions for reducing security risks.

7.1. Overlapping Ranges

Range requests containing overlapping ranges can lead to the situation where a server is sending far more data than the size of the complete resource representation.

8. Acknowledgments

See Section 9 of [Part1].

9. References

9.1. Normative References

[Part1] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "HTTP/1.1, part 1: Message Routing and Syntax"",

draft-ietf-httpbis-p1-messaging-20 (work in progress), July 2012.

[Part2] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "HTTP/1.1, part 2: Semantics and Payloads", draft-ietf-httpbis-p2-semantics-20 (work in progress), July 2012.

[Part4] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "HTTP/1.1, part 4: Conditional Requests", draft-ietf-httpbis-p4-conditional-20 (work in progress), July 2012.

[Part6] Fielding, R., Ed., Lafon, Y., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1, part 6: Caching", draft-ietf-httpbis-p6-cache-20 (work in progress), July 2012.

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

9.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Appendix A. Internet Media Type multipart/byteranges

When an HTTP 206 (Partial Content) response message includes the content of multiple ranges (a response to a request for multiple non-overlapping ranges), these are transmitted as a multipart message body ([RFC2046], Section 5.1). The media type for this purpose is called "multipart/byteranges". The following is to be registered with IANA [RFC4288].

The multipart/byteranges media type includes one or more parts, each with its own Content-Type and Content-Range fields. The required boundary parameter specifies the boundary string used to separate each body-part.

Type name: multipart

Subtype name: byteranges

Required parameters: boundary

Optional parameters: none

Encoding considerations: only "7bit", "8bit", or "binary" are permitted

Security considerations: none

Interoperability considerations: none

Published specification: This specification (see Appendix A).

Applications that use this media type: HTTP components supporting multiple ranges in a single request.

Additional information:

Magic number(s): none

File extension(s): none

Macintosh file type code(s): none

Person and email address to contact for further information: See Authors Section.

Intended usage: COMMON

Restrictions on usage: none

Author/Change controller: IESG

Note: Despite the name "multipart/byteranges" is not limited to the byte ranges only.

For example:

```
  HTTP/1.1 206 Partial Content
  Date: Wed, 15 Nov 1995 06:25:24 GMT
  Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT
  Content-type: multipart/byteranges; boundary=THIS_STRING_SEPARATES

  --THIS_STRING_SEPARATES
  Content-type: application/pdf
  Content-range: bytes 500-999/8000

  ...the first range...
  --THIS_STRING_SEPARATES
  Content-type: application/pdf
  Content-range: bytes 7000-7999/8000

  ...the second range
  --THIS_STRING_SEPARATES--
```

Another example, using the "exampleunit" range unit:

```
  HTTP/1.1 206 Partial Content
  Date: Tue, 14 Nov 1995 06:25:24 GMT
  Last-Modified: Tue, 14 July 04:58:08 GMT
  Content-type: multipart/byteranges; boundary=THIS_STRING_SEPARATES

  --THIS_STRING_SEPARATES
  Content-type: video/example
  Content-range: exampleunit 1.2-4.3/25

  ...the first range...
  --THIS_STRING_SEPARATES
  Content-type: video/example
  Content-range: exampleunit 11.2-14.3/25

  ...the second range
  --THIS_STRING_SEPARATES--
```

Notes:

1. Additional CRLFs MAY precede the first boundary string in the body.

2. Although [RFC2046] permits the boundary string to be quoted, some existing implementations handle a quoted boundary string incorrectly.

3. A number of clients and servers were coded to an early draft of the byteranges specification to use a media type of multipart/x-byteranges, which is almost, but not quite compatible with the version documented in HTTP/1.1.

Appendix B. Changes from RFC 2616

Introduce Range Specifier Registry. (Section 2.1)

Clarify that it is not ok to use a weak validator in a 206 response. (Section 3.1)

Change ABNF productions for header fields to only define the field value. (Section 5)

Clarify that multipart/byteranges can consist of a single part. (Appendix A)

Appendix C. Imported ABNF

The following core rules are included by reference, as defined in Appendix B.1 of [RFC5234]: ALPHA (letters), CR (carriage return), CRLF (CR LF), CTL (controls), DIGIT (decimal 0-9), DQUOTE (double quote), HEXDIG (hexadecimal 0-9/A-F/a-f), LF (line feed), OCTET (any 8-bit sequence of data), SP (space), and VCHAR (any visible US-ASCII character).

Note that all rules derived from token are to be compared case-insensitively, like range-unit and acceptable-ranges.

The rules below are defined in [Part1]:

```
OWS        = <OWS, defined in [Part1], Section 3.2.1>
token      = <token, defined in [Part1], Section 3.2.4>
```

The rules below are defined in other parts:

```
HTTP-date  = <HTTP-date, defined in [Part2], Section 5.1>
entity-tag = <entity-tag, defined in [Part4], Section 2.3>
```

Appendix D. Collected ABNF

```
   Accept-Ranges = acceptable-ranges

   Content-Range = byte-content-range-spec / other-content-range-spec

   HTTP-date = <HTTP-date, defined in [Part2], Section 5.1>

   If-Range = entity-tag / HTTP-date

   OWS = <OWS, defined in [Part1], Section 3.2.1>

   Range = byte-ranges-specifier / other-ranges-specifier

   acceptable-ranges = ( *( "," OWS ) range-unit *( OWS "," [ OWS
    range-unit ] ) ) / "none"

   byte-content-range-spec = bytes-unit SP byte-range-resp-spec "/" (
    instance-length / "*" )
   byte-range-resp-spec = ( first-byte-pos "-" last-byte-pos ) / "*"
   byte-range-set = *( "," OWS ) ( byte-range-spec /
    suffix-byte-range-spec ) *( OWS "," [ OWS ( byte-range-spec /
    suffix-byte-range-spec ) ] )
   byte-range-spec = first-byte-pos "-" [ last-byte-pos ]
   byte-ranges-specifier = bytes-unit "=" byte-range-set
   bytes-unit = "bytes"

   entity-tag = <entity-tag, defined in [Part4], Section 2.3>

   first-byte-pos = 1*DIGIT

   instance-length = 1*DIGIT

   last-byte-pos = 1*DIGIT

   other-content-range-spec = other-range-unit SP other-range-resp-spec
   other-range-resp-spec = *CHAR
   other-range-set = 1*CHAR
   other-range-unit = token
   other-ranges-specifier = other-range-unit "=" other-range-set

   range-unit = bytes-unit / other-range-unit

   suffix-byte-range-spec = "-" suffix-length
   suffix-length = 1*DIGIT

   token = <token, defined in [Part1], Section 3.2.4>
```

Appendix E. Change Log (to be removed by RFC Editor before publication)

Changes up to the first Working Group Last Call draft are summarized in <http://tools.ietf.org/html/draft-ietf-httpbis-p5-range-19#appendix-D>.

E.1. Since draft-ietf-httpbis-p5-range-19

Closed issues:

- <http://tools.ietf.org/wg/httpbis/trac/ticket/358>: "ABNF list expansion code problem"
- <http://tools.ietf.org/wg/httpbis/trac/ticket/361>: "ABNF requirements for recipients"
- <http://tools.ietf.org/wg/httpbis/trac/ticket/367>: "reserve 'none' as byte range unit"
- <http://tools.ietf.org/wg/httpbis/trac/ticket/368>: "note introduction of new IANA registries as normative changes"
- <http://tools.ietf.org/wg/httpbis/trac/ticket/369>: "range units vs leading zeroes vs size"

Index

2
   206 Partial Content (status code) 6

4
   416 Requested Range Not Satisfiable (status code) 7

A
   Accept-Ranges header field 9

C
   Content-Range header field 10

G
   Grammar
      Accept-Ranges 9
      acceptable-ranges 9
      byte-content-range-spec 10
      byte-range-resp-spec 10
      byte-range-set 12
      byte-range-spec 12
      byte-ranges-specifier 12

bytes-unit 5
Content-Range 10
first-byte-pos 12
If-Range 11
instance-length 10
last-byte-pos 12
other-range-unit 5
Range 14
range-unit 5
ranges-specifier 12
suffix-byte-range-spec 13
suffix-length 13

H
Header Fields
Accept-Ranges 9
Content-Range 10
If-Range 11
Range 12

I
If-Range header field 11

M
Media Type
multipart/byteranges 18
multipart/x-byteranges 20
multipart/byteranges Media Type 18
multipart/x-byteranges Media Type 20

R
Range header field 12

S
Status Codes
206 Partial Content 6
416 Requested Range Not Satisfiable 7

Authors' Addresses

Roy T. Fielding (editor)
Adobe Systems Incorporated
345 Park Ave
San Jose, CA 95110
USA

EMail: fielding@gbiv.com
URI: http://roy.gbiv.com/

Yves Lafon (editor)
World Wide Web Consortium
W3C / ERCIM
2004, rte des Lucioles
Sophia-Antipolis, AM 06902
France

EMail: ylafon@w3.org
URI: http://www.raubacapeu.net/people/yves/

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

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