Internet Engineering Task Force (IETF)
Request for Comments: 7233
Obsoletes: 2616
Category: Standards Track
ISSN: 2070-1721

R. Fielding, Editor
Adobe
Y. Lafon, Editor
W3C
J. Reschke, Editor
greenbytes
June 2014

Hypertext Transfer Protocol (HTTP/1.1): Range Requests

Abstract

The Hypertext Transfer Protocol (HTTP) is a stateless 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.

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/rfc7233[1].

Copyright Notice

[1] http://www.rfc-editor.org/info/rfc7233
[2] http://trustee.ietf.org/license-info

Table of Contents

1 Introduction....4
1.1 Conformance and Error Handling....4
1.2 Syntax Notation....4
2 Range Units....5
2.1 Byte Ranges....5
2.2 Other Range Units....6
2.3 Accept-Ranges....6
3 Range Requests....7
3.1 Range....7
3.2 If-Range....7
4 Responses to a Range Request....9
4.1 206 Partial Content....9
4.2 Content-Range....10
4.3 Combining Ranges....11
4.4 416 Range Not Satisfiable....12
5 IANA Considerations....13
5.1 Range Unit Registry....13
5.1.1 Procedure....13
5.1.2 Registrations....13
5.2 Status Code Registration....13
5.3 Header Field Registration....13
5.4 Internet Media Type Registration....13
5.4.1 Internet Media Type multipart/byteranges....14
6 Security Considerations....15
6.1 Denial-of-Service Attacks Using Range....15
7 Acknowledgments....16
8 References....17
8.1 Normative References....17
8.2 Informative References....17
A Internet Media Type multipart/byteranges....18
B Changes from RFC 2616....19
C Imported ABNF....20
D Collected ABNF....21
Index....22

Authors' Addresses **23**

1. Introduction

Hypertext Transfer Protocol (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. Likewise, devices with limited local storage might benefit from being able to request only a subset of a larger representation, such as a single page of a very large document, or the dimensions of an embedded image.

This document defines HTTP/1.1 range requests, partial responses, and the multipart/byteranges media type. Range requests are an OPTIONAL feature of HTTP, designed so that recipients not implementing this feature (or not supporting it for the target resource) 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 caches that might not implement the feature.

Although the 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].

Conformance criteria and considerations regarding error handling are defined in Section 2.5 of [RFC7230].

1.2. Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] with a list extension, defined in Section 7 of [RFC7230], that allows for compact definition of comma-separated lists using a '#' operator (similar to how the '*' operator indicates repetition). Appendix C describes rules imported from other documents. Appendix D shows the collected grammar with all list operators expanded to standard ABNF notation.

2. Range Units

A representation can be partitioned into subranges according to various structural units, depending on the structure inherent in the representation's media type. This "*range unit*" is used in the Accept-Ranges (Section 2.3) response header field to advertise support for range requests, the Range (Section 3.1) request header field to delineate the parts of a representation that are requested, and the Content-Range (Section 4.2) payload header field to describe which part of a representation is being transferred.

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

2.1. Byte Ranges

Since representation data is transferred in payloads as a sequence of octets, a byte range is a meaningful substructure for any representation transferable over HTTP (Section 3 of [RFC7231]). The "bytes" range unit is defined for expressing subranges of the data's octet sequence.

```
bytes-unit       = "bytes"
```

A byte-range request can 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.

Examples of byte-ranges-specifier values:

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

```
bytes=0-499
```

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

```
bytes=500-999
```

A byte-range-spec is invalid if the last-byte-pos value is present and less than the first-byte-pos.

A client can limit the number of bytes requested without knowing the size of the selected representation. If the last-byte-pos value is absent, or if the value is greater than or equal to the current length of the representation data, the byte range is interpreted as the remainder of the representation (i.e., the server replaces the value of last-byte-pos with a value that is one less than the current length of the selected representation).

A client can request the last N bytes of the selected representation using a suffix-byte-range-spec.

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

If the selected representation is shorter than the specified suffix-length, the entire representation is used.

Additional examples, assuming a representation of length 10000:

- 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
```

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

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

If a valid byte-range-set includes at least one byte-range-spec with a first-byte-pos that 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.

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 a payload, recipients MUST anticipate potentially large decimal numerals and prevent parsing errors due to integer conversion overflows.

2.2. Other Range Units

Range units are intended to be extensible. New range units ought to be registered with IANA, as defined in Section 5.1.

```
other-range-unit = token
```

2.3. Accept-Ranges

The "Accept-Ranges" header field allows a server to indicate that it supports range requests for the target resource.

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

An origin server that supports byte-range requests for a given target resource MAY send

```
Accept-Ranges: bytes
```

to indicate what range units are supported. A client MAY generate range requests without having received this header field for the resource involved. Range units are defined in Section 2.

A server that does not support any kind of range request for the target resource MAY send

```
Accept-Ranges: none
```

to advise the client not to attempt a range request.

3. Range Requests

3.1. Range

The "Range" header field on a GET request modifies the method semantics to request transfer of only one or more subranges of the selected representation data, rather than the entire selected representation data.

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

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 partial retrieval of large representations. A server MUST ignore a Range header field received with a request method other than GET.

An origin server MUST ignore a Range header field that contains a range unit it does not understand. A proxy MAY discard a Range header field that contains a range unit it does not understand.

A server that supports range requests MAY ignore or reject a Range header field that consists of more than two overlapping ranges, or a set of many small ranges that are not listed in ascending order, since both are indications of either a broken client or a deliberate denial-of-service attack (Section 6.1). A client SHOULD NOT request multiple ranges that are inherently less efficient to process and transfer than a single range that encompasses the same data.

A client that is requesting multiple ranges SHOULD list those ranges in ascending order (the order in which they would typically be received in a complete representation) unless there is a specific need to request a later part earlier. For example, a user agent processing a large representation with an internal catalog of parts might need to request later parts first, particularly if the representation consists of pages stored in reverse order and the user agent wishes to transfer one page at a time.

The Range header field is evaluated after evaluating the precondition header fields defined in [RFC7232], and only if the result in absence of the Range header field would be a 200 (OK) response. In other words, Range is ignored when a conditional GET would result in a 304 (Not Modified) response.

The If-Range header field (Section 3.2) can be used as a precondition to applying the Range header field.

If all of the preconditions are true, the server supports the Range header field for the target resource, and the specified range(s) are valid and satisfiable (as defined in Section 2.1), the server SHOULD send a 206 (Partial Content) response with a payload containing one or more partial representations that correspond to the satisfiable ranges requested, as defined in Section 4.

If all of the preconditions are true, the server supports the Range header field for the target resource, and the specified range(s) are invalid or unsatisfiable, the server SHOULD send a 416 (Range Not Satisfiable) response.

3.2. 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 precondition 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 as follows: if the representation is unchanged, send me the part(s) that I am requesting in Range; otherwise, send me the entire representation.

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

A client MUST NOT generate an If-Range header field in a request that does not contain a Range header field. A server MUST ignore an If-Range header field received in a request that does not contain a Range header field. An origin server MUST ignore an If-Range header field received in a request for a target resource that does not support Range requests.

A client MUST NOT generate an If-Range header field containing an entity-tag that is marked as weak. A client MUST NOT generate an If-Range header field containing an HTTP-date unless the client has no entity-tag for the corresponding representation and the date is a strong validator in the sense defined by Section 2.2.2 of [RFC7232].

A server that evaluates an If-Range precondition MUST use the strong comparison function when comparing entity-tags (Section 2.3.2 of [RFC7232]) and MUST evaluate the condition as false if an HTTP-date validator is provided that is not a strong validator in the sense defined by Section 2.2.2 of [RFC7232]. A valid entity-tag can be distinguished from a valid HTTP-date by examining the first two characters for a DQUOTE.

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 process the Range header field as requested. If the validator does not match, the server MUST ignore the Range header field. Note that this comparison by exact match, including when the validator is an HTTP-date, differs from the "earlier than or equal to" comparison used when evaluating an If-Unmodified-Since conditional.

4. Responses to a Range Request

4.1. 206 Partial Content

The *206 (Partial Content)* status code indicates that the server is successfully fulfilling a range request for the target resource by transferring one or more parts of the selected representation that correspond to the satisfiable ranges found in the request's Range header field (Section 3.1).

If a single part is being transferred, the server generating the 206 response MUST generate a Content-Range header field, describing what range of the selected representation is enclosed, and a payload consisting of the range. 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

... 26012 bytes of partial image data ...
```

If multiple parts are being transferred, the server generating the 206 response MUST generate a "multipart/byteranges" payload, as defined in Appendix A, and a Content-Type header field containing the multipart/byteranges media type and its required boundary parameter. To avoid confusion with single-part responses, a server MUST NOT generate a Content-Range header field in the HTTP header section of a multiple part response (this field will be sent in each part instead).

Within the header area of each body part in the multipart payload, the server MUST generate a Content-Range header field corresponding to the range being enclosed in that body part. If the selected representation would have had a Content-Type header field in a 200 (OK) response, the server SHOULD generate that same Content-Type field in the header area of each body part. 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-Length: 1741
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--
```

When multiple ranges are requested, a server MAY coalesce any of the ranges that overlap, or that are separated by a gap that is smaller than the overhead of sending multiple parts, regardless of the order in which the corresponding byte-range-spec appeared in the received Range header field. Since the typical overhead between parts of a multipart/byteranges payload is around 80 bytes, depending on the selected representation's

media type and the chosen boundary parameter length, it can be less efficient to transfer many small disjoint parts than it is to transfer the entire selected representation.

A server MUST NOT generate a multipart response to a request for a single range, since a client that does not request multiple parts might not support multipart responses. However, a server MAY generate a multipart/byteranges payload with only a single body part if multiple ranges were requested and only one range was found to be satisfiable or only one range remained after coalescing. A client that cannot process a multipart/byteranges response MUST NOT generate a request that asks for multiple ranges.

When a multipart response payload is generated, the server SHOULD send the parts in the same order that the corresponding byte-range-spec appeared in the received Range header field, excluding those ranges that were deemed unsatisfiable or that were coalesced into other ranges. A client that receives a multipart response MUST inspect the Content-Range header field present in each body part in order to determine which range is contained in that body part; a client cannot rely on receiving the same ranges that it requested, nor the same order that it requested.

When a 206 response is generated, the server MUST generate the following header fields, in addition to those required above, if the field would have been sent in a 200 (OK) response to the same request: Date, Cache-Control, ETag, Expires, Content-Location, and Vary.

If a 206 is generated in response to a request with an If-Range header field, the sender SHOULD NOT generate other representation header fields beyond those required above, because the client is understood to already have a prior response containing those header fields. Otherwise, the sender MUST generate all of the representation header fields that would have been sent in a 200 (OK) response to the same request.

A 206 response is cacheable by default; i.e., unless otherwise indicated by explicit cache controls (see Section 4.2.2 of [RFC7234]).

4.2. Content-Range

The "Content-Range" header field is sent in a single part 206 (Partial Content) response to indicate the partial range of the selected representation enclosed as the message payload, sent in each part of a multipart 206 response to indicate the range enclosed within each body part, and sent in 416 (Range Not Satisfiable) responses to provide information about the selected representation.

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

byte-content-range  = bytes-unit SP
                      ( byte-range-resp / unsatisfied-range )

byte-range-resp     = byte-range "/" ( complete-length / "*" )
byte-range          = first-byte-pos "-" last-byte-pos
unsatisfied-range   = "*/" complete-length

complete-length     = 1*DIGIT

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

If a 206 (Partial Content) response contains a Content-Range header field with a range unit (Section 2) that the recipient does not understand, the recipient MUST NOT attempt to recombine it with a stored representation. A proxy that receives such a message SHOULD forward it downstream.

For byte ranges, a sender SHOULD indicate the complete length of the representation from which the range has been extracted, unless the complete length is unknown or difficult to determine. An asterisk character ("*") in

place of the complete-length indicates that the representation length was unknown when the header field was generated.

The following example illustrates when the complete length of the selected representation is known by the sender to be 1234 bytes:

```
Content-Range: bytes 42-1233/1234
```

and this second example illustrates when the complete length is unknown:

```
Content-Range: bytes 42-1233/*
```

A Content-Range field value is invalid if it contains a byte-range-resp that has a last-byte-pos value less than its first-byte-pos value, or a complete-length value less than or equal to its last-byte-pos value. The recipient of an invalid Content-Range MUST NOT attempt to recombine the received content with a stored representation.

A server generating a 416 (Range Not Satisfiable) response to a byte-range request SHOULD send a Content-Range header field with an unsatisfied-range value, as in the following example:

```
Content-Range: bytes */1234
```

The complete-length in a 416 response indicates the current length of the selected representation.

The Content-Range header field has no meaning for status codes that do not explicitly describe its semantic. For this specification, only the 206 (Partial Content) and 416 (Range Not Satisfiable) status codes describe a meaning for Content-Range.

The following are examples of Content-Range values in which the selected representation contains a total of 1234 bytes:

- The first 500 bytes:

```
Content-Range: bytes 0-499/1234
```

- The second 500 bytes:

```
Content-Range: bytes 500-999/1234
```

- All except for the first 500 bytes:

```
Content-Range: bytes 500-1233/1234
```

- The last 500 bytes:

```
Content-Range: bytes 734-1233/1234
```

4.3. 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 (Section 2.1 of [RFC7232]).

A client that has received multiple partial responses to GET requests on a target resource MAY combine those responses into a larger continuous range if they share the same strong validator.

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

If the most recent 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 recent 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 client MUST process the combined response as if it were a complete 200 (OK) response, including a Content-Length header field that reflects the complete length. Otherwise, the client MUST process the set of continuous ranges as one of the following: an incomplete 200 (OK) response if the combined response is a prefix of the representation, a single 206 (Partial Content) response containing a multipart/byteranges body, or multiple 206 (Partial Content) responses, each with one continuous range that is indicated by a Content-Range header field.

4.4. 416 Range Not Satisfiable

The *416 (Range Not Satisfiable)* status code indicates that none of the ranges in the request's Range header field (Section 3.1) overlap the current extent of the selected resource or that the set of ranges requested has been rejected due to invalid ranges or an excessive request of small or overlapping ranges.

For byte ranges, failing to overlap the current extent means that the first-byte-pos of all of the byte-range-spec values were greater than the current length of the selected representation. When this status code is generated in response to a byte-range request, the sender SHOULD generate a Content-Range header field specifying the current length of the selected representation (Section 4.2).

For example:

```
HTTP/1.1 416 Range Not Satisfiable
Date: Fri, 20 Jan 2012 15:41:54 GMT
Content-Range: bytes */47022
```

Note: Because servers are free to ignore Range, many implementations will simply respond with the entire selected representation in a 200 (OK) response. That is partly because most clients are prepared to receive a 200 (OK) to complete the task (albeit less efficiently) and partly because clients might not stop making an invalid partial request until they have received a complete representation. Thus, clients cannot depend on receiving a 416 (Range Not Satisfiable) response even when it is most appropriate.

5. IANA Considerations

5.1. Range Unit Registry

The "HTTP Range Unit Registry" defines the namespace for the range unit names and refers to their corresponding specifications. The registry has been created and is now maintained at <http://www.iana.org/assignments/http-parameters>.

5.1.1. Procedure

Registration of an HTTP Range Unit MUST include the following fields:

- Name
- Description
- Pointer to specification text

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

5.1.2. Registrations

The initial range unit registry contains the registrations below:

Range Unit Name	Description	Reference
bytes	a range of octets	Section 2.1
none	reserved as keyword, indicating no ranges are supported	Section 2.3

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

5.2. Status Code Registration

The "Hypertext Transfer Protocol (HTTP) Status Code Registry" located at <http://www.iana.org/assignments/http-status-codes> has been updated to include the registrations below:

Value	Description	Reference
206	Partial Content	Section 4.1
416	Range Not Satisfiable	Section 4.4

5.3. Header Field Registration

HTTP header fields are registered within the "Message Headers" registry maintained at <http://www.iana.org/assignments/message-headers/>.

This document defines the following HTTP header fields, so their associated registry entries have been updated according to the permanent registrations below (see [BCP90]):

Header Field Name	Protocol	Status	Reference
Accept-Ranges	http	standard	Section 2.3
Content-Range	http	standard	Section 4.2
If-Range	http	standard	Section 3.2
Range	http	standard	Section 3.1

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

5.4. Internet Media Type Registration

IANA maintains the registry of Internet media types [BCP13] at <http://www.iana.org/assignments/media-types>.

This document serves as the specification for the Internet media type "multipart/byteranges". The following has been registered with IANA.

5.4.1. Internet Media Type multipart/byteranges

Type name:	multipart	
Subtype name:	byteranges	
Required parameters:	boundary	
Optional parameters:	N/A	
Encoding considerations:	only "7bit", "8bit", or "binary" are permitted	
Security considerations:	see Section 6	
Interoperability considerations:	N/A	
Published specification:	This specification (see Appendix A).	
Applications that use this media type:	HTTP components supporting multiple ranges in a single request.	
Fragment identifier considerations:	N/A	
Additional information:	Deprecated alias names for this type:	N/A
	Magic number(s):	N/A
	File extension(s):	N/A
	Macintosh file type code(s):	N/A
Person and email address to contact for further information:	See Authors' Addresses section.	
Intended usage:	COMMON	
Restrictions on usage:	N/A	
Author:	See Authors' Addresses section.	
Change controller:	IESG	

6. Security Considerations

This section is meant to inform developers, information providers, and users of known security concerns specific to the HTTP range request mechanisms. More general security considerations are addressed in HTTP messaging [RFC7230] and semantics [RFC7231].

6.1. Denial-of-Service Attacks Using Range

Unconstrained multiple range requests are susceptible to denial-of-service attacks because the effort required to request many overlapping ranges of the same data is tiny compared to the time, memory, and bandwidth consumed by attempting to serve the requested data in many parts. Servers ought to ignore, coalesce, or reject egregious range requests, such as requests for more than two overlapping ranges or for many small ranges in a single set, particularly when the ranges are requested out of order for no apparent reason. Multipart range requests are not designed to support random access.

7. Acknowledgments

See Section 10 of [RFC7230].

8. References

8.1. Normative References

[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.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

8.2. Informative References

[BCP13] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[BCP90] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[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.

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

A. Internet Media Type multipart/byteranges

When a 206 (Partial Content) response message includes the content of multiple ranges, they are transmitted as body parts in a multipart message body ([RFC2046], Section 5.1) with the media type of "multipart/byteranges".

The multipart/byteranges media type includes one or more body 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.

Implementation Notes:

1. Additional CRLFs might 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 that used a media type of multipart/x-byteranges, which is almost (but not quite) compatible with this type.

Despite the name, the "multipart/byteranges" media type is not limited to byte ranges. The following example uses an "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-Length: 2331785
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--
```

B. Changes from RFC 2616

Servers are given more leeway in how they respond to a range request, in order to mitigate abuse by malicious (or just greedy) clients. (Section 3.1)

A weak validator cannot be used in a 206 response. (Section 4.1)

The Content-Range header field only has meaning when the status code explicitly defines its use. (Section 4.2)

This specification introduces a Range Unit Registry. (Section 5.1)

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

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 [RFC7230]:

```
OWS        = <OWS, see [RFC7230], Section 3.2.3>
token      = <token, see [RFC7230], Section 3.2.6>
```

The rules below are defined in other parts:

```
HTTP-date  = <HTTP-date, see [RFC7231], Section 7.1.1.1>
entity-tag = <entity-tag, see [RFC7232], Section 2.3>
```

D. Collected ABNF

In the collected ABNF below, list rules are expanded as per Section 1.2 of [RFC7230].

```
Accept-Ranges = acceptable-ranges

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

HTTP-date = <HTTP-date, see [RFC7231], Section 7.1.1.1>

If-Range = entity-tag / HTTP-date

OWS = <OWS, see [RFC7230], Section 3.2.3>

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

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

byte-content-range = bytes-unit SP ( byte-range-resp /
 unsatisfied-range )
byte-range = first-byte-pos "-" last-byte-pos
byte-range-resp = byte-range "/" ( complete-length / "*" )
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"

complete-length = 1*DIGIT

entity-tag = <entity-tag, see [RFC7232], Section 2.3>

first-byte-pos = 1*DIGIT

last-byte-pos = 1*DIGIT

other-content-range = other-range-unit SP other-range-resp
other-range-resp = *CHAR
other-range-set = 1*VCHAR
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, see [RFC7230], Section 3.2.6>

unsatisfied-range = "*/" complete-length
```

Index

2
206 Partial Content (status code) **9**, 13, 19

4
416 Range Not Satisfiable (status code) **12**, 13

A
Accept-Ranges header field 5, **6**, 13, 13

B
BCP13 13, **17**
BCP90 13, **17**

C
Content-Range header field 5, **10**, 12, 13, 19

G
Grammar
`Accept-Ranges` **6**
`acceptable-ranges` **6**
`byte-content-range` **10**
`byte-range` **10**
`byte-range-resp` **10**
`byte-range-set` **5**
`byte-range-spec` **5**
`byte-ranges-specifier` **5**
`bytes-unit` 5, **5**
`complete-length` **10**
`Content-Range` **10**
`first-byte-pos` **5**
`If-Range` **7**
`last-byte-pos` **5**
`other-content-range` **10**
`other-range-resp` **10**
`other-range-unit` 5, **6**
`Range` **7**
`range-unit` **5**
`ranges-specifier` **5**
`suffix-byte-range-spec` **5**
`suffix-length` **5**
`unsatisfied-range` **10**

I
If-Range header field 7, **7**, 13

M
Media Type
multipart/byteranges **14**, **18**
multipart/x-byteranges 18
multipart/byteranges Media Type **14**, **18**
multipart/x-byteranges Media Type 18

R
Range header field 5, **7**, 9, 12, 13, 19
RFC2046 **17**, 18, 18
Section 5.1 18
RFC2119 4, **17**
RFC2616 **17**
RFC5226 13, **17**
Section 4.1 13
RFC5234 4, **17**, 20
Appendix B.1 20
RFC7230 4, 4, 15, 16, **17**, 20, 20, 20, 21
Section 1.2 21
Section 2.5 4
Section 3.2.3 20
Section 3.2.6 20
Section 7 4
Section 10 16
RFC7231 5, 15, **17**, 20
Section 3 5
Section 7.1.1.1 20
RFC7232 7, 8, 8, 8, 11, **17**, 20
Section 2.1 11
Section 2.2.2 8, 8
Section 2.3 20
Section 2.3.2 8
RFC7234 10, **17**
Section 4.2.2 10

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/