HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: March 8, 2018

C. Pratt
CableLabs
B. Stark
AT&T
D. Thakore
CableLabs

September 4, 2017

HTTP Random Access and Live Content
draft-ietf-httpbis-rand-access-live-01

Abstract

To accommodate byte range requests for content that has data appended over time, this document defines semantics that allow a HTTP client and server to perform byte-range GET and HEAD requests that start at an arbitrary byte offset within the representation and ends at an indeterminate offset.

Editorial Note (To be removed by RFC Editor before publication)

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

Working Group information can be found at <http://httpwg.github.io/>; source code and issues list for this draft can be found at <https://github.com/httpwg/http-extensions/labels/rand-access-live>.

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 8, 2018.

Table of Contents

1. Introduction

Some Hypertext Transfer Protocol (HTTP) Clients use byte-range requests (Range requests using the "bytes" Range Unit) to transfer select portions of large representations. And in some cases large representations require content to be continuously or periodically appended - such as representations consisting of live audio or video

sources, blockchain databases, and log files. Clients cannot access the appended/live content using a Range request with the bytes range unit using the currently defined byte-range semantics without accepting performance or behavior sacrifices which are not acceptable for many applications.

For instance, HTTP Clients have the ability to access appended content by simply transferring the entire accessible portion of the representation from the beginning and continuing to read the appended content as it's made available. Obviously, this is highly inefficient for cases where the representation is large and only a portion of the randomly-accessible content is needed by the Client. And when bandwidth is limited, the Client may never be able to transfer all of the currently accessible potion and "catch up" with the appending content.

Alternatively, Clients can also access appended content by sending periodic open-ended bytes Range requests using the last-known end byte position as the range start. Performing low-frequency periodic bytes Range requests in this fashion (polling) introduces latency since the Client will necessarily be somewhat behind the aggregated content - mimicking the behavior (and latency) of segmented content representations such as HLS or MPEG-DASH. And performing these Range requests at higher frequency incurs more processing overhead and HTTP exchanges as the requests will often return no content - since content is usually aggregated in groups of bytes (e.g. a video frame, audio sample, block, or log entry).

To accommodate byte-range requests on large representations which have data appended over time efficiently and with low latency, this recommendation defines semantics whereby the HTTP Client performs byte-range requests using a combination of open-ended byte-range HEAD requests and GET requests using "Large Value" last-byte-pos values.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Performing Range requests on Random-Access Aggregating ("live") Content

There are two critical operations for accessing randomly accessing live/aggregating representations:

- Establishing the randomly-accessible range of the representation, and
- Performing range requests that continue beyond the randomly-accessible range.

2.1. Establishing the Randomly Accessible Byte Range

Establishing if a representation is continuously aggregating ("live") and determining the randomly-accessible byte range can both be determined using the existing definition for an open-ended byte-range request. Specifically, [RFC7233] defines a byte-range request of the form:

```
byte-range-spec = first-byte-pos "-" [ last-byte-pos ]
```

which allows a Client to send a HEAD request with a first-byte-pos and leave last-byte-pos absent. A Server that receives a satisfiable byte-range request (with first-byte-pos smaller than the current representation length) must respond with a 206 status code (Partial Content) with a Content-Range header indicating the currently satisfiable byte range. For example, a Client-issued HEAD request performed against a continuously aggregating representation hosted on a Server could contain a byte-range header of the form:

```
Range: bytes=0-
```

could return the header

```
Content-Range: bytes 0-1234567/*
```

from the Server indicating that (1) the complete representation length is unknown (via the "*" in place of the complete-length field) and (2) that only bytes 0-1234567 were accessable at the time the request was processed by the Server. The Client can infer from this

response that bytes 0-1234567 of the representation can be requested and returned in a timely fashion (the bytes are immediately available).

2.2. Byte-Range Requests Beyond the Randomly Accessible Byte Range

Once a Client has determined that a representation has an indeterminate length and established the byte range that can be accessed, it may want to perform a request with a start position within the randomly-accessible content range and an end position at an indefinite "live" point - a point where the byte-range GET request is fulfilled on-demand as the content is aggregated.

For example, for a large video asset, a client may wish to start a content transfer from the video "key" frame immediately before the point of aggregation and continue the content transfer indefinitely as content is aggregated - in order to support low-latency startup of a live video stream.

Unlike a byte-range Range request, a byte-range Content-Range response header cannot be "open ended", per [RFC7233]:

```
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
```

Specifically, last-byte-pos is required in byte-range. So in order to preserve interoperability with existing HTTP clients, servers, proxies, and caches, this document proposes a mechanism for a Client to indicate support for handling an indeterminate-length byte-range response, and a mechanism for a Server to indicate if/when it's providing a indeterminate-length response.

A Client can indicate support for handling indeterminate-length byte-range responses by providing a Very Large Value for the last-byte-pos in the byte-range request. For example, a Client can perform a byte-

range GET request of the form:

```
Range: bytes=1230000-999999999999
```

where the last-byte-pos in the Request is much larger than the last-byte-pos returned in response to an open-ended byte-range HEAD request, as described above.

2.3. Byte-Range Responses Beyond the Randomly Accessible Byte Range

A Server may indicate that it is supplying a continuously aggregating ("live") response by supplying the Client request's last-byte-pos in the Content-Range response header.

For example:

```
Range: bytes=1230000-999999999999
```

could return

```
Content-Range: bytes 1230000-999999999999/*
```

from the Server to indicate that the response will start at byte 1230000 and continues indefinitely to include all aggregated content, as it becomes available.

A Server that doesn't support or supply a continuously aggregating ("live") response should supply the currently satisfiable byte range, as it would with an open-ended byte request.

For example:

```
Range: bytes=1230000-999999999999
```

could return

```
Content-Range: bytes 1230000-1234567/*
```

from the Server to indicate that the response will start at byte 1230000 and end at byte 1234567 and will not include any aggregated content. This is the response expected from a typical HTTP Server -

one that doesn't support byte-range requests on aggregating content.

A Client that doesn't receive a response indicating it is continuously aggregating must use other means to access aggregated content (e.g. periodic byte-range polling).

A Server that does return a continuously aggregating ("live") response should return data using chunked transfer coding and not provide a Content-Length header. A 0-length chunk indicates that aggregation of the transferring resource is permanently discontinued, per section 4.1 of [RFC7233].

3. Other Applications of Random-Access Aggregating Content

3.1. Requests Starting at the Aggregation ("Live") Point

If a Client would like to start the content transfer at the Aggregation ("live") point without including any randomly-accessible portion of the representation, then it should supply the last-byte-pos from the most-recently received byte-range-spec and a Very Large Value for the last-byte-pos in the byte-range request.

For example a HEAD request containing:

```
Range: bytes=0-
```

could return

```
Content-Range: bytes 0-1234567/*
```

and a GET request containing

```
Range: bytes=1234567-999999999999
```


could return

```
Content-Range: bytes 1234567-999999999999/*
```

with the response body starting with continuously aggregating ("live") data and continuing indefinitely.

3.2. Shift Buffer Representations

Some representations lend themselves to front-end content deletion in addition to aggregation. While still supporting random access, representations of this type have a portion at the beginning (the "0" end) of the randomly-accessible region that become inaccessible over time. Examples of this kind of representation would be an audio-video time-shift buffer or a rolling log file.

For example a HEAD Range request containing:

```
Range: bytes=0-
```

could return

```
Content-Range: bytes 1000000-1234567/*
```

indicating that the first 1000000 bytes were not accessible at the time the HEAD request was processed. Subsequent HEAD requests could return:

```
Content-Range: bytes 1000000-1234567/*

Content-Range: bytes 1010000-1244567/*

Content-Range: bytes 1020000-1254567/*
```

Note though that the difference between the first-byte-pos and last-byte-pos need not be constant.

The Client could then follow-up with a GET Range request containing

```
Range: bytes=1020000-999999999999
```

with the Server returning

```
Content-Range: bytes 1020000-999999999999/*
```

with the response body returning bytes 1020000-1254567 immediately and aggregated ("live") data being returned as the content is aggregated.

4. Security Considerations

One potential issue with this recommendation is related to the use of very-large last-byte-pos values. Some Client and Server implementations may not be prepared to deal with byte position values of 2^^63 and beyond. So in applications where there's no expectation that the representation will ever exceed 2^^63, a value smaller than this value should be used as the Very Large last-byte-pos in a byte-seek request or content-range response.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/info/rfc7233>.

5.2. Informative References

[RANGE-UNIT-REGISTRY] IANA, "Hypertext Transfer Protocol (HTTP) Parameters", 2016, <http://www.iana.org/assignments/http-parameters/http-parameters.xhtml#range-units>.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, DOI 10.17487/RFC4234, October 2005, <https://www.rfc-editor.org/info/rfc4234>.

Appendix A. Acknowledgements

Mark Nottingham, Patrick McManus, Julian Reschke, Remy Lebeau, Rodger Combs, Thorsten Lohmar, Martin Thompson, Adrien de Croy, K. Morgan, Roy T. Fielding, Jeremy Poulter.

Authors' Addresses

Craig Pratt
CableLabs
858 Coal Creek Circle
Louisville, CO 80027

Email: pratt@acm.org

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com

Darshak Thakore
CableLabs
858 Coal Creek Circle
Louisville, CO 80027

Email: d.thakore@cablelabs.com