Byte Range PATCH

A media type for writing at offsets (part 2)

Austin Wright, July 2023
Review: The Problem

• Problem: I only want to change the first four bytes of a file over HTTP

• Current solutions:
  • Endpoint-specific POST URI  
    
  • Endpoint-specific URI format that identifies only selected bytes, e.g. 
    
  • RFC 9110 Content-Range PUT request 

• All require prior coordination, none can be made opportunistically
Review: Content-Range in PUT
RFC 9110 Content-Range in PUT is insufficient for many applications

- Requires prior agreement, otherwise it will overwrite the entire resource
  - There’s no benefit over using POST endpoint
  - No way to safely opt back out: Once implemented in clients, removing support will cause corruption.

- Content-Range does not permit indeterminate length payloads
  - e.g. live streams that may continue indefinitely

- A media type is useful for describing changes outside the context of an HTTP request
Proposal

• Create a media type for PATCH

• Re-use the RFC 9110 Content-Range-in-PUT semantics in multipart/byteranges

• Also define message/byterange and application/byteranges, since decoding multipart messages is unpopular (it requires scanning of the part body)
To anticipate a question
“Isn’t three media types excessive?”

(1) multipart/byteranges already exists

(2) application/byteranges adapts it to a binary format because many people really despise multipart

(3) message/byterange is by far the simplest, if you only need to patch a single range

(4) Successful binary formats tend to have a human-readable counterpart
Example: Segmented Uploads
Create a new file across two requests

PATCH /data/bulk.json HTTP/1.1
Content-Type: message/byterange
If-None-Match: *

Content-Range: 0-99/200
Content-Type: application/json

First 100 bytes of content...

PATCH /data/bulk.json HTTP/1.1
Content-Type: message/byterange
If-Match: “e4912”

Content-Range: 100-199/200
Content-Type: application/json

...Last 100 bytes of content
Update 1: Indeterminate length writes

- When you don’t know how much data you’re sending at the start, e.g. live streams

- Content-Range: 300-/*

  - Not a valid Content-Range header (missing end range), but special cased for this situation

  - Could potentially be adopted in HTTP to support indefinitely long 206 Partial Content responses. (But that’s a question for HTTP WG.)
Update 2: Set Size/Truncate

• A common filesystem operation

• Indicated with the unsatisfied-range form, e.g.
  
  • Content-Range: */0 (truncate to 0)

• Included on the principle that patch operations each have an inverse… So if you can use message/byterange to append, then you should be able to undo that with a truncate, too.
Update 3: add Prefer: transaction

- Define Prefer: transaction=atomic or transaction=partial to designate how to handle uploaded data after an interruption.

- Not strictly related, but adds lots of value: enables certain cases of resuming interrupted uploads
  - Suitable for any HTTP request
Implementation (message/byterange)


Patch editor

/D foam

Ascii

Dorothy lived in the midst of the great London prairies, with Uncle Henry, who was a farmer, and Aunt Em, who was the farmer's wife. Their house

1. PATCH /foo.txt HTTP/1.1
   Content-Type: message/byterange

   Content-Range: bytes 40-45/144
   Content-Length: 6

   London
Implementation (binary)

Patch editor

 PATCH /foo.txt HTTP/1.1
  Content-Type: application/byteranges


1. PATCH /foo.txt HTTP/1.1
   Content-Type: application/byteranges
   Content-Range: / Content-Length: 6

Dorothy lived in the midst of the great London prairies, with Uncle Henry, who was a farmer, and Aunt Em, who was the farmer's wife. Their house
Questions?