

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
<http://example.com/logs.update>
 - Endpoint-specific URI format that identifies only selected bytes, e.g.
<http://example.com/logs?bytes=200-299>
 - 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)

<https://awwright.github.io/http-progress/demo-patch-byterange-server/client.xhtml>

Patch editor

Overwrite

Offset	Hex	ASCII
000000	44 6F 72 6F 74 68 79 20 6C 69 76 65 64 20 69 6E	Dorothy lived in
000010	20 74 68 65 20 6D 69 64 73 74 20 6F 66 20 74 68	the midst of th
000020	65 20 67 72 65 61 74 20 4C 6F 6E 64 6F 6E 20 70	e great London p
000030	72 61 69 72 69 65 73 2C 20 77 69 74 68 20 55 6E	rairies, with Un
000040	63 6C 65 20 48 65 6E 72 79 2C 20 77 68 6F 20 77	cle Henry, who w
000050	61 73 20 61 20 66 61 72 6D 65 72 2C 20 61 6E 64	as a farmer, and
000060	20 41 75 6E 74 20 45 6D 2C 20 77 68 6F 20 77 61	Aunt Em, who wa
000070	73 20 74 68 65 20 66 61 72 6D 65 72 27 73 20 77	s the farmer's w
000080	69 66 65 2E 20 54 68 65 69 72 20 68 6F 75 73 65	ife. Their house

Request body display

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

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

London

Implementation (binary)

<https://awwright.github.io/http-progress/demo-patch-byterange-server/client.xhtml>

Patch editor

Overwrite

Offset	Hex	ASCII
000000	44 6F 72 6F 74 68 79 20 6C 69 76 65 64 20 69 6E	Dorothy lived in
000010	20 74 68 65 20 6D 69 64 73 74 20 6F 66 20 74 68	the midst of th
000020	65 20 67 72 65 61 74 20 4C 6F 6E 64 6F 6E 20 70	e great London p
000030	72 61 69 72 69 65 73 2C 20 77 69 74 68 20 55 6E	rairies, with Un
000040	63 6C 65 20 48 65 6E 72 79 2C 20 77 68 6F 20 77	cle Henry, who w
000050	61 73 20 61 20 66 61 72 6D 65 72 2C 20 61 6E 64	as a farmer, and
000060	20 41 75 6E 74 20 45 6D 2C 20 77 68 6F 20 77 61	Aunt Em, who wa
000070	73 20 74 68 65 20 66 61 72 6D 65 72 27 73 20 77	s the farmer's w
000080	69 66 65 2E 20 54 68 65 69 72 20 68 6F 75 73 65	ife. Their house

Request body display

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

```
08 2f 0d 43 6f 6e 74 65 6e 74 2d 52 61 6e 67 65  BS /CRContent-Range
0f 62 79 74 65 73 20 34 30 2d 34 35 2f 31 34 34  SI bytes 40-45/144
0e 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 01  SO Content-Length SOH
36 06 4c 6f 6e 64 6f 6e 6 ACK London
```

Questions?