

# Byte Range PATCH

**A media type for writing at offsets**

**Austin Wright, March 2023 [v2]**

# Motivation

- 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 to identify only the selected bytes, e.g.  
<http://example.com/logs?bytes=200-299>
  - RFC 9110 Content-Range PUT request
- None of these can be worked into generic HTTP toolchains

# Workaround

## Endpoint-specific POST request

```
POST /log HTTP/1.1
Content-Type: application/x-www-urlencoded

range=4-7&bytes=EFGH
```

# Workaround

## Endpoint-specific URI format

```
GET /log?bytes=4-7 HTTP/1.1  
Content-Type: application/octet-stream
```

EFGH

```
PUT /log?bytes=4-7 HTTP/1.1  
Content-Type: application/octet-stream
```

WXYZ

# RFC 9110 Content-Range PUT

①      GET / HTTP/1.1                      HTTP/1.1 206 Partial Content  
Range: bytes=4275-4302                  Content-Range: 4275-4302/7550  
  
was ALICE'S SPOON! And nobod

---

②      **PUT** / HTTP/1.1                      HTTP/1.1 200 OK  
**Content-Range: 4275-4302/7550**  
  
was CAROL'S SPOON! And nobod

---

③      GET / HTTP/1.1                      HTTP/1.1 206 Partial Content  
Range: bytes=4275-4302                  Content-Range: 4275-4302/7550  
  
was CAROL'S SPOON! And nobod

# Content-Range PUT Shortcomings

**RFC 9110 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 breakage.
- Content-Range does not permit indeterminate length responses
  - e.g. live streams that may continue indefinitely
- A media type is useful for describing changes outside the context of an HTTP request

# Barriers

## Why this hasn't been standardized yet?

- HTTP resources aren't *exactly* files on a filesystem
- ... But they still both represent a string of bytes
- PATCH is relatively new
- And actually, RFC 9110 standardized a partial solution

# Use Cases

- Segmented/chunked uploads
- Resuming broken uploads
- Writes to block devices
- Optimizes many file formats, e.g. embedded databases, append-only files, media files with indexes at the start of the file.



# Use Cases: Segmented/chunked uploads

- PUT /file/data.json.000 HTTP/1.1  
PUT /file/0001.json.001 HTTP/1.1  
PUT /file/0002.json.002 HTTP/1.1
- Problem: Each resource will be malformed, client must perform endpoint-specific steps to recreate the correct resource

# Use Cases: Resuming broken uploads

## Resumable Uploads for HTTP

### Abstract

HTTP clients often encounter interrupted data transfers as a result of canceled requests or dropped connections. Prior to interruption, part of a representation may have been exchanged. To complete the data transfer of the entire representation, it is often desirable to issue subsequent requests that transfer only the remainder of the representation. HTTP range requests support this concept of resumable downloads from server to client. This document describes a mechanism that supports resumable uploads from client to server using HTTP. ¶

### 9.1. Upload-Offset

The `Upload-Offset` request and response header field is an Item Structured Header indicating the resumption offset of corresponding upload, counted in bytes. Its value **MUST** be an integer. Its ABNF is ¶

```
Upload-Offset = sf-integer
```

 ¶

# Use Cases: Appending to a log file

```
POST / HTTP/1.1
Host: logs.monitoring.us-west-1.amazonaws.com
X-Amz-Date: 20130315T092054Z
Authorization: AWS4-HMAC-SHA256 ...
User-Agent: FooBar/2.0
Accept: application/json
Content-Type: application/x-amz-json-1.1
Content-Length: 332
X-Amz-Target: Logs_20140328.PutLogEvents
```

```
{
  "logGroupName": "my-log-group",
  "logStreamName": "my-log-stream",
  "logEvents": [
    { "timestamp": 1396035378988, "message": "Example event 1" },
    { "timestamp": 1396035378988, "message": "Example event 2" }
  ]
}
```

# Use Cases: Optimized file formats

## Appending to a WAV file

32-bit little-endian file length  $\approx$  40M

00000000	52	49	46	46	e0	64	81	02	57	41	56	45	66	6d	74	20	RIFF.d..WAVEfmt
00000010	10	00	00	00	01	00	02	00	44	ac	00	00	10	b1	02	00	.....D.....
00000020	04	00	10	00	4c	49	53	54	b4	00	00	00	49	4e	46	4f	....LIST....INFO
00000030	49	41	52	54	18	00	00	00	54	68	65	20	4c	61	7a	69	IART....The Lazi
00000040	65	73	74	20	4d	65	6e	20	6f	6e	20	4d	61	72	73	00	est Men on Mars.
00000050	49	43	4d	54	2f	00	00	00	68	74	74	70	3a	2f	2f	77	ICMT/...http://w
00000060	77	77	2e	61	6c	62	69	6e	6f	62	6c	61	63	6b	73	68	ww.albinoblacksh
00000070	65	65	70	2e	63	6f	6d	2f	66	6c	61	73	68	2f	62	61	eep.com/flash/ba
00000080	73	65	2e	70	68	70	00	00	49	43	52	44	05	00	00	00	se.php..ICRD....
00000090	32	30	30	30	00	00	49	47	4e	52	05	00	00	00	47	61	2000..IGNR....Ga
000000a0	6d	65	00	00	49	4e	41	4d	1e	00	00	00	49	6e	76	61	me..INAM....Inva
000000b0	73	69	6f	6e	20	6f	66	20	74	68	65	20	47	61	62	62	sion of the Gabb
000000c0	65	72	20	52	6f	62	6f	74	73	00	49	53	46	54	0e	00	er Robots.ISFT..
000000d0	00	00	4c	61	76	66	35	39	2e	32	37	2e	31	30	30	00	..Lavf59.27.100.
000000e0	64	61	74	61	00	64	81	02	00	00	00	00	00	00	00	00	data.d.....
000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

# Byte range patch

## Functional requirements

- Opportunistic — the server will return 412 Unknown Media Type if unsupported
  - More useful than 400 Client Error for rejecting Content-Range PUT
- Maps directly to filesystem operations, including **append** and **overwrite** that many file formats are optimized for.
- Support for determinate length and indeterminate length resources
  - Writing log files (determinate length), or live streaming an audio feed (indeterminate length)

# HTTP extension mechanisms

- Use HTTP existing extension mechanisms
  - **Header/Field** — used for opportunistically signaling something to the server, but ignoring is OK.
  - **Method or media type** — used if some part of the request *must* be understood.
    - The PATCH method satisfies all the necessary semantics (create or update the target resource according to some enclosed instructions), so a new media type to encode these instructions is most suitable.

# Scope

- For simplicity, **limit to standard filesystem operations**. A small patch should cause a small write. More complicated operations may be implemented with a different patch media type, e.g.
  - support for prepending and splicing in the middle of a file, would be a new media type (`application/splice?`)
  - Sophisticated delta algorithms, e.g. VCDIFF, could be registered as a media type.
- Splices/insertions can cause the whole file to be rewritten. Servers shouldn't be required to implement this just to support simpler operations.

# Forward compatibility

## What about future extensions?

- By requiring that at least one Content-Range field to be present in the patch, use of a field different than Content-Range field can be used as an extension mechanism.



# Existing Formats

- Idea: 206 Partial Content with `multipart/byteranges` indicates the response is a wrapper, not the literal resource
- Likewise, the PATCH method indicates the request is not the literal resource, but instructions for processing.

```
--THIS_STRING_SEPARATES  
Content-Range: bytes 2-6/25  
Content-Type: text/plain
```

```
23456  
--THIS_STRING_SEPARATES  
Content-Range: bytes 17-21/25  
Content-Type: text/plain
```

```
78901  
--THIS_STRING_SEPARATES--
```

# Possible Syntaxes

- Existing `multipart/byterange` format
- Adapt existing `message/http` format for requests targeting only one range
- Create binary format — should be suitable for multipart responses too

# Example: Log files

- Provide a byte offset to ensure that the remote copy matches the local copy

```
PATCH /log/prod-prndl-mysql3/mysqld-2023-03-05.log HTTP/1.1  
Content-Type: message/byterange
```

**Content-Range: 4275-5130/\***

```
2022-10-26 23:47:59.500681Z Manifest version: 1  
2022-10-26 23:47:59.511891Z Loaded Persona Generation ID from manifest:1  
2022-10-26 23:47:59.512362Z PersonaType:3 is in the manifest  
2022-10-26 23:47:59.515182Z PersonaType:5 is in the manifest  
2022-10-26 23:47:59.515222Z PersonaType:4 is in the manifest  
2022-10-26 23:47:59.515263Z All default System/System Proxy present  
2022-10-26 23:47:59.535016Z Loaded persona manifest
```

# Example: Log files

- Or omit the offset, and append to the end of the resource

```
PATCH /log/prod-prndl-mysql3/mysqld-2023-03-05.log HTTP/1.1
Content-Type: message/byterange
```

**Content-Range: \*/\***

```
2022-10-26 23:47:59.500681Z Manifest version: 1
2022-10-26 23:47:59.511891Z Loaded Persona Generation ID from manifest:1
2022-10-26 23:47:59.512362Z PersonaType:3 is in the manifest
2022-10-26 23:47:59.515182Z PersonaType:5 is in the manifest
2022-10-26 23:47:59.515222Z PersonaType:4 is in the manifest
2022-10-26 23:47:59.515263Z All default System/System Proxy present
2022-10-26 23:47:59.535016Z Loaded persona manifest
```

# Example: Segmented Uploads

- Upload a file in parts, or complete uploading an interrupted upload.
  - Nit: Resuming an interrupted upload requires that the server preserves state from an unfinished request.

```
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...

# Questions?