Resumable Uploads

draft-ietf-httpbis-resumable-upload
IETF 118: November 10, 2023
Marius Kleidl

Hackathon (1/2)

- Project: Automated server conformity checker
  - Suite of tests based on the draft specification
  - Run against a server and report violations

  → Improve interoperability

  → Spot implementation issues

- https://github.com/tus/ietf-hackathon/tree/main/tests
- Thanks, Piotrek Zadroga

Hackathon (1/2)

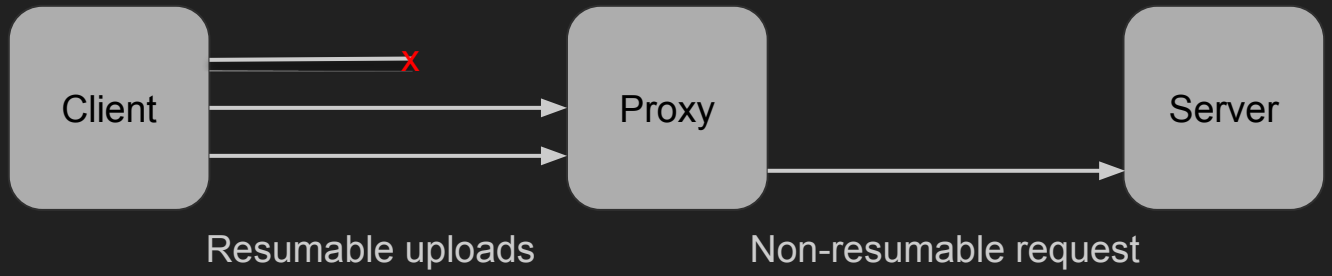
```
test_rufh.py ...FF..                                                     [100%]

================================================ FAILURES ================================================
_____________________________ TestHead.test_offset_retrieval_bad_head _____________________________

self = <test_rufh.TestHead object at 0x106435310>, post_with_file = <PreparedRequest [POST]>
session = <requests.sessions.Session object at 0x106437510>, head = <PreparedRequest [HEAD]>

    def test_offset_retrieval_bad_head(self, post_with_file, session, head):
        post_with_file.headers[UPLOAD_COMPLETE] = TRUE

        r = session.send(post_with_file)
        assert r.status_code == 104, "104 status code was expected"
        assert LOCATION in r.headers, "Location header was expected"
        assert r.headers[LOCATION], "Non-empty Location header value was expected"

        location = r.headers[LOCATION]
        # urlparse will throw an exception in case the location url is not correct
        # to pass the test, we don't expect the exception
        urlparse(location)
        head.url = location
        # The request MUST NOT include an Upload-Offset
        head.headers[UPLOAD_OFFSET] = str(10)
        r = session.send(head)
        # The request MUST NOT include an Upload-Offset or Upload-Complete header field.
        # The server MUST reject requests with either of these fields by responding with
        # a 400 (Bad Request) status code.
>       assert r.status_code == 400, "400 status code was expected"
E       AssertionError: 400 status code was expected
E       assert 204 == 400
E        +  where 204 = <Response [204]>.status_code

test_rufh.py:107: AssertionError
```

Hackathon (2/2)

- Project: Implement resumable uploads inside proxies
  - Idea: Transform resumable uploads into “regular” upload requests

    → No explicit support in application servers is needed

  - Plugin for Caddy: https://github.com/Murderlon/caddy-rufh (thanks, Merlijn Vos)
  - Plugin for Nginx: https://github.com/tus/ietf-hackathon/tree/main/nginx-rufh



Changes in -02 draft

- Use `Upload-Complete` instead of `Upload-Incomplete` (#2500)
- Clarify handling of `Content-Length` and `Upload-Complete` (#2544)
  - Server must reject requests with inconsistent `Content-Length`
- Editorial changes (thanks to Mark and Lucas)

PATCH request media type (#2501, #2610)

- PATCH requests need a media type in Content-Type
- Option 1 (current draft):
  - Request body is only the file to be uploaded → simple approach
  - Can we use `application/octet-stream`?
  - Should we use a new media type, e.g. `application/offset+octet-stream`?
- Option 2:
  - Use message/byterange from draft-wright-http-patch-byterange
  - Request body is an encapsulated patch document including the file chunk and offset
  - → more complex to implement
- Other options?

Responses to uploads (#2312)

- For “regular” upload requests the client receives a response, for example with information extracted from the uploaded file
- Resumable uploads should be a transparent upgrade to regular requests
- 1. What is the response to a resumable upload?
  - The response to the POST/PATCH requests, which completed the upload?
- 2. What happens if the client misses the response after the upload is completed?
  - Can we use an empty PATCH request to retrieve the response again?
- 3. What does a 500 status code mean in such a response?
  - Should the client resume the upload?
  - Or is this the final response if the server failed to process the uploaded file?

Upload progress via informational responses (#2291)

Server could send multiple 1xx responses with the current `Upload-Offset`

Allows the client to release memory associated with the transmitted data

```
POST /upload HTTP/1.1
upload-complete: ?1
content-length: 100
[content (100 bytes)]

HTTP/1.1 104 Upload Resumption Supported
location: https://example.com/upload/b530ce8ff

HTTP/1.1 104 Upload Resumption Supported
upload-offset: 50

HTTP/1.1 201 Created
location: https://example.com/upload/b530ce8ff
upload-offset: 100
```

Interoperability with HTTP digests (#2408)

- HTTP digests can be used to verify the integrity of the uploaded data
- Should we include additional details in the draft about how resumable uploads and digests interact?

Other open issues

- Error handling for Upload Creation Procedure (#2596)
- Allow 200 status code for offset retrieval (#2662)
- Fetch API proposal (WHATWG #1626)