Resumable Uploads

draft-ietf-httpbis-resumable-upload
HTTP Interim, 05 Oct 2022
Marius Kleidl

Open Issues

- Use of server-generated upload URI vs. client-generated token (#2239)
- Automatic feature detection and transparent upload upgrade (#2240, #2243)
- Compatibility with current browsers (#2247)

Upload Identifier

- #2239
- Question: What approach to use for identifying an upload?
- draft-ietf-httpbis-resumable-upload-00 uses a client-generated upload token
- tus v1 uses a server-generated upload URL

Upload Identifier: Client-generated Upload Token

- Client generates a random upload token before any data is transmitted
- Token is included in every request (e.g. Upload-Token: 10c83bf6…)
- Pros:
  - No additional round trip for upload creation is required
  - Upload can be resumed in any state (creation is idempotent)
  - Token can be used to trace requests through the system
- Cons:
  - Server must handle possible collision between tokens
  - Identification of upload resource is done using header and not the URI
- Used in draft-ietf-httpbis-resumable-upload-00

Upload Identifier: Server-generated upload URL

- Client sends a request for upload creation
- Server responds with URI for upload resource (e.g. Location: /files/1fba5…)
- Response can be 2XX or 1XX
- Pros:
  - Server is in full control of upload identifier
  - Upload creation fits nicely into usual request/response schema in HTTP
- Cons:
  - Data can only be transferred after the initial response
  - Upload can not be resumed if initial response is lost (upload creation is not idempotent)
- Used in tus v1 and many production services

Feature Detection

- #2240 and #2243
- Goal: Let the client easily discover that server supports resumable uploads
  - Allow transparent upgrades to resumable uploads, if possible
- Current approach:
  - Client indicates interest in resumable uploads using header (e.g. Prefer, Upload-Token)
  - Server responds with a 104 Resumption Supported status
- Problems:
  - What if the client does not or cannot receive the 104 status?
- Alternatives?

Browser Compatibility

- #2247
- Question: Can we make resumable upload compatible with current browsers?
- Fetch API does not expose access to 1XX responses
- Requiring the use of 1XX would hinder adoption of resumable uploads
- Goal: Allow resumable uploads using existing JavaScript APIs

Other Issues

- Prioritization of concurrent uploads (#2241)
- Upload-Incomplete is not interoperable (#2241)