httpbis Working Group
Internet-Draft
Intended status: Informational
Expires: December 30, 2019

A. Frindell
Facebook

June 28, 2019

HTTP Partial POST Replay
draft-frindell-httpbis-partial-post-replay-00

Abstract

This memo introduces a method of exchanging HTTP [RFC7230] messages between a web server and a cooperating intermediary - such as a reverse proxy load balancer - that enables faster restarts for the web server with minimal disruption for users.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 30, 2019.

1. Introduction

Web servers need to drain traffic periodically for configuration changes, software updates and maintenance. As continuous deployment becomes more common, the frequency of such events increases. When a server shuts down, it chooses whether to let all existing requests run to completion, or abort some or all in-progress requests. Aborted requests lead to poor user experiences including error messages or additional latency while the request is resent. Partial POST Replay makes it possible to eliminate this class of errors by handing off in-process requests to another server within a deployment.

1.1. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Partial POST Replay

This section describes the Partial POST Replay mechanism for handing off a request with a partially transferred entity body to another server instance.

2.1. Response Message

When the server begins restarting, it responds to any unprocessed requests with incomplete entity bodies with a new 3xx status code (TBD). The HTTP/1.1 status message is Partial POST Replay. Once this status is sent the server MUST NOT process this request other than is specified in this document.

The server MUST have prior knowledge that the intermediary supports Partial POST Replay before sending the 3xx response. If a server sends this response to an intermediary that does not understand it, the response will likely be forwarded back to the client.

2.1.1. Response Headers

Each request header is echoed in the response message with the prefix "Echo-". For example, the "User-Agent: Foo" request header would be included in the response as "Echo-User-Agent: Foo". HTTP/2 [RFC7540] and HTTP/3 {{?HTTP3} request pseudo-headers (beginning with ':') are echoed in the response message with the prefix "Pseudo-Echo-", and with the ':' removed. For example, ":path: /" is echoed as "pseudo-

echo-path: /". The server MUST NOT insert any Echo- or Pseudo-Echo headers in the response if the corresponding header was not present in the request.

Because there might be request body bytes in flight to the server when the 3xx response is generated, the length of the response body is unknown. The response SHOULD NOT include a "Content-Length" header (but will include a "Echo-Content-Length" header, if the request contained "Content-Length"). If the request protocol is HTTP/1.1, the server SHOULD use chunked transfer encoding for the response.

HTTP/1.1 server SHOULD include a "Connection: close" header in the response to prevent the intermediary from reusing the connection for a new request. HTTP/2 and HTTP/3 servers SHOULD emit a GOAWAY frame on each open connection when shutdown is initiated.

2.2. Intermediary Processing

Intermediaries MUST track the number of body bytes forwarded to the server for any request that could be replayed by the server. When an intermediary receives a 3xx status code from the server, it stops forwarding any new HTTP data from the client to this server. The intermediary does not forward the 3xx response to the client, but instead reconstructs the original HTTP request message from headers in the response beginning with the "Echo-" or "Pseudo-Echo" prefixes. Alternatively, if the intermediary retained a copy of the request it MAY use that and discard the response headers.

The intermediary can choose to buffer the response before selecting a new server, or can immediately select a new server and begin forwarding the request there. When the entire replayed request body has been sent to the new server, the intermediary can begin forwarding new HTTP data from the client to the new server.

If the intermediary receives more body bytes from the server than it forwarded, or if the response is terminated before receiving all forwarded bytes, the intermediary MUST fail the request with a 5xx status.

2.3. Original Request Termination

When the intermediary has received in the response body all of the request bytes forwarded to the original server, it completes the request message to the original server, according to the semantics of the transport protocol:

o For HTTP/1.0, the intermediary half-closes the connection

- For HTTP/1.1, the intermediary sends the final chunk terminator, or half-closes the connection if the request did not use chunked transfer encoding.

- For HTTP/2, the intermediary sends a DATA frame with the END_STREAM flag set on the request stream

- For HTTP/3, the intermediary sends a FIN on the request stream

When the server processes the end of the request, it completes the response message according to the semantics of the transport protocol.

Note that some HTTP server implementations treat the termination of the request with fewer bytes than specified in the Content-Length header as an error. Because all required information has been transferred to the intermediary before this error occurs, the server can abort the response and ignore the error without impacting the final status of the request.

It is possible that the entire entity body was sent by the intermediary before it received the Partial POST Replay status message. In this case the intermediary will receive the entire entity body in the response.

2.4. Preventing Loops

To prevent the intermediary from becoming stuck in an infinite redirect loop, it SHOULD add a 'Partial-Post-Replay: 1' header whenever forwarding to a new server. An intermediary that receives a redirect response with more "Echo-Partial-Post-Replay" headers than it supports SHOULD fail original request with a 5xx response.

3. Existing Solutions

There are several existing solutions to handling requests while draining traffic from a web server, but each has drawbacks that Partial POST Replay does not.

3.1. Drain Timeout

When servers stop accepting new connections, they often set a timeout during which existing requests can continue processing. At the end of the timeout, the server will abort any unfinished requests. During this phase, the server is not operating at full capacity, and requests that exceed the timeout are still terminated with error.

3.2. GOAWAY

HTTP/2 introduced the GOAWAY frame which a server can use to indicate which requests will not be processed, and which can be safely retried by the client. There are two problems with this mechanism.

First, the server cannot use this mechanism to refuse requests with stream IDs lower than the highest stream ID it has already processed. For example, if the server has received a partial request on stream ID=3, but has already begun processing a request on stream ID=5, it cannot send a GOAWAY with a Last-Stream-ID lower than 5. HTTP/2 does not have a status code that indicates an individual request is retryable

Second, an intermediary cannot seamlessly retry a POST request unless it has buffered the entire request body. Buffering all request bodies presents an enormous scalability challenge for intermediaries.

3.3. State Handover

Another possible technique is to pass state from a draining web server to a new instance. Such deployments start a new instance to handle new work in parallel with the instance that is shutting down. This requires that the system have enough resources to run two instances of the server simultaneously, for a potentially very long time.

4. Security Considerations

An intermediary must trust the server to echo back the headers and body of the original request. A malicious server could replay a different request to the intermediary, who would then send it to another server. The response to this forged request would be interpreted as a response to the original request.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[HTTP3] Bishop, M., Ed., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-latest (work in progress).

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

Acknowledgments

This draft evolved from a feature developed at Facebook. Thanks to Mohammad Husain, Woo Xie and David Langevin who worked on the initial implementation and deployment of this feature.

Author's Address

Alan Frindell
Facebook

Email: afrind@fb.com