HTTP
Internet-Draft
Intended status: Standards Track
Expires: November 1, 2017

P. McManus
Mozilla

April 30, 2017

HTTP Immutable Responses
draft-ietf-httpbis-immutable-02

Abstract

The immutable HTTP response Cache-Control extension allows servers to identify resources that will not be updated during their freshness lifetime. This assures that a client never needs to revalidate a cached fresh resource to be certain it has not been modified.

Note to Readers

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/ .

Working Group information can be found at http://httpwg.github.io/ ; source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions/labels/immutable .

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 http://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 November 1, 2017.

Copyright Notice

1. Introduction

HTTP's freshness lifetime mechanism [RFC7234] allows a client to safely reuse a stored response to satisfy future requests for a specified period of time. However, it is still possible that the resource will be modified during that period.

For instance, a front page newspaper photo with a freshness lifetime of one hour would mean that no user would see a cached photo more than one hour old. However, the photo could be updated at any time resulting in different users seeing different photos depending on the contents of their caches for up to one hour. This is compliant with the caching mechanism defined in [RFC7234].

Users that need to confirm there have been no updates to their cached responses typically use the reload (or refresh) mechanism in their user agents. This in turn generates a conditional request [RFC7232] and either a new representation or, if unmodified, a 304 (Not Modified) response [RFC7232] is returned. A user agent that understands HTML and fetches its dependent sub-resources might issue hundreds of conditional requests to refresh all portions of a common page [REQPERPAGE].

However some content providers never create more than one variant of a sub-resource, because they use "versioned" URLs. When these resources need an update they are simply published under a new URL, typically embedding an identifier unique to that version of the resource in the path, and references to the sub-resource are updated with the new path information.

For example, "https://www.example.com/101016/main.css" might be updated and republished as "https://www.example.com/102026/main.css", with any links that reference it being changed at the same time.

This design pattern allows a very large freshness lifetime to be used for the sub-resource without guessing when it will be updated in the future.

Unfortunately, the user agent does not know when this versioned URL design pattern is used. As a result, user-driven refreshes still translate into wasted conditional requests for each sub-resource as each will return 304 responses.

The "immutable" HTTP response Cache-Control extension allows servers to identify responses that will not be updated during their freshness lifetimes.

This effectively informs clients that any conditional request for that response can be safely skipped without worrying that it has been updated.

2. The immutable Cache-Control extension

When present in an HTTP response, the "immutable" Cache-Control extension indicates that the origin server will not update the representation of that resource during the freshness lifetime of the response.

Clients SHOULD NOT issue a conditional request during the response's freshness lifetime (e.g. upon a reload) unless explicitly overridden by the user (e.g. a force reload).

The immutable extension only applies during the freshness lifetime of the stored response. Stale responses SHOULD be revalidated as they normally would be in the absence of immutable.

The immutable extension takes no arguments. If any arguments are present, they have no meaning, and MUST be ignored. Multiple instances of the immutable extension are equivalent to one instance. The presence of an immutable Cache-Control extension in a request has no effect.

2.1. About Intermediaries

An immutable response has the same semantic meaning when received by proxy clients as it does when received by User-Agent based clients. Therefore proxies SHOULD skip conditionally revalidating fresh responses containing the immutable extension unless there is a signal from the client that a validation is necessary (e.g. a no-cache Cache-Control request directive).

A proxy that uses immutable to bypass a conditional revalidation may choose whether to reply with a 304 or 200 to its requesting client based on the request headers the proxy received.

2.2. Example

Cache-Control: max-age=31536000, immutable

3. Security Considerations

The immutable mechanism acts as form of soft pinning and, as with all pinning mechanisms, creates a vector for amplification of cache corruption incidents. These incidents include cache poisoning attacks. Three mechanisms are suggested for mitigation of this risk:

- Clients SHOULD ignore immutable from resources that are not part of an authenticated context such as HTTPS. Authenticated resources are less vulnerable to cache poisoning.

- User-Agents often provide two different refresh mechanisms: reload and some form of force-reload. The latter is used to rectify interrupted loads and other corruption. These reloads, typically indicated through no-cache request attributes, SHOULD ignore immutable as well.

- Clients SHOULD ignore immutable for resources that do not provide a strong indication that the stored response size is the correct response size such as responses delimited by connection close.

4. IANA Considerations

[RFC7234] sections 7.1 and 7.1.2 require registration of the immutable extension in the "Hypertext Transfer Protocol (HTTP) Cache

Directive Registry" with IETF Review.

- Cache-Directive: immutable

- Pointer to specification text: [this document]

5. Acknowledgments

Thank you to Ben Maurer for partnership in developing and testing this idea. Thank you to Amos Jeffries for help with proxy interactions and to Mark Nottingham for help with the documentation.

6. References

6.1. Normative References

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

6.2. Informative References

[REQPERPAGE] "HTTP Archive", n.d., <http://httparchive.org/interesting.php#reqTotal>.

Author's Address

Patrick McManus
Mozilla

Email: pmcmanus@mozilla.com