HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 31, 2019

M. Nottingham
Fastly

January 27, 2019

The Cache HTTP Response Header
draft-ietf-httpbis-cache-header-00

Abstract

To aid debugging, HTTP caches often append headers to a response detailing how they handled the request. This specification codifies that practice and updates it for HTTP's current caching model.

Note to Readers

RFC EDITOR: please remove this section before publication

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/ [1].

Working Group information can be found at https://httpwg.org/ [2]; source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions/labels/cache-header [3].

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 July 31, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

To aid debugging, HTTP caches often append headers to a response detailing how they handled the request.

Unfortunately, the semantics of these headers are often unclear, and both the semantics and syntax used vary greatly between implementations.

This specification defines a single, new HTTP response header field, "Cache" for this purpose.

For example:

```
Cache: HIT_FRESH; node="reverse-proxy.example.com:80";
                  key="https://example.com/foo|Accept-Encoding:gzip",
```

```
HIT_STALE; node="FooCDN parent"; fresh=-45; age=200; latency=3,
MISS; node="FooCDN edge"; fresh=-45; age=200; latency=98
```


1.1. Notational Conventions

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.

This document uses ABNF as defined in [RFC5234], along with the "%s" extension for case sensitivity defined in [RFC7405].

2. The Cache HTTP Response Header

The Cache HTTP response header indicates the handling of the request corresponding to the response it occurs within by caches along the path.

Its value is a Parameterised List [I-D.ietf-httpbis-header-structure]:

```
Cache   = sh-param-list
```

Each member of the parameterised list represents a cache that has handled the request.

The first member of the list represents the cache closest to the origin server, and the last member of the list represents the cache closest to the user agent (possibly including the user agent's cache itself, if it chooses to append a value).

Caches determine when it is appropriate to add the Cache header field to a response. Some might decide to add it to all responses, whereas others might only do so when specifically configured to, or when the request contains a header that activates a debugging mode.

When adding a value to the Cache header field, caches SHOULD preserve

the existing contents of the header, to allow debugging of the entire chain of caches handling the request.

Identifiers in the parameterised list members are expected to be cache-actions:


```
cache-action   = %s"HIT_FRESH"
               / %s"HIT_STALE"
               / %s"HIT_REFRESH_MODIFIED"
               / %s"HIT_REFRESH_NOT_MODIFIED"
               / %s"HIT_REFRESH_STALE"
               / %s"MISS"
               / %s"MISS_CLIENT"
               / %s"BYPASS"
               / %s"ERROR"
```

The semantics of cache-actions are:

- HIT_FRESH - The cache used a fresh stored response to satisfy the request without going forward

- HIT_STALE - The cache used a stale stored response to satisfy the request without going forward

- HIT_REFRESH_MODIFIED - The cache had a stale stored response, went forward to validate it, and used the new response to satisfy the request

- HIT_REFRESH_NOT_MODIFIED - The cache had a stale stored response, went forward to validate it, and used the stored response to satisfy the request

- HIT_REFRESH_STALE - The cache had a stale stored response, went forward to validate it, and encountered a problem, so the stored response was used to satisfy the request

- MISS - The cache did not have a stored response, so the request was forwarded

- MISS_CLIENT - The client included request directives (e.g., Pragma, Cache-Control) that prevented the cache from returning a response, so the request was forwarded

- BYPASS - The cache was configured to forward the request without attempting to use a stored response

- ERROR - The cache was unable to use a stored response or obtain one by going forward

Caches SHOULD use the most specific cache-action to a given response, but are not required to use all cache-actions. Future updates to this specification can add additional cache-actions.

Each member of the Cache header can also have any (or all, or none) of the following parameters:

```
node           = sh-string
fresh          = sh-integer
age            = sh-integer
cacheable      = sh-boolean
key            = sh-string
latency        = sh-integer
cl_nm          = sh-boolean
```

Their semantics are:

- "node" - a string identifying for the cache node. MAY be a hostname, IP address, or alias.

- "fresh" - an integer indicating the cache's estimation of the freshness lifetime ([RFC7234], Section 4.2.1) of this response in seconds, including any locally applied configuration. MAY be negative.

- "age" - an integer indicating the cache's estimation of the age

([RFC7234], Section 4.2.3) of this response in seconds. MUST be 0 or greater.

- "cacheable" - a boolean indicating whether the cache can store this response, according to [RFC7234], Section 3 and any locally applied configuration.

- "key" - a string representing the key that the cache has associated with this response. This might include the request URL, request headers, and other values.

- "latency" - an integer indicating the amount of time in milliseconds between the receipt of a complete set of request headers and sending the complete set of response headers of this response, from the viewpoint of the cache. Note that this may not include buffering time in transport protocols and similar delays.

- "cl_nm" - a boolean indicating whether the response to the client had a 304 Not Modified status code.

While all of these parameters are OPTIONAL, caches are encouraged to use the 'node' parameter to identify themselves.

3. Security Considerations

Information about a cache's content can be used to infer the activity of those using it. Generally, access to sensitive information in a cache is limited to those who are authorised to access that information (using a variety of techniques), so this does not represent an attack vector in the general sense.

However, if the Cache header is exposed to parties who are not authorised to obtain the response it occurs within, it could expose information about that data.

For example, if an attacker were able to obtain the Cache header from a response containing sensitive information and access were limited to one person (or limited set of people), they could determine

whether that information had been accessed before. This is similar to the information exposed by various timing attacks, but is arguably more reliable, since the cache is directly reporting its state.

Mitigations include use of encryption (e.g., TLS [RFC8446])) to protect the response, and careful controls over access to response headers (as are present in the Web platform). When in doubt, the Cache header field can be omitted.

4. References

4.1. Normative References

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Headers for HTTP", draft-ietf-httpbis-header-structure-09 (work in progress), December 2018.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <https://www.rfc-editor.org/info/rfc7405>.

[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>.

4.2. Informative References

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

4.3. URIs

[1] https://lists.w3.org/Archives/Public/ietf-http-wg/

[2] https://httpwg.org/

[3] https://github.com/httpwg/http-extensions/labels/cache-header

Author's Address

Mark Nottingham
Fastly

Email: mnot@mnot.net
URI: https://www.mnot.net/