HTTP
Internet-Draft
Intended status: Standards Track
Expires: 18 February 2022

M. Nottingham
Fastly

17 August 2021

The Cache-Status HTTP Response Header Field
draft-ietf-httpbis-cache-header-10

Abstract

To aid debugging, HTTP caches often append header fields to a response explaining how they handled the request in an ad hoc manner. This specification defines a standard mechanism to do so that is aligned with HTTP's 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/ (https://lists.w3.org/Archives/Public/ietf-http-wg/).

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

Status of This Memo

Table of Contents

1. Introduction

To aid debugging (both by humans and automated tools), HTTP caches often append header fields to a response explaining how they handled the request. Unfortunately, the semantics of these headers are often unclear, and both the semantics and syntax used vary between implementations.

This specification defines a new HTTP response header field, "Cache-Status" for this purpose, with standardized syntax and semantics.

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], with rules prefixed with "sf-" and the "key" rule as defined in [STRUCTURED-FIELDS]. It uses terminology from [HTTP] and [HTTP-CACHING].

2. The Cache-Status HTTP Response Header Field

The Cache-Status HTTP response header field indicates how caches have handled that response and its corresponding request. The syntax of this header field conforms to [STRUCTURED-FIELDS].

Its value is a List ([STRUCTURED-FIELDS], Section 3.1):

```
Cache-Status   = sf-list
```

Each member of the 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 (possibly including the user agent's cache itself, if it appends a value).

Caches determine when it is appropriate to add the Cache-Status header field to a response. Some might add it to all responses, whereas others might only do so when specifically configured to, or when the request contains a header field that activates a debugging mode. See Section 6 for related security considerations.

An intermediary SHOULD NOT append a Cache-Status member to responses that it generates locally, even if that intermediary contains a cache, unless the generated response is based upon a stored response (e.g., 304 Not Modified and 206 Partial Content are both based upon a stored response). For example, a proxy generating a 400 response due to a malformed request will not add a Cache-Status value, because that response was generated by the proxy, not the origin server.

When adding a value to the Cache-Status header field, caches SHOULD preserve the existing field value, to allow debugging of the entire chain of caches handling the request.

Each list member identifies the cache that inserted it and this identifier MUST be a String or Token. Depending on the deployment, this might be a product or service name (e.g., ExampleCache or "Example CDN"), a hostname ("cache-3.example.com"), an IP address, or a generated string.

Each member of the list can have parameters that describe that cache's handling of the request. While these parameters are OPTIONAL, caches are encouraged to provide as much information as possible.

This specification defines the following parameters:

```
hit          = sf-boolean
fwd          = sf-token
fwd-status   = sf-integer
ttl          = sf-integer
stored       = sf-boolean
collapsed    = sf-boolean
key          = sf-string
detail       = sf-token / sf-string
```

2.1. The hit parameter

"hit", when true, indicates that the request was satisfied by the cache; i.e., it was not forwarded, and the response was obtained from the cache.

A response that was originally produced by the origin but was modified by the cache (for example, a 304 or 206 status code) is still considered a hit, as long as it did not go forward (e.g., for validation).

A response that was in cache but not able to be used without going forward (e.g., because it was stale, or partial) is not considered a hit. Note that a stale response that is used without going forward (e.g., because the origin server is not available) can be considered a hit.

"hit" and "fwd" are exclusive; only one of them should appear on each list member.

2.2. The fwd parameter

"fwd" indicates that the request went forward towards the origin, and why.

The following parameter values are defined to explain why the request went forward, from most specific to least:

* bypass - The cache was configured to not handle this request

* method - The request method's semantics require the request to be forwarded

* uri-miss - The cache did not contain any responses that matched the request URI

* vary-miss - The cache contained a response that matched the request URI, but could not select a response based upon this request's headers and stored Vary headers.

* miss - The cache did not contain any responses that could be used to satisfy this request (to be used when an implementation cannot distinguish between uri-miss and vary-miss)

* request - The cache was able to select a fresh response for the request, but the request's semantics (e.g., Cache-Control request directives) did not allow its use

* stale - The cache was able to select a response for the request, but it was stale

* partial - The cache was able to select a partial response for the request, but it did not contain all of the requested ranges (or the request was for the complete response)

The most specific reason that the cache is aware of SHOULD be used, to the extent that it is possible to implement. See also [HTTP-CACHING], Section 4.

2.3. The fwd-status parameter

"fwd-status" indicates what status code (see [HTTP], Section 15) the next hop server returned in response to the forwarded request. Only meaningful when "fwd" is present; if "fwd-status" is not present but "fwd" is, it defaults to the status code sent in the response.

This parameter is useful to distinguish cases when the next hop server sends a 304 Not Modified response to a conditional request, or a 206 Partial Response because of a range request.

2.4. The ttl parameter

"ttl" indicates the response's remaining freshness lifetime (see [HTTP-CACHING], Section 4.2.1) as calculated by the cache, as an integer number of seconds, measured as closely as possible to when the response header section is sent by the cache. This includes freshness assigned by the cache; e.g., through heuristics (see [HTTP-CACHING], Section 4.2.2), local configuration, or other factors. May be negative, to indicate staleness.

2.5. The stored parameter

"stored" indicates whether the cache stored the response (see [HTTP-CACHING], Section 3); a true value indicates that it did. Only meaningful when fwd is present.

2.6. The collapsed parameter

"collapsed" indicates whether this request was collapsed together with one or more other forward requests (see [HTTP-CACHING], Section 4); if true, the response was successfully reused; if not, a new request had to be made. If not present, the request was not collapsed with others. Only meaningful when fwd is present.

2.7. The key parameter

"key" conveys a representation of the cache key (see [HTTP-CACHING], Section 2) used for the response. Note that this may be implementation-specific.

2.8. The detail parameter

"detail" allows implementations to convey additional information not captured in other parameters; for example, implementation-specific states, or other caching-related metrics.

For example:

```
Cache-Status: ExampleCache; hit; detail=MEMORY
```

The semantics of a detail parameter are always specific to the cache that sent it; even if a member of details from another cache shares the same name, it might not mean the same thing.

This parameter is intentionally limited. If an implementation's developer or operator needs to convey additional information in an interoperable fashion, they are encouraged to register extension parameters (see Section 4) or define another header field.

3. Examples

The most minimal cache hit:

```
Cache-Status: ExampleCache; hit
```

... but a polite cache will give some more information, e.g.:

```
Cache-Status: ExampleCache; hit; ttl=376
```

A stale hit just has negative freshness:

```
Cache-Status: ExampleCache; hit; ttl=-412
```

Whereas a complete miss is:

```
Cache-Status: ExampleCache; fwd=uri-miss
```

A miss that successfully validated on the back-end server:

```
Cache-Status: ExampleCache; fwd=stale; fwd-status=304
```

A miss that was collapsed with another request:

```
Cache-Status: ExampleCache; fwd=uri-miss; collapsed
```

A miss that the cache attempted to collapse, but couldn't:

```
Cache-Status: ExampleCache; fwd=uri-miss; collapsed=?0
```

Going through two separate layers of caching, where the cache closest to the origin responded to an earlier request with a stored response, and a second cache stored that response and later reused it to satisfy the current request:

```
Cache-Status: OriginCache; hit; ttl=1100,
              "CDN Company Here"; hit; ttl=545
```

Going through a three-layer caching system, where the closest to the origin is a reverse proxy (where the response was served from cache), the next is a forward proxy interposed by the network (where the request was forwarded because there wasn't any response cached with its URI, the request was collapsed with others, and the resulting response was stored), and the closest to the user is a browser cache (where there wasn't any response cached with the request's URI):

```
Cache-Status: ReverseProxyCache; hit
Cache-Status: ForwardProxyCache; fwd=uri-miss; collapsed; stored
Cache-Status: BrowserCache; fwd=uri-miss
```

4. Defining New Cache-Status Parameters

New Cache-Status Parameters can be defined by registering them in the HTTP Cache-Status Parameters registry.

Registration requests are reviewed and approved by a Designated Expert, as per [RFC8126], Section 4.5. A specification document is appreciated, but not required.

The Expert(s) should consider the following factors when evaluating requests:

* Community feedback
* If the value is sufficiently well-defined
* Generic parameters are preferred over vendor-specific, application-specific, or deployment-specific values. If a generic value cannot be agreed upon in the community, the parameter's name should be correspondingly specific (e.g., with a prefix that identifies the vendor, application or deployment).

Registration requests should use the following template:

* Name: [a name for the Cache-Status Parameter that matches the 'key' ABNF rule]
* Description: [a description of the parameter semantics and value]
* Reference: [to a specification defining this parameter, if available]

See the registry at https://iana.org/assignments/http-cache-status (https://iana.org/assignments/http-cache-status) for details on where to send registration requests.

5. IANA Considerations

Upon publication, please create the HTTP Cache-Status Parameters registry at https://iana.org/assignments/http-cache-status (https://iana.org/assignments/http-cache-status) and populate it with the types defined in Section 2; see Section 4 for its associated procedures.

Also, please create the following entry in the Hypertext Transfer Protocol (HTTP) Field Name Registry defined in [HTTP], Section 18.4:

* Field name: Cache-Status
* Status: permanent
* Specification document: [this document]
* Comments:

6. Security Considerations

Attackers can use the information in Cache-Status to probe the behaviour of the cache (and other components), and infer the activity of those using the cache. The Cache-Status header field may not create these risks on its own, but can assist attackers in exploiting them.

For example, knowing if a cache has stored a response can help an attacker execute a timing attack on sensitive data.

Additionally, exposing the cache key can help an attacker understand modifications to the cache key, which may assist cache poisoning attacks. See [ENTANGLE] for details.

The underlying risks can be mitigated with a variety of techniques (e.g., use of encryption and authentication; avoiding the inclusion of attacker-controlled data in the cache key), depending on their exact nature. Note that merely obfuscating the key does not mitigate this risk.

To avoid assisting such attacks, the Cache-Status header field can be omitted, only sent when the client is authorized to receive it, or only send sensitive information (e.g., the key parameter) when the client is authorized.

7. References

7.1. 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/rfc/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[STRUCTURED-FIELDS] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

[HTTP] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-17, 25 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-17>.

[HTTP-CACHING] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Caching", Work in Progress, Internet-Draft, draft-ietf-httpbis-cache-17, 25 July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cache-17>.

[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/rfc/rfc8174>.

[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/rfc/rfc5234>.

7.2. Informative References

[ENTANGLE] Kettle, J., "Web Cache Entanglement: Novel Pathways to Poisoning", 2020, <https://i.blackhat.com/USA-20/Wednesday/us-20-Kettle-Web-Cache-Entanglement-Novel-Pathways-To-Poisoning-wp.pdf>.

Author's Address

Mark Nottingham
Fastly
Prahran VIC
Australia

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