Problem with HTTP/1.1 Warning header, and proposed fix

draft-ietf-http-warning-00.txt

STATUS OF THIS MEMO

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the HTTP working group at <http-wg@cuckoo.hpl.hp.com>. Discussions of the working group are archived at <URL:http://www.ics.uci.edu/pub/ietf/http/>. General discussions about HTTP and the applications which use HTTP should take place on the <www-talk@w3.org> mailing list.

ABSTRACT

The current HTTP/1.1 (RFC2068) specification introduces a new "Warning" header, meant to carry status information about a request that cannot or should not be carried by the response status code. The existing specification for the interaction between Warning and HTTP caches is faulty, in that it may allow incorrect results after cache validation operations. This document identifies two separate (but related) problems, and proposes revisions of the HTTP/1.1 specification to solve these problems.
1 Introduction

The HTTP/1.1 specification (RFC2068) [1] introduces a new response header named "Warning", described this way (RFC2068 section 14.45):

The Warning response-header field is used to carry additional information about the status of a response which may not be reflected by the response status code. This information is typically, though not exclusively, used to warn about a possible lack of semantic transparency from caching operations.

The specification also says:

A cache MUST NOT delete any Warning header that it received with a response. However, if a cache successfully validates a cache entry, it SHOULD remove any Warning headers previously attached to that entry except as specified for specific Warning codes. It MUST then add any Warning headers received in the validating response.

while RFC2068 section 13.5.3 states, regarding what a cache does when receiving a 304 response:

The end-to-end headers stored in the cache entry are used for the constructed response, except that any end-to-end headers provided in the 304 response MUST replace the corresponding headers from the cache entry.

Note that there is an implicit disagreement about whether existing
Warning headers are associated with a cache entry after it is revalidated.

Also, \textit{RFC2068} section 13.1.2 states:

\begin{quote}
 Warnings are always cachable, because they never weaken the transparency of a response. This means that warnings can be passed to HTTP/1.0 caches without danger; such caches will simply pass the warning along as an entity-header in the response.
\end{quote}

This statement turns out to be somewhat erroneous.

The (implicit) goal behind the design of the Warning mechanism was that when an HTTP/1.1 client receives a message with a Warning header, that header should accurately reflect the status of the message. However, since the issuance of \textit{RFC2068}, it has been pointed out that the current specification can lead to Warning headers being attached to messages that should not have them, because of the interaction between the Warning specification and other specifications for HTTP/1.1 caches.

Another goal of the Warning design is that when a Warning is properly attached to a response, the response should be delivered to any end-client with the Warning intact. This should be true even if the path includes one or more HTTP/1.0 proxies.

2 Description of the problems

The two known problems both involve a form of the following scenario.

Suppose that the end-client is connected to the origin server via a path with two or more caching proxies:

\begin{center}
\begin{tabular}{c c c}
Origin & HTTP/1.1 & HTTP/1.1 \\
Server & Proxy P1 & Proxy P2 & Client \\
\end{tabular}
\end{center}

Also suppose that proxy P1 has a cached response for resource D in its cache, but proxy P2 does not. Also suppose that proxy P1 is configured to provide stale responses in some situations (e.g., because of network bandwidth constraints).

Now the end-client requests resource D, via proxy P2. Proxy P2 does not have a cache entry for D, so it forwards the request to proxy P1. Proxy P1 does have a stale cache entry for D, but decides to provide the response without validating the cache entry (i.e., without sending a conditional GET to the origin server.) So proxy P1 sends a response containing its cache entry for D, and attaches

\begin{quote}
  Warning: 10 P1 "Response is stale"
\end{quote}
P2 receives this response, forwards it to the end-client, and also stores the response in its own cache. So far, there is no problem.

At a later time some client of proxy P2 (perhaps the same one as before) requests resource D again from that proxy. This time, both proxies decide to validate their cached responses (since both responses are stale). So proxy P2 forwards a conditional GET to proxy P1, which forwards a conditional GET to the origin server. Suppose that the origin server sends a 304 (Not Modified) response to proxy P1, which forwards it to proxy P2.

At this point, the "Response is stale" Warning stored in the cache entry at P2 is clearly not correct; the response has been successfully revalidated. The correct behavior in this case would be for P2 to remove the warning from its cache entry, and to ensure that it is not attached to the response it forwards to the end-client.

Given this scenario, there are two different situations, depending on whether P2 implements HTTP/1.1 or higher (i.e., it understands the Warning header), or HTTP/1.0 or lower (i.e., it does not understand the warning header).

2.1 Proxy P2 implements HTTP/1.1
The HTTP/1.1 specification already hints at a solution for the case where the proxy (or any other caching agent) in question actually implements the Warning header. RFC2068 section 14.45 says:

A cache MUST NOT delete any Warning header that it received with a response. However, if a cache successfully validates a cache entry, it SHOULD remove any Warning headers previously attached to that entry except as specified for specific Warning codes. It MUST then add any Warning headers received in the validating response.

Implicit in this is that there are two categories of Warnings:

1. Those that describe the freshness or revalidation status of the response, and so must be deleted after a successful revalidation.

2. Those that describe some aspect of the entity body or entity headers that is not rectified by a revalidation; for example, a lossy compression of the entity body. These Warnings cannot be deleted after a revalidation.

Also implicit here is that a cache can reliably tell the difference between Warnings that should be deleted upon revalidation, and Warnings that should be retained even with revalidation. The problem here is that the set of Warning codes might be expanded beyond the set listed in the HTTP/1.1 specification.
For example, if an HTTP/1.1 cache receives

Warning: 37 "P1" "My hovercraft is full of eels"

that cache would not be able to tell if the Warning should be deleted or retained after a revalidation of the response.

2.2 Proxy P2 implements HTTP/1.0

If the proxy implements HTTP/1.0 (or lower), then it does not understand the Warning header at all. Therefore, it will always retain the Warning header in its cache entry, even after revalidation, and will pass this on to its clients.

This is not a problem for HTTP/1.0 clients (which ignore Warning), but HTTP/1.1 clients of proxy P2 could receive spurious Warning headers for the indefinite future.

3 Proposed solutions

This section proposes independent solutions to the two problems described in section 2. These solutions, together, are consistent with the two goals stated for Warnings in section 1. (These goals would prevent the adoption of several other proposed solutions, such as changing the Expires header when forwarding a response to an HTTP/1.0 client, or deleting the Warning header in this case.)

3.1 Solution to the ambiguity of future Warning codes

Section 2.1 shows that HTTP/1.1 caches do not have unambiguous information about whether a Warning should be retained or deleted upon revalidation of a cached response.

This can be solved by adding explicit information to the Warning values, so that the retain/delete choice is encoded in a way that does not require a cache to understand the full set of Warning codes.

Several protocols, including FTP [2] and HTTP itself, use three-digit status codes where the first digit conveys information about the success or failure of an operation; this allows some extensibility of the set of status codes without breaking existing implementations.

The same general technique can be used for the Warning code:

- Expand the warning code from two digits to three digits

- Use the leading digit to divide the Warnings into those that refer to the freshness (validation state) of a response, and must be deleted upon revalidation, and those that refer to other aspects of the response, and must be retained after revalidation.
- Redefine the existing Warning codes according to this scheme.

- Update the language in other parts of the specification to be consistent with the new scheme.

This approach allows the agent that initially attaches a Warning code, and so presumably knows the desired semantics, to explicitly indicate whether it should be retained or deleted after revalidation. It entirely removes the need for a recipient to make a non-trivial decision.

Section 3.3.3 provides a revised specification for Warning.

3.2 Solution to the HTTP/1.0 proxy problem

The solution to the problem described in section 2.2 is more difficult, because HTTP/1.0 caches cannot be expected to protect against incorrect caching of received Warning headers.

Since HTTP/1.0 agents ignore Warning entirely, the specific problem is to prevent an HTTP/1.1 recipient from interpreting a Warning header that has incorrectly been associated with a response. If such an incorrect association has been made, this is because an HTTP/1.0 cache first received a response with an appropriate Warning header, then revalidated the response but failed to delete the Warning header.

In other words, an HTTP/1.1 recipient of a Warning needs a way to determine whether it came from the most recent validation attempt for a cache entry, or whether it came with an earlier response.

Every HTTP response carries a Date header indicating when it was generated. It appears that when an HTTP/1.0 cache forwards a request and receives a 304 (Not Modified) response, the response that it forwards to its own client carries the (more recent) Date of the 304 response, not the Date that was associated with the original (cached) response.

---------

NOTE: We need to verify that this is indeed the way that HTTP/1.0 caches behave!

---------

If Warning headers were to carry an (optional) warning-date value, duplicating the value of the Date header associated with the response when the Warning was first attached to it, then an HTTP/1.1 recipient could compare this warning-date to the Date header in the received response. If these timestamps are the same, then (within the 1-second resolution of the Date header) the recipient can be sure that the Warning was not left over from an earlier response.
If the warning-date is earlier than the Date, however, then the
Warning had to have been created for an earlier response. If, in
this case, the Warning code indicates that the Warning should have
been deleted upon revalidation, then the recipient can safely delete
it.

---------
NOTE: The warning-date should never be later than the Date, if
I understand things correctly. If this ever happens, it
implies that some cache is not updating the Date of a cached
response with the Date of a subsequent 304 response. This
should never happen with HTTP/1.1 proxies, but it’s not clear
if this could ever happen with an HTTP/1.0 proxy.
---------

Therefore, a solution to the problem of HTTP/1.0 caches and Warnings
is for

- HTTP/1.1 proxies that create or forward a Warning to copy
  the Date value into an optional field of the Warning, if
  and only if the message is being sent to an HTTP/1.0 or
  lower recipient.

- HTTP/1.1 caches that receive an HTTP/1.0 response with a
  Warning to delete that Warning if its warning-date does not
  match the Date in the message, and if the Warning code
  indicates that it should be deleted upon revalidation.

Section 3.3.3 provides a revised specification for Warning, which
includes this optional warning-date field.

3.3 Editorial issues
This section describes the editorial changes to RFC2068 that are
required to implement both of the proposed modifications, and to
eliminate contradictory language regarding the Warning header.

3.3.1 RFC2068 Section 13.1.2: Warnings
RFC2068 section 13.1.2 states:

Warnings are always cachable, because they never weaken the
transparency of a response. This means that warnings can be
passed to HTTP/1.0 caches without danger; such caches will
simply pass the warning along as an entity-header in the
response.

Warnings are assigned numbers between 0 and 99. This
specification defines the code numbers and meanings of each
currently assigned warnings, allowing a client or cache to
take automated action in some (but not all) cases.

This passaged must be revised as follows:
Warnings come in two categories:

1. Those that describe the freshness or revalidation status of the response, and so MUST be deleted after a successful revalidation (see section 13.3 for a definition of revalidation).

2. Those that describe some aspect of the entity body or entity headers that is not rectified by a revalidation; for example, a lossy compression of the entity body. These Warnings MUST NOT be deleted after a successful revalidation.

Warnings are assigned 3-digit code numbers. The first digit indicates whether the Warning must or must not be deleted from a cached response after it is successfully revalidated. This specification defines the code numbers and meanings of each currently assigned warnings, allowing a client or cache to take automated action in some (but not all) cases.

HTTP/1.0 caches will cache all Warnings, without deleting the ones in the first category. Warnings that are passed to HTTP/1.0 caches carry an extra warning-date field, which prevents a future HTTP/1.1 recipient from believing an erroneously cached Warning.

3.3.2 RFC2068 Section 13.5.3 Combining Headers

RFC2068 section 13.5.3 states:

When a cache makes a validating request to a server, and the server provides a 304 (Not Modified) response, the cache must construct a response to send to the requesting client. The cache uses the entity-body stored in the cache entry as the entity-body of this outgoing response. The end-to-end headers stored in the cache entry are used for the constructed response, except that any end-to-end headers provided in the 304 response MUST replace the corresponding headers from the cache entry. Unless the cache decides to remove the cache entry, it MUST also replace the end-to-end headers stored with the cache entry with corresponding headers received in the incoming response.

In other words, the set of end-to-end headers received in the incoming response overrides all corresponding end-to-end headers stored with the cache entry. The cache may add Warning headers (see section 14.45) to this set.

This must be revised to say:

When a cache makes a validating request to a server, and the server provides a 304 (Not Modified) response, the cache must
construct a response to send to the requesting client. The cache uses the entity-body stored in the cache entry as the entity-body of this outgoing response. The end-to-end headers stored in the cache entry are used for the constructed response, except that

- any stored Warning headers with warn-code 1XX (see section 14.45) are deleted from the cache entry and the forwarded response.

- any stored Warning headers with warn-code 2XX are retained in the cache entry and the forwarded response.

- any end-to-end headers provided in the 304 response MUST replace the corresponding headers from the cache entry.

Unless the cache decides to remove the cache entry, it MUST also replace the end-to-end headers stored with the cache entry with corresponding headers received in the incoming response.

In other words, the set of end-to-end headers received in the incoming response overrides all corresponding end-to-end headers stored with the cache entry (except for stored Warning headers with warn-code 1XX, which are deleted even if not overridden).

3.3.3 RFC2068 Section 14.45 Warning

The entire RFC2068 Section 14.45 (Warning) is replaced as follows:

The Warning response-header field is used to carry additional information about the status of a response which may not be reflected by the response status code. This information is typically, though not exclusively, used to warn about a possible lack of semantic transparency from caching operations.

Warning headers are sent with responses using:

Warning = "Warning" ":" 1#warning-value

warning-value = warn-code SP warn-agent SP warn-text
               [SP warn-date]

warn-code = 3DIGIT

warn-agent = ( host [ ":" port ] ) | pseudonym
            ; the name or pseudonym of the server adding
            ; the Warning header, for use in debugging

warn-text = quoted-string

warn-date = "<" HTTP-date ">

Mogul, Luotonen                                                   [Page 9]
NOTE: The warn-date syntax requires the quotes because of the comma in the HTTP-date syntax. The comma serves to separate the wkday (day of week) token from the rest of the date, and since the wkday is completely redundant, we could save seven bytes (including the two commas) by defining a different date format for use here. We could also eliminate the "GMT" from this format, since it is mandatory for HTTP-Date.

A response may carry more than one Warning header.

The warn-text should be in a natural language and character set that is most likely to be intelligible to the human user receiving the response. This decision may be based on any available knowledge, such as the location of the cache or user, the Accept-Language field in a request, the Content-Language field in a response, etc. The default language is English and the default character set is ISO-8859-1.

If a character set other than ISO-8859-1 is used, it MUST be encoded in the warn-text using the method described in RFC 1522 [14]. Any server or cache may add Warning headers to a response. New Warning headers should be added after any existing Warning headers. A cache MUST NOT delete any Warning header that it received with a response. However, if a cache successfully validates a cache entry, it SHOULD remove any Warning headers previously attached to that entry except as specified for specific Warning codes. It MUST then add any Warning headers received in the validating response. In other words, Warning headers are those that would be attached to the most recent relevant response.

When multiple Warning headers are attached to a response, the user agent SHOULD display as many of them as possible, in the order that they appear in the response. If it is not possible to display all of the warnings, the user agent should follow these heuristics:

- Warnings that appear early in the response take priority over those appearing later in the response.
- Warnings in the user’s preferred character set take priority over warnings in other character sets but with identical warn-codes and warn-agents.

Systems that generate multiple Warning headers should order them with this user agent behavior in mind.

The warn-code consists of three digits. The first digit indicates whether the Warning MUST or MUST NOT be deleted from a stored cache entry after a successful revalidation:
1XX  Warnings that describe the freshness or revalidation status of the response, and so MUST be deleted after a successful revalidation.

2XX  Warnings that describe some aspect of the entity body or entity headers that is not rectified by a revalidation, and which MUST NOT be deleted after a successful revalidation.

This is a list of the currently-defined warn-codes, each with a recommended warn-text in English, and a description of its meaning.

110 Response is stale
   MUST be included whenever the returned response is stale.

111 Revalidation failed
   MUST be included if a cache returns a stale response because an attempt to revalidate the response failed, due to an inability to reach the server.

112 Disconnected operation
   SHOULD be included if the cache is intentionally disconnected from the rest of the network for a period of time.

113 Heuristic expiration
   MUST be included if the cache heuristically chose a freshness lifetime greater than 24 hours and the response’s age is greater than 24 hours.

199 Miscellaneous warning
   The warning text may include arbitrary information to be presented to a human user, or logged. A system receiving this warning MUST NOT take any automated action.

214 Transformation applied
   MUST be added by an intermediate cache or proxy if it applies any transformation changing the content-coding (as specified in the Content-Encoding header) or media-type (as specified in the Content-Type header) of the response, unless this Warning-Type header already appears in the response.

299 Miscellaneous persistent warning
   The warning text may include arbitrary information to be presented to a human user, or logged. A system receiving this warning MUST NOT take any automated action.
If an implementation sends a response with one or more Warning
headers to a client whose version is HTTP/1.0 or lower, then the
sender MUST include a warn-date in each warning-value.

If an implementation receives a response with a warning-value that
includes a warn-date, and that warn-date is different from the Date
value in the response, then that warning-value MUST be deleted from
the message before storing, forwarding, or using it. If all of the
warning-values are deleted for this reason, the Warning header MUST
be deleted as well.

4 Security Considerations

No known security implications beyond those listed in RFC2068.

5 Revision history

Minor clarifications, and grammar and spelling corrections, are not
listed here.

5.1 draft-ietf-http-warning-00.txt
   Initial draft.

6 Acknowledgements

We gratefully acknowledge the constructive comments received from Roy
Fielding, Koen Holtman, and Ben Laurie.

7 References

1. Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk
   Nielsen, and Tim Berners-Lee. Hypertext Transfer Protocol --


8 Authors’ addresses

Jeffrey C. Mogul
Western Research Laboratory
Digital Equipment Corporation
250 University Avenue
Palo Alto, California, 94305, U.S.A.
Email: mogul@wrl.dec.com
Phone: 1 415 617 3304 (email preferred)
Ari Luotonen
Netscape Communications Corp.
501 East Middlefield Road
Mountain View, CA 94043
Email: ari@netscape.com