TOC

HTTPbis Working Group	R. Fielding, Ed.
Internet-Draft	Day Software
Obsoletes: 2616 (if approved)	J. Gettys
Intended status: Standards Track	One Laptop per Child
Expires: April 29, 2010	J. Mogul
	HP
	H. Frystyk
	Microsoft
	L. Masinter
	Adobe Systems
	P. Leach
	Microsoft
	T. Berners-Lee
	W3C/MIT
	Y. Lafon, Ed.
	W3C
	M. Nottingham, Ed.
	J. Reschke, Ed.
	greenbytes
	October 26, 2009

HTTP/1.1, part 6: Caching
draft-ietf-httpbis-p6-cache-08

Status of this Memo

Copyright Notice

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This document is Part 6 of the seven-part specification that defines the protocol referred to as "HTTP/1.1" and, taken together, obsoletes RFC 2616. Part 6 defines requirements on HTTP caches and the associated header fields that control cache behavior or indicate cacheable response messages.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft should take place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org). The current issues list is at http://tools.ietf.org/wg/httpbis/trac/report/11 and related documents (including fancy diffs) can be found at http://tools.ietf.org/wg/httpbis/.

The changes in this draft are summarized in Appendix C.9 (Since draft-ietf-httpbis-p6-cache-07).

Table of Contents

1. Introduction

TOC

HTTP is typically used for distributed information systems, where performance can be improved by the use of response caches. This document defines aspects of HTTP/1.1 related to caching and reusing response messages.

1.1. Purpose

TOC

An HTTP cache is a local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server that is acting as a tunnel.

Caching would be useless if it did not significantly improve performance. The goal of caching in HTTP/1.1 is to reuse a prior response message to satisfy a current request. In some cases, a stored response can be reused without the need for a network request, reducing latency and network round-trips; a "freshness" mechanism is used for this purpose (see Section 2.3 (Freshness Model)). Even when a new request is required, it is often possible to reuse all or parts of the payload of a prior response to satisfy the request, thereby reducing network bandwidth usage; a "validation" mechanism is used for this purpose (see Section 2.4 (Validation Model)).

1.2. Terminology

TOC

This specification uses a number of terms to refer to the roles played by participants in, and objects of, HTTP caching.

cacheable

> A response is cacheable if a cache is allowed to store a copy of the response message for use in answering subsequent requests. Even when a response is cacheable, there may be additional constraints on whether a cache can use the cached copy to satisfy a particular request.

explicit expiration time

> The time at which the origin server intends that an entity should no longer be returned by a cache without further validation.

heuristic expiration time

> An expiration time assigned by a cache when no explicit expiration time is available.

age

> The age of a response is the time since it was sent by, or successfully validated with, the origin server.

first-hand

> A response is first-hand if the freshness model is not in use; i.e., its age is 0.

freshness lifetime

> The length of time between the generation of a response and its expiration time.

fresh

> A response is fresh if its age has not yet exceeded its freshness lifetime.

stale

> A response is stale if its age has passed its freshness lifetime (either explicit or heuristic).

validator

> A protocol element (e.g., an entity tag or a Last-Modified time) that is used to find out whether a stored response is an equivalent copy of an entity.

shared cache

> A cache that is accessible to more than one user. A non-shared cache is dedicated to a single user.

1.3. Requirements

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

1.4. Syntax Notation

TOC

This specification uses the ABNF syntax defined in Section 1.2 of [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.) (which extends the syntax defined in [RFC5234] (Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.) with a list rule). Appendix B (Collected ABNF) shows the collected ABNF, with the list rule expanded.

The following core rules are included by reference, as defined in [RFC5234] (Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.), Appendix B.1: ALPHA (letters), CR (carriage return), CRLF (CR LF), CTL (controls), DIGIT (decimal 0-9), DQUOTE (double quote), HEXDIG (hexadecimal 0-9/A-F/a-f), LF (line feed), OCTET (any 8-bit sequence of data), SP (space), VCHAR (any visible USASCII character), and WSP (whitespace).

1.4.1. Core Rules

TOC

The core rules below are defined in Section 1.2.2 of [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.):

```
quoted-string = <quoted-string, defined in [Part1], Section 1.2.2>
token         = <token, defined in [Part1], Section 1.2.2>
OWS           = <OWS, defined in [Part1], Section 1.2.2>
```

1.4.2. ABNF Rules defined in other Parts of the Specification

TOC

The ABNF rules below are defined in other parts:

```
field-name    = <field-name, defined in [Part1], Section 3.2>
HTTP-date     = <HTTP-date, defined in [Part1], Section 6.1>
port          = <port, defined in [Part1], Section 2.6>
pseudonym     = <pseudonym, defined in [Part1], Section 9.9>
uri-host      = <uri-host, defined in [Part1], Section 2.6>
```

2. Cache Operation

TOC

2.1. Response Cacheability

TOC

A cache MUST NOT store a response to any request, unless:

- The request method is defined as being cacheable, and

- the "no-store" cache directive (see Section 3.2 (Cache-Control)) does not appear in request or response headers, and

- the "private" cache response directive (see Section 3.2 (Cache-Control) does not appear in the response, if the cache is shared, and

- the "Authorization" header (see Section 3.1 of [Part7] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 7: Authentication,” October 2009.)) does not appear in the request, if the cache is shared (unless the "public" directive is present; see Section 3.2 (Cache-Control)), and

- the cache understands partial responses, if the response is partial or incomplete (see Section 2.1.1 (Storing Partial and Incomplete Responses)).

Note that in normal operation, most caches will not store a response that has neither a cache validator nor an explicit expiration time, as such responses are not usually useful to store. However, caches are not prohibited from storing such responses.

2.1.1. Storing Partial and Incomplete Responses

TOC

A cache that receives an incomplete response (for example, with fewer bytes of data than specified in a Content-Length header) can store the response, but MUST treat it as a partial response [Part5] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 5: Range Requests and Partial Responses,” October 2009.). Partial responses can be combined as described in Section 4 of [Part5] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 5: Range Requests and Partial Responses,” October 2009.); the result might be a full response or might still be partial. A cache MUST NOT return a partial response to a client without explicitly marking it as such using the 206 (Partial Content) status code.
A cache that does not support the Range and Content-Range headers MUST NOT store incomplete or partial responses.

2.2. Constructing Responses from Caches

TOC

For a presented request, a cache MUST NOT return a stored response, unless:

- The presented Request-URI and that of the stored response match ([TODO-Request-URI] (Need to find a new term for this, as Part 1 doesn't define Request-URI anymore; the new term request-target does not work for this. (see <http://tools.ietf.org/wg/httpbis/trac/ticket/196>))), and

- the request method associated with the stored response allows it to be used for the presented request, and

- selecting request-headers nominated by the stored response (if any) match those presented (see Section 2.6 (Caching Negotiated Responses)), and

- the presented request and stored response are free from directives that would prevent its use (see Section 3.2 (Cache-Control) and Section 3.4 (Pragma)), and
- the stored response is either:
  - fresh (see Section 2.3 (Freshness Model)), or
  - allowed to be served stale (see Section 2.3.3 (Serving Stale Responses)), or
  - successfully validated (see Section 2.4 (Validation Model)).

[TODO-method-cacheability] (define method cacheability for GET, HEAD and POST in p2-semantics.)

When a stored response is used to satisfy a request, caches MUST include a single Age header field (Section 3.1 (Age)) in the response with a value equal to the stored response's current_age; see Section 2.3.2 (Calculating Age). [anchor1] (DISCUSS: this currently includes successfully validated responses.)

Requests with methods that are unsafe (Section 7.1.1 of [Part2] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 2: Message Semantics,” October 2009.)) MUST be written through the cache to the origin server; i.e., A cache must not reply to such a request before having forwarded the request and having received a corresponding response.

Also, note that unsafe requests might invalidate already stored responses; see Section 2.5 (Request Methods that Invalidate).

Caches MUST use the most recent response (as determined by the Date header) when more than one suitable response is stored. They can also forward a request with "Cache-Control: max-age=0" or "Cache-Control: no-cache" to disambiguate which response to use.

[TODO-header-properties] (end-to-end and hop-by-hop headers, non-modifiable headers removed; re-spec in p1)

2.3. Freshness Model

TOC

When a response is "fresh" in the cache, it can be used to satisfy subsequent requests without contacting the origin server, thereby improving efficiency.

The primary mechanism for determining freshness is for an origin server to provide an explicit expiration time in the future, using either the Expires header (Section 3.3 (Expires)) or the max-age response cache directive (Section 3.2.2 (Response Cache-Control Directives)).

Generally, origin servers will assign future explicit expiration times to responses in the belief that the entity is not likely to change in a semantically significant way before the expiration time is reached.

If an origin server wishes to force a cache to validate every request, it can assign an explicit expiration time in the past. This means that the response is always stale, so that caches should validate it before using it for subsequent requests. [anchor2] (This wording may cause confusion, because the response may still be served stale.)

Since origin servers do not always provide explicit expiration times, HTTP caches may also assign heuristic expiration times when they are not specified, employing algorithms that use other header values (such as the Last-Modified time) to estimate a plausible expiration time. The HTTP/1.1 specification does not provide specific algorithms, but does impose worst-case constraints on their results.

The calculation to determine if a response is fresh is:

```
        response_is_fresh = (freshness_lifetime > current_age)
```

The freshness_lifetime is defined in Section 2.3.1 (Calculating Freshness Lifetime); the current_age is defined in Section 2.3.2 (Calculating Age).

Additionally, clients may need to influence freshness calculation. They can do this using several request cache directives, with the effect of either increasing or loosening constraints on freshness. See Section 3.2.1 (Request Cache-Control Directives).

[anchor3] (ISSUE: there are not requirements directly applying to cache-request-directives and freshness.)

Note that freshness applies only to cache operation; it cannot be used to force a user agent to refresh its display or reload a resource. See Section 4 (History Lists) for an explanation of the difference between caches and history mechanisms.

2.3.1. Calculating Freshness Lifetime

TOC

A cache can calculate the freshness lifetime (denoted as freshness_lifetime) of a response by using the first match of:

- If the cache is shared and the s-maxage response cache directive (Section 3.2.2 (Response Cache-Control Directives)) is present, use its value, or

- If the max-age response cache directive (Section 3.2.2 (Response Cache-Control Directives)) is present, use its value, or

- If the Expires response header (Section 3.3 (Expires)) is present, use its value minus the value of the Date response header, or

- Otherwise, no explicit expiration time is present in the response, but a heuristic may be used; see Section 2.3.1.1 (Calculating Heuristic Freshness).

Note that this calculation is not vulnerable to clock skew, since all of the information comes from the origin server.

2.3.1.1. Calculating Heuristic Freshness

TOC

If no explicit expiration time is present in a stored response that has a status code of 200, 203, 206, 300, 301 or 410, a heuristic expiration time can be calculated. Heuristics MUST NOT be used for other response status codes.

When a heuristic is used to calculate freshness lifetime, the cache SHOULD attach a Warning header with a 113 warn-code to the response if its current_age is more than 24 hours and such a warning is not already present.

Also, if the response has a Last-Modified header (Section 6.6 of [Part4] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 4: Conditional Requests,” October 2009.)), the heuristic expiration value SHOULD be no more than some fraction of the interval since that time. A typical setting of this fraction might be 10%.

[anchor4] (REVIEW: took away HTTP/1.0 query string heuristic uncacheability.)

2.3.2. Calculating Age

TOC

HTTP/1.1 uses the Age response-header to convey the estimated age of the response message when obtained from a cache. The Age field value is the cache's estimate of the amount of time since the response was generated or validated by the origin server. In essence, the Age value is the sum of the time that the response has been resident in each of the caches along the path from the origin server, plus the amount of time it has been in transit along network paths.

The term "age_value" denotes the value of the Age header, in a form appropriate for arithmetic operations.

HTTP/1.1 requires origin servers to send a Date header, if possible, with every response, giving the time at which the response was generated (see Section 9.3 of [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.)). The term "date_value" denotes the value of the Date header, in a form appropriate for arithmetic operations.

The term "now" means "the current value of the clock at the host performing the calculation." Hosts that use HTTP, but especially hosts running origin servers and caches, SHOULD use NTP [RFC1305] (Mills, D., “Network Time Protocol (Version 3) Specification, Implementation,” March 1992.) or some similar protocol to synchronize their clocks to a globally accurate time standard.

A response's age can be calculated in two entirely independent ways:

1. now minus date_value, if the local clock is reasonably well synchronized to the origin server's clock. If the result is negative, the result is replaced by zero.

2. age_value, if all of the caches along the response path implement HTTP/1.1.

These are combined as

```
         corrected_received_age = max(now - date_value, age_value)
```

When an Age value is received, it MUST be interpreted relative to the time the request was initiated, not the time that the response was received.

```
        corrected_initial_age = corrected_received_age
                              + (now - request_time)
```

where "request_time" is the time (according to the local clock) when the request that elicited this response was sent.

The current_age of a stored response can then be calculated by adding the amount of time (in seconds) since the stored response was last validated by the origin server to the corrected_initial_age.

In summary:

```
       age_value     - Age header field-value received with the response
       date_value    - Date header field-value received with the response
       request_time  - local time when the cache made the request
                      resulting in the stored response
       response_time - local time when the cache received the response
       now           - current local time

       apparent_age = max(0, response_time - date_value);
       corrected_received_age = max(apparent_age, age_value);
       response_delay = response_time - request_time;
       corrected_initial_age = corrected_received_age + response_delay;
       resident_time = now - response_time;
       current_age   = corrected_initial_age + resident_time;
```

2.3.3. Serving Stale Responses

TOC

A "stale" response is one that either has explicit expiry information, or is allowed to have heuristic expiry calculated, but is not fresh according to the calculations in Section 2.3 (Freshness Model).

Caches MUST NOT return a stale response if it is prohibited by an explicit in-protocol directive (e.g., by a "no-store" or "no-cache" cache directive, a "must-revalidate" cache-response-directive, or an applicable "s-maxage" or "proxy-revalidate" cache-response-directive; see Section 3.2.2 (Response Cache-Control Directives)).

Caches SHOULD NOT return stale responses unless they are disconnected (i.e., it cannot contact the origin server or otherwise find a forward path) or otherwise explicitly allowed (e.g., the max-stale request directive; see Section 3.2.1 (Request Cache-Control Directives)).

Stale responses SHOULD have a Warning header with the 110 warn-code (see Section 3.6 (Warning)). Likewise, the 112 warn-code SHOULD be sent on stale responses if the cache is disconnected.

If a cache receives a first-hand response (either an entire response, or a 304 (Not Modified) response) that it would normally forward to the requesting client, and the received response is no longer fresh, the cache SHOULD forward it to the requesting client without adding a new Warning (but without removing any existing Warning headers). A cache SHOULD NOT attempt to validate a response simply because that response became stale in transit.

2.4. Validation Model

TOC

When a cache has one or more stored responses for a requested URI, but cannot serve any of them (e.g., because they are not fresh, or one cannot be selected; see Section 2.6 (Caching Negotiated Responses)), it can use the conditional request mechanism [Part4] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 4: Conditional Requests,” October 2009.) in the forwarded request to give the origin server an opportunity to both select a valid stored response to be used, and to update it. This process is known as "validating" or "revalidating" the stored response.

When sending such a conditional request, the cache SHOULD add an If-Modified-Since header whose value is that of the Last-Modified header from the selected (see Section 2.6 (Caching Negotiated Responses)) stored response, if available.

Additionally, the cache SHOULD add an If-None-Match header whose value is that of the ETag header(s) from all responses stored for the requested URI, if present. However, if any of the stored responses contains only partial content, its entity-tag SHOULD NOT be included in the If-None-Match header field unless the request is for a range that would be fully satisfied by that stored response.

A 304 (Not Modified) response status code indicates that the stored response can be updated and reused; see Section 2.7 (Combining Responses).

A full response (i.e., one with a response body) indicates that none of the stored responses nominated in the conditional request is suitable. Instead, the full response is used both to satisfy the request and replace the stored response. [anchor5] (Should there be a requirement here?)

If a cache receives a 5xx response while attempting to validate a response, it MAY either forward this response to the requesting client, or act as if the server failed to respond. In the latter case, it MAY return a previously stored response (see Section 2.3.3 (Serving Stale Responses)).

2.5. Request Methods that Invalidate

TOC

Because unsafe methods (Section 7.1.1 of [Part2] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 2: Message Semantics,” October 2009.)) have the potential for changing state on the origin server, intervening caches can use them to keep their contents up-to-date.

The following HTTP methods MUST cause a cache to invalidate the Request-URI as well as the URI(s) in the Location and Content-Location headers (if present):

- PUT
- DELETE
- POST

An invalidation based on a URI from a Location or Content-Location header MUST NOT be performed if the host part of that URI differs from the host part in the Request-URI. This helps prevent denial of service attacks.

[anchor6] (TODO: "host part" needs to be specified better.)

A cache that passes through requests for methods it does not understand SHOULD invalidate the Request-URI.

Here, "invalidate" means that the cache will either remove all stored responses related to the Request-URI, or will mark these as "invalid" and in need of a mandatory validation before they can be returned in response to a subsequent request.

Note that this does not guarantee that all appropriate responses are invalidated. For example, the request that caused the change at the

origin server might not have gone through the cache where a response is stored.
[anchor7] (TODO: specify that only successful (2xx, 3xx?) responses invalidate.)

2.6. Caching Negotiated Responses

TOC

When a cache receives a request that can be satisfied by a stored response that has a Vary header field (Section 3.5 (Vary)), it MUST NOT use that response unless all of the selecting request-headers nominated by the Vary header match in both the original request (i.e., that associated with the stored response), and the presented request.
The selecting request-headers from two requests are defined to match if and only if the selecting request-headers in the first request can be transformed to the selecting request-headers in the second request by adding or removing linear white space [anchor8] ([ref]) at places where this is allowed by the corresponding ABNF, and/or combining multiple message-header fields with the same field name following the rules about header fields in Section 3.2 of [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.).
If a header field is absent from a request, it can only match another request if it is also absent there.
A Vary header field-value of "*" always fails to match, and subsequent requests to that resource can only be properly interpreted by the origin server.
The stored response with matching selecting request-headers is known as the selected response.
If no selected response is available, the cache MAY forward the presented request to the origin server in a conditional request; see Section 2.4 (Validation Model).

2.7. Combining Responses

TOC

When a cache receives a 304 (Not Modified) response or a 206 (Partial Content) response (in this section, the "new" response"), it needs to created an updated response by combining the stored response with the new one, so that the updated response can be used to satisfy the request.
If the new response contains an ETag, it identifies the stored response to use. [anchor9] (may need language about Content-Location here)[anchor10] (cover case where INM with multiple etags was sent)

If the status code is 206 (partial content), both the stored and new responses MUST have validators, and those validators MUST match using the strong comparison function (see Section 4 of [Part4] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 4: Conditional Requests,” October 2009.)). Otherwise, the responses MUST NOT be combined.

The stored response headers are used as those of the updated response, except that

- *any stored Warning headers with warn-code 1xx (see Section 3.6 (Warning)) MUST be deleted from the stored response and the updated response.
- *any stored Warning headers with warn-code 2xx MUST be retained in the stored response and the updated response.
- *any headers provided in the new response MUST replace the corresponding headers from the stored response.

If a header field-name in the new response matches more than one header in the stored response, all such stored headers MUST be replaced.

The updated response can [[anchor11] (requirement?)]] be used to replace the stored response in cache. In the case of a 206 response, the combined entity-body MAY be stored.

[anchor12] (ISSUE: discuss how to handle HEAD updates)

3. Header Field Definitions

TOC

This section defines the syntax and semantics of HTTP/1.1 header fields related to caching.

For entity-header fields, both sender and recipient refer to either the client or the server, depending on who sends and who receives the entity.

3.1. Age

TOC

The "Age" response-header field conveys the sender's estimate of the amount of time since the response was generated or successfully validated at the origin server. Age values are calculated as specified in Section 2.3.2 (Calculating Age).

```
       Age   = "Age" ":" OWS Age-v
       Age-v = delta-seconds
```

Age field-values are non-negative integers, representing time in seconds.

```
       delta-seconds  = 1*DIGIT
```

If a cache receives a value larger than the largest positive integer it can represent, or if any of its age calculations overflows, it MUST transmit an Age header with a field-value of 2147483648 (2^31). Caches SHOULD use an arithmetic type of at least 31 bits of range.
The presence of an Age header field in a response implies that a response is not first-hand. However, the converse is not true, since HTTP/1.0 caches may not implement the Age header field.

3.2. Cache-Control

TOC

The "Cache-Control" general-header field is used to specify directives that MUST be obeyed by all caches along the request/response chain. Such cache directives are unidirectional in that the presence of a directive in a request does not imply that the same directive is to be given in the response.

> Note that HTTP/1.0 caches might not implement Cache-Control and might only implement Pragma: no-cache (see Section 3.4 (Pragma)).

Cache directives MUST be passed through by a proxy or gateway application, regardless of their significance to that application, since the directives might be applicable to all recipients along the request/response chain. It is not possible to target a directive to a specific cache.

```
       Cache-Control   = "Cache-Control" ":" OWS Cache-Control-v
       Cache-Control-v = 1#cache-directive

       cache-directive = cache-request-directive
          / cache-response-directive

       cache-extension = token [ "=" ( token / quoted-string ) ]
```

3.2.1. Request Cache-Control Directives

TOC

```
cache-request-directive =
     "no-cache"
   / "no-store"
   / "max-age" "=" delta-seconds
   / "max-stale" [ "=" delta-seconds ]
   / "min-fresh" "=" delta-seconds
   / "no-transform"
   / "only-if-cached"
   / cache-extension
```

no-cache

> The no-cache request directive indicates that a stored response MUST NOT be used to satisfy the request without successful validation on the origin server.

no-store

> The no-store request directive indicates that a cache MUST NOT store any part of either this request or any response to it. This directive applies to both non-shared and shared caches. "MUST NOT store" in this context means that the cache MUST NOT intentionally store the information in non-volatile storage, and MUST make a best-effort attempt to remove the information from volatile storage as promptly as possible after forwarding it.
>
> This directive is NOT a reliable or sufficient mechanism for ensuring privacy. In particular, malicious or compromised caches might not recognize or obey this directive, and communications networks may be vulnerable to eavesdropping.

max-age

> The max-age request directive indicates that the client is willing to accept a response whose age is no greater than the specified time in seconds. Unless max-stale directive is also included, the client is not willing to accept a stale response.

max-stale

> The max-stale request directive indicates that the client is willing to accept a response that has exceeded its expiration time. If max-stale is assigned a value, then the client is willing to accept a response that has exceeded its expiration time by no more than the specified number of seconds. If no value is assigned to max-stale, then the client is willing to accept a stale response of any age. [anchor13] (mnot: of any staleness?)

min-fresh

> The min-fresh request directive indicates that the client is willing to accept a response whose freshness lifetime is no less than its current age plus the specified time in seconds. That is, the client wants a response that will still be fresh for at least the specified number of seconds.

no-transform

> The no-transform request directive indicates that an intermediate cache or proxy MUST NOT change the Content-Encoding, Content-Range or Content-Type request headers, nor the request entity-body.

only-if-cached

> The only-if-cached request directive indicates that the client only wishes to return a stored response. If it receives this directive, a cache SHOULD either respond using a stored response that is consistent with the other constraints of the request, or respond with a 504 (Gateway Timeout) status. If a group of caches is being operated as a unified system with good internal connectivity, such a request MAY be forwarded within that group of caches.

3.2.2. Response Cache-Control Directives

TOC

```
cache-response-directive =
     "public"
   / "private" [ "=" DQUOTE 1#field-name DQUOTE ]
   / "no-cache" [ "=" DQUOTE 1#field-name DQUOTE ]
   / "no-store"
   / "no-transform"
   / "must-revalidate"
   / "proxy-revalidate"
   / "max-age" "=" delta-seconds
   / "s-maxage" "=" delta-seconds
   / cache-extension
```

public

> The public response directive indicates that the response MAY be cached, even if it would normally be non-cacheable or cacheable only within a non-shared cache. (See also Authorization, Section 3.1 of [Part7] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 7: Authentication,” October 2009.), for additional details.)

private

> The private response directive indicates that the response message is intended for a single user and MUST NOT be stored by a shared cache. A private (non-shared) cache MAY store the response.
>
> If the private response directive specifies one or more field-names, this requirement is limited to the field-values associated with the listed response headers. That is, the specified field-names(s) MUST NOT be stored by a shared cache, whereas the remainder of the response message MAY be.
>
> Note: This usage of the word private only controls where the response may be stored, and cannot ensure the privacy of the message content. Also, private response directives with field-names are often handled by implementations as if an unqualified private directive was recieved; i.e., the special handling for the qualified form is not widely implemented.

no-cache

> The no-cache response directive indicates that the response MUST NOT be used to satisfy a subsequent request without successful validation on the origin server. This allows an origin server to prevent caching even by caches that have been configured to return stale responses.
>
> If the no-cache response directive specifies one or more field-names, this requirement is limited to the field-values associated with the listed response headers. That is, the specified field-name(s) MUST NOT be sent in the response to a subsequent request without successful validation on the origin server. This allows an origin server to prevent the re-use of certain header fields in a response, while still allowing caching of the rest of the response.
>
> Note: Most HTTP/1.0 caches will not recognize or obey this directive. Also, no-cache response directives with field-names are often handled by implementations as if an unqualified no-cache directive was recieved; i.e., the special handling for the qualified form is not widely implemented.

no-store

> The no-store response directive indicates that a cache MUST NOT store any part of either the immediate request or response. This directive applies to both non-shared and shared caches. "MUST NOT store" in this context means that the cache MUST NOT intentionally store the information in non-volatile storage, and MUST make a best-effort attempt to remove the information from volatile storage as promptly as possible after forwarding it.

> This directive is NOT a reliable or sufficient mechanism for ensuring privacy. In particular, malicious or compromised caches might not recognize or obey this directive, and communications networks may be vulnerable to eavesdropping.

must-revalidate

> The must-revalidate response directive indicates that once it has become stale, the response MUST NOT be used to satisfy subsequent requests without successful validation on the origin server.
>
> The must-revalidate directive is necessary to support reliable operation for certain protocol features. In all circumstances an HTTP/1.1 cache MUST obey the must-revalidate directive; in particular, if the cache cannot reach the origin server for any reason, it MUST generate a 504 (Gateway Timeout) response.
>
> Servers SHOULD send the must-revalidate directive if and only if failure to validate a request on the entity could result in incorrect operation, such as a silently unexecuted financial transaction.

proxy-revalidate

> The proxy-revalidate response directive has the same meaning as the must-revalidate response directive, except that it does not apply to non-shared caches.

max-age

> The max-age response directive indicates that response is to be considered stale after its age is greater than the specified number of seconds.

s-maxage

> The s-maxage response directive indicates that, in shared caches, the maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header. The s-maxage directive also implies the semantics of the proxy-revalidate response directive.

no-transform

> The no-transform response directive indicates that an intermediate cache or proxy MUST NOT change the Content-Encoding, Content-Range or Content-Type response headers, nor the response entity-body.

3.2.3. Cache Control Extensions

TOC

The Cache-Control header field can be extended through the use of one or more cache-extension tokens, each with an optional value. Informational extensions (those that do not require a change in cache behavior) can be added without changing the semantics of other directives. Behavioral extensions are designed to work by acting as modifiers to the existing base of cache directives. Both the new directive and the standard directive are supplied, such that applications that do not understand the new directive will default to the behavior specified by the standard directive, and those that understand the new directive will recognize it as modifying the requirements associated with the standard directive. In this way, extensions to the cache-control directives can be made without requiring changes to the base protocol.

This extension mechanism depends on an HTTP cache obeying all of the cache-control directives defined for its native HTTP-version, obeying certain extensions, and ignoring all directives that it does not understand.

For example, consider a hypothetical new response directive called "community" that acts as a modifier to the private directive. We define this new directive to mean that, in addition to any non-shared cache, any cache that is shared only by members of the community named within its value may cache the response. An origin server wishing to allow the UCI community to use an otherwise private response in their shared cache(s) could do so by including

```
  Cache-Control: private, community="UCI"
```

A cache seeing this header field will act correctly even if the cache does not understand the community cache-extension, since it will also see and understand the private directive and thus default to the safe behavior.

Unrecognized cache directives MUST be ignored; it is assumed that any cache directive likely to be unrecognized by an HTTP/1.1 cache will be combined with standard directives (or the response's default cacheability) such that the cache behavior will remain minimally correct even if the cache does not understand the extension(s).

3.3. Expires

TOC

The "Expires" entity-header field gives the date/time after which the response is considered stale. See Section 2.3 (Freshness Model) for further discussion of the freshness model.

The presence of an Expires field does not imply that the original resource will change or cease to exist at, before, or after that time.

The field-value is an absolute date and time as defined by HTTP-date in Section 6.1 of [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.); it MUST be sent in rfc1123-date format.

```
  Expires   = "Expires" ":" OWS Expires-v
  Expires-v = HTTP-date
```

For example

```
  Expires: Thu, 01 Dec 1994 16:00:00 GMT
```

> Note: if a response includes a Cache-Control field with the max-age directive (see Section 3.2.2 (Response Cache-Control Directives)), that directive overrides the Expires field. Likewise, the s-maxage directive overrides Expires in shared caches.

HTTP/1.1 servers SHOULD NOT send Expires dates more than one year in the future.
HTTP/1.1 clients and caches MUST treat other invalid date formats, especially including the value "0", as in the past (i.e., "already expired").

3.4. Pragma

TOC

The "Pragma" general-header field is used to include implementation-specific directives that might apply to any recipient along the request/response chain. All pragma directives specify optional behavior from the viewpoint of the protocol; however, some systems MAY require that behavior be consistent with the directives.

```
  Pragma            = "Pragma" ":" OWS Pragma-v
  Pragma-v          = 1#pragma-directive
  pragma-directive  = "no-cache" / extension-pragma
  extension-pragma  = token [ "=" ( token / quoted-string ) ]
```

When the no-cache directive is present in a request message, an application SHOULD forward the request toward the origin server even if it has a cached copy of what is being requested. This pragma directive has the same semantics as the no-cache response directive (see Section 3.2.2 (Response Cache-Control Directives)) and is defined here for backward compatibility with HTTP/1.0. Clients SHOULD include both header fields when a no-cache request is sent to a server not known to be HTTP/1.1 compliant. HTTP/1.1 caches SHOULD treat "Pragma: no-cache" as if the client had sent "Cache-Control: no-cache".

> Note: because the meaning of "Pragma: no-cache" as a response-header field is not actually specified, it does not provide a reliable replacement for "Cache-Control: no-cache" in a response.

This mechanism is deprecated; no new Pragma directives will be defined in HTTP.

3.5. Vary

TOC

The "Vary" response-header field conveys the set of request-header fields that were used to select the representation.
Caches use this information, in part, to determine whether a stored response can be used to satisdy a given request; see Section 2.6 (Caching Negotiated Responses). determines, while the response is fresh, whether a cache is permitted to use the response to reply to a subsequent request without validation; see Section 2.6 (Caching Negotiated Responses).
In uncacheable or stale responses, the Vary field value advises the user agent about the criteria that were used to select the representation.

```
       Vary   = "Vary" ":" OWS Vary-v
       Vary-v = "*" / 1#field-name
```

The set of header fields named by the Vary field value is known as the selecting request-headers.
Servers SHOULD include a Vary header field with any cacheable response that is subject to server-driven negotiation. Doing so allows a cache to properly interpret future requests on that resource and informs the user agent about the presence of negotiation on that resource. A server MAY include a Vary header field with a non-cacheable response that is subject to server-driven negotiation, since this might provide the user agent with useful information about the dimensions over which the response varies at the time of the response.
A Vary field value of "*" signals that unspecified parameters not limited to the request-headers (e.g., the network address of the client), play a role in the selection of the response representation; therefore, a cache cannot determine whether this response is appropriate. The "*" value MUST NOT be generated by a proxy server; it may only be generated by an origin server.
The field-names given are not limited to the set of standard request-header fields defined by this specification. Field names are case-insensitive.

TOC

3.6. Warning

The "Warning" general-header field is used to carry additional information about the status or transformation of a message that might not be reflected in the message. This information is typically used to warn about possible incorrectness introduced by caching operations or transformations applied to the entity body of the message.
Warnings can be used for other purposes, both cache-related and otherwise. The use of a warning, rather than an error status code, distinguish these responses from true failures.
Warning headers can in general be applied to any message, however some warn-codes are specific to caches and can only be applied to response messages.

```
       Warning    = "Warning" ":" OWS Warning-v
       Warning-v  = 1#warning-value

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

       warn-code  = 3DIGIT
       warn-agent = ( uri-host [ ":" port ] ) / pseudonym
                       ; the name or pseudonym of the server adding
                       ; the Warning header, for use in debugging
       warn-text  = quoted-string
       warn-date  = DQUOTE HTTP-date DQUOTE
```

Multiple warnings can be attached to a response (either by the origin server or by a cache), including multiple warnings with the same code number, only differing in warn-text.
When this occurs, the user agent SHOULD inform the user of as many of them as possible, in the order that they appear in the response.
Systems that generate multiple Warning headers SHOULD order them with this user agent behavior in mind. New Warning headers SHOULD be added after any existing Warning headers.
Warnings are assigned three digit warn-codes. The first digit indicates whether the Warning is required to be deleted from a stored response after validation:

- 1xx Warnings describe the freshness or validation status of the response, and so MUST be deleted by caches after validation. They can only be generated by a cache when validating a cached entry, and MUST NOT be generated in any other situation.

- 2xx Warnings describe some aspect of the entity body or entity headers that is not rectified by a validation (for example, a lossy compression of the entity bodies) and MUST NOT be deleted by caches after validation, unless a full response is returned, in which case they MUST be.

If an implementation sends a message with one or more Warning headers to a receiver whose version is HTTP/1.0 or lower, then the sender MUST include in each warning-value a warn-date that matches the Date header in the message.

If an implementation receives a message 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. (preventing the consequences of naive caching of Warning header fields.) If all of the warning-values are deleted for this reason, the Warning header MUST be deleted as well.

The following warn-codes are defined by this specification, each with a recommended warn-text in English, and a description of its meaning.

110 Response is stale

> SHOULD be included whenever the returned response is stale.

111 Revalidation failed

> SHOULD be included if a cache returns a stale response because an attempt to validate 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

> SHOULD 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 can include arbitrary information to be presented to a human user, or logged. A system receiving this warning MUST NOT take any automated action, besides presenting the warning to the user.

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, or the entity-body of the response, unless this Warning code already appears in the response.

299 Miscellaneous persistent warning

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

4. History Lists

TOC

User agents often have history mechanisms, such as "Back" buttons and history lists, that can be used to redisplay an entity retrieved earlier in a session.

History mechanisms and caches are different. In particular history mechanisms SHOULD NOT try to show a correct view of the current state of a resource. Rather, a history mechanism is meant to show exactly what the user saw at the time when the resource was retrieved.

By default, an expiration time does not apply to history mechanisms. If the entity is still in storage, a history mechanism SHOULD display it even if the entity has expired, unless the user has specifically configured the agent to refresh expired history documents.

This is not to be construed to prohibit the history mechanism from telling the user that a view might be stale.

> Note: if history list mechanisms unnecessarily prevent users from viewing stale resources, this will tend to force service authors to avoid using HTTP expiration controls and cache controls when they would otherwise like to. Service authors may consider it important that users not be presented with error messages or warning messages when they use navigation controls (such as BACK) to view previously fetched resources. Even though sometimes such resources ought not be cached, or ought to expire quickly, user interface considerations may force service authors to resort to other means of preventing caching (e.g. "once-only" URLs) in order not to suffer the effects of improperly functioning history mechanisms.

5. IANA Considerations

TOC

5.1. Message Header Registration

TOC

The Message Header Registry located at http://www.iana.org/assignments/message-headers/message-header-index.html should be updated with the permanent registrations below (see [RFC3864] (Klyne, G., Nottingham, M., and J. Mogul, “Registration Procedures for Message Header Fields,” September 2004.)):

Header Field Name	Protocol	Status	Reference
Age	http	standard	Section 3.1 (Age)
Cache-Control	http	standard	Section 3.2 (Cache-Control)
Expires	http	standard	Section 3.3 (Expires)
Pragma	http	standard	Section 3.4 (Pragma)
Vary	http	standard	Section 3.5 (Vary)
Warning	http	standard	Section 3.6 (Warning)

The change controller is: "IETF (iesg@ietf.org) - Internet Engineering Task Force".

6. Security Considerations

TOC

Caches expose additional potential vulnerabilities, since the contents of the cache represent an attractive target for malicious exploitation. Because cache contents persist after an HTTP request is complete, an attack on the cache can reveal information long after a user believes that the information has been removed from the network. Therefore, cache contents should be protected as sensitive information.

7. Acknowledgments

TOC

Much of the content and presentation of the caching design is due to suggestions and comments from individuals including: Shel Kaphan, Paul Leach, Koen Holtman, David Morris, and Larry Masinter.

8. References

TOC

8.1. Normative References

TOC

[Part1]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” draft-ietf-httpbis-p1-messaging-08 (work in progress), October 2009.
[Part2]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 2: Message Semantics,” draft-ietf-httpbis-p2-semantics-08 (work in progress), October 2009.
[Part3]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 3: Message Payload and Content Negotiation,” draft-ietf-httpbis-p3-payload-08 (work in progress), October 2009.
[Part4]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 4: Conditional Requests,” draft-ietf-httpbis-p4-conditional-08 (work in progress), October 2009.
[Part5]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 5: Range Requests and Partial Responses,” draft-ietf-httpbis-p5-range-08 (work in progress), October 2009.
[Part7]	Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 7: Authentication,” draft-ietf-httpbis-p7-auth-08 (work in progress), October 2009.
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997.
[RFC5234]	Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” STD 68, RFC 5234, January 2008.

8.2. Informative References

TOC

[RFC1305]	Mills, D., “Network Time Protocol (Version 3) Specification, Implementation,” RFC 1305, March 1992.
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” RFC 2616, June 1999.
[RFC3864]	

Klyne, G., Nottingham, M., and J. Mogul, “Registration Procedures for Message Header Fields,” BCP 90, RFC 3864, September 2004.

Appendix A. Compatibility with Previous Versions

TOC

A.1. Changes from RFC 2068

TOC

A case was missed in the Cache-Control model of HTTP/1.1; s-maxage was introduced to add this missing case. (Sections 2.1 (Response Cacheability), 3.2 (Cache-Control)).

Transfer-coding and message lengths all interact in ways that required fixing exactly when chunked encoding is used (to allow for transfer encoding that may not be self delimiting); it was important to straighten out exactly how message lengths are computed. (see also [Part1] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” October 2009.), [Part3] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 3: Message Payload and Content Negotiation,” October 2009.) and [Part5] (Fielding, R., Ed., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Ed., and J. Reschke, Ed., “HTTP/1.1, part 5: Range Requests and Partial Responses,” October 2009.)) [anchor16] (jre: This used to refer to the text about non-modifiable headers, and will have to be updated later on.)

Proxies should be able to add Content-Length when appropriate. [anchor17] (jre: This used to refer to the text about non-modifiable headers, and will have to be updated later on.)

Range request responses would become very verbose if all meta-data were always returned; by allowing the server to only send needed headers in a 206 response, this problem can be avoided. (Section 2.7 (Combining Responses))

The Cache-Control: max-age directive was not properly defined for responses. (Section 3.2.2 (Response Cache-Control Directives))

Warnings could be cached incorrectly, or not updated appropriately. (Section 2.3 (Freshness Model), 2.7 (Combining Responses), 3.2 (Cache-Control), and 3.6 (Warning)) Warning also needed to be a general header, as PUT or other methods may have need for it in requests.

A.2. Changes from RFC 2616

TOC

Remove requirement to consider Content-Location in successful responses in order to determine the appropriate response to use. (Section 2.4 (Validation Model))

Clarify denial of service attack avoidance requirement. (Section 2.5 (Request Methods that Invalidate))

Do not mention RFC 2047 encoding and multiple languages in Warning headers anymore, as these aspects never were implemented. (Section 3.6 (Warning))

TOC

Appendix B. Collected ABNF

```
Age = "Age:" OWS Age-v
Age-v = delta-seconds

Cache-Control = "Cache-Control:" OWS Cache-Control-v
Cache-Control-v = *( "," OWS ) cache-directive *( OWS "," [ OWS
 cache-directive ] )

Expires = "Expires:" OWS Expires-v
Expires-v = HTTP-date

HTTP-date = <HTTP-date, defined in [Part1], Section 6.1>

OWS = <OWS, defined in [Part1], Section 1.2.2>

Pragma = "Pragma:" OWS Pragma-v
Pragma-v = *( "," OWS ) pragma-directive *( OWS "," [ OWS
 pragma-directive ] )

Vary = "Vary:" OWS Vary-v
Vary-v = "*" / ( *( "," OWS ) field-name *( OWS "," [ OWS field-name
 ] ) )

Warning = "Warning:" OWS Warning-v
Warning-v = *( "," OWS ) warning-value *( OWS "," [ OWS warning-value
 ] )

cache-directive = cache-request-directive / cache-response-directive
cache-extension = token [ "=" ( token / quoted-string ) ]
cache-request-directive = "no-cache" / "no-store" / ( "max-age="
 delta-seconds ) / ( "max-stale" [ "=" delta-seconds ] ) / (
 "min-fresh=" delta-seconds ) / "no-transform" / "only-if-cached" /
 cache-extension
cache-response-directive = "public" / ( "private" [ "=" DQUOTE *( ","
 OWS ) field-name *( OWS "," [ OWS field-name ] ) DQUOTE ] ) / (
 "no-cache" [ "=" DQUOTE *( "," OWS ) field-name *( OWS "," [ OWS
 field-name ] ) DQUOTE ] ) / "no-store" / "no-transform" /
 "must-revalidate" / "proxy-revalidate" / ( "max-age=" delta-seconds
 ) / ( "s-maxage=" delta-seconds ) / cache-extension

delta-seconds = 1*DIGIT

extension-pragma = token [ "=" ( token / quoted-string ) ]

field-name = <field-name, defined in [Part1], Section 3.2>

port = <port, defined in [Part1], Section 2.6>
pragma-directive = "no-cache" / extension-pragma
pseudonym = <pseudonym, defined in [Part1], Section 9.9>
```

```
     quoted-string = <quoted-string, defined in [Part1], Section 1.2.2>

     token = <token, defined in [Part1], Section 1.2.2>

     uri-host = <uri-host, defined in [Part1], Section 2.6>

     warn-agent = ( uri-host [ ":" port ] ) / pseudonym
     warn-code = 3DIGIT
     warn-date = DQUOTE HTTP-date DQUOTE
     warn-text = quoted-string
     warning-value = warn-code SP warn-agent SP warn-text [ SP warn-date
      ]
```

ABNF diagnostics:

```
     ; Age defined but not used
     ; Cache-Control defined but not used
     ; Expires defined but not used
     ; Pragma defined but not used
     ; Vary defined but not used
     ; Warning defined but not used
```

Appendix C. Change Log (to be removed by RFC Editor before publication)

TOC

C.1. Since RFC2616

TOC

Extracted relevant partitions from [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.).

C.2. Since draft-ietf-httpbis-p6-cache-00

TOC

Closed issues:

- http://tools.ietf.org/wg/httpbis/trac/ticket/9: "Trailer" (http://purl.org/NET/http-errata#trailer-hop)

- http://tools.ietf.org/wg/httpbis/trac/ticket/12: "Invalidation after Update or Delete" (http://purl.org/NET/http-errata#invalidupd)

- http://tools.ietf.org/wg/httpbis/trac/ticket/35: "Normative and Informative references"

- http://tools.ietf.org/wg/httpbis/trac/ticket/48: "Date reference typo"

- http://tools.ietf.org/wg/httpbis/trac/ticket/49: "Connection header text"

- http://tools.ietf.org/wg/httpbis/trac/ticket/65: "Informative references"

- http://tools.ietf.org/wg/httpbis/trac/ticket/66: "ISO-8859-1 Reference"

- http://tools.ietf.org/wg/httpbis/trac/ticket/86: "Normative up-to-date references"

- http://tools.ietf.org/wg/httpbis/trac/ticket/87: "typo in 13.2.2"

Other changes:

- Use names of RFC4234 core rules DQUOTE and HTAB (work in progress on http://tools.ietf.org/wg/httpbis/trac/ticket/36)

C.3. Since draft-ietf-httpbis-p6-cache-01

TOC

Closed issues:

- http://tools.ietf.org/wg/httpbis/trac/ticket/82: "rel_path not used"

Other changes:

- Get rid of duplicate BNF rule names ("host" -> "uri-host") (work in progress on http://tools.ietf.org/wg/httpbis/trac/ticket/36)

- Add explicit references to BNF syntax and rules imported from other parts of the specification.

C.4. Since draft-ietf-httpbis-p6-cache-02

TOC

Ongoing work on IANA Message Header Registration (http://tools.ietf.org/wg/httpbis/trac/ticket/40):

*Reference RFC 3984, and update header registrations for headers defined in this document.

C.5. Since draft-ietf-httpbis-p6-cache-03

TOC

Closed issues:

*http://tools.ietf.org/wg/httpbis/trac/ticket/106: "Vary header classification"

C.6. Since draft-ietf-httpbis-p6-cache-04

TOC

Ongoing work on ABNF conversion (http://tools.ietf.org/wg/httpbis/trac/ticket/36):

*Use "/" instead of "|" for alternatives.

*Introduce new ABNF rules for "bad" whitespace ("BWS"), optional whitespace ("OWS") and required whitespace ("RWS").

*Rewrite ABNFs to spell out whitespace rules, factor out header value format definitions.

C.7. Since draft-ietf-httpbis-p6-cache-05

TOC

This is a total rewrite of this part of the specification.
Affected issues:

*http://tools.ietf.org/wg/httpbis/trac/ticket/54: "Definition of 1xx Warn-Codes"

*http://trac.tools.ietf.org/wg/httpbis/trac/ticket/60: "Placement of 13.5.1 and 13.5.2"

- http://trac.tools.ietf.org/wg/httpbis/trac/ticket/138: "The role of Warning and Semantic Transparency in Caching"

- http://trac.tools.ietf.org/wg/httpbis/trac/ticket/139: "Methods and Caching"

In addition: Final work on ABNF conversion (http://tools.ietf.org/wg/httpbis/trac/ticket/36):

- Add appendix containing collected and expanded ABNF, reorganize ABNF introduction.

C.8. Since draft-ietf-httpbis-p6-cache-06

TOC

Closed issues:

- http://tools.ietf.org/wg/httpbis/trac/ticket/161: "base for numeric protocol elements"

Affected issues:

- http://tools.ietf.org/wg/httpbis/trac/ticket/37: Vary and non-existant headers

C.9. Since draft-ietf-httpbis-p6-cache-07

TOC

Closed issues:

- http://tools.ietf.org/wg/httpbis/trac/ticket/54: "Definition of 1xx Warn-Codes"

- http://tools.ietf.org/wg/httpbis/trac/ticket/167: "Content-Location on 304 responses"

- http://tools.ietf.org/wg/httpbis/trac/ticket/169: "private and no-cache CC directives with headers"

- http://tools.ietf.org/wg/httpbis/trac/ticket/187: "RFC2047 and warn-text"

Index

TOC

Authors' Addresses

TOC

	Roy T. Fielding (editor)
	Day Software
	23 Corporate Plaza DR, Suite 280
	Newport Beach, CA 92660
	USA
Phone:	+1-949-706-5300
Fax:	+1-949-706-5305
Email:	fielding@gbiv.com
URI:	http://roy.gbiv.com/
	Jim Gettys
	One Laptop per Child
	21 Oak Knoll Road
	Carlisle, MA 01741
	USA
Email:	jg@laptop.org
URI:	http://www.laptop.org/
	Jeffrey C. Mogul
	Hewlett-Packard Company
	HP Labs, Large Scale Systems Group
	1501 Page Mill Road, MS 1177
	Palo Alto, CA 94304
	USA
Email:	JeffMogul@acm.org
	Henrik Frystyk Nielsen
	Microsoft Corporation
	1 Microsoft Way
	Redmond, WA 98052
	USA
Email:	henrikn@microsoft.com
	Larry Masinter
	Adobe Systems, Incorporated

	345 Park Ave
	San Jose, CA 95110
	USA
Email:	LMM@acm.org
URI:	http://larry.masinter.net/
	Paul J. Leach
	Microsoft Corporation
	1 Microsoft Way
	Redmond, WA 98052
Email:	paulle@microsoft.com
	Tim Berners-Lee
	World Wide Web Consortium
	MIT Computer Science and Artificial Intelligence Laboratory
	The Stata Center, Building 32
	32 Vassar Street
	Cambridge, MA 02139
	USA
Email:	timbl@w3.org
URI:	http://www.w3.org/People/Berners-Lee/
	Yves Lafon (editor)
	World Wide Web Consortium
	W3C / ERCIM
	2004, rte des Lucioles
	Sophia-Antipolis, AM 06902
	France
Email:	ylafon@w3.org
URI:	http://www.raubacapeu.net/people/yves/
	Mark Nottingham (editor)
Email:	mnot@mnot.net
URI:	http://www.mnot.net/
	Julian F. Reschke (editor)
	greenbytes GmbH
	Hafenweg 16
	Muenster, NW 48155
	Germany
Phone:	+49 251 2807760
Fax:	+49 251 2807761
Email:	julian.reschke@greenbytes.de
URI:	http://greenbytes.de/tech/webdav/