

HTTPbis Working Group  
Internet-Draft  
Obsoletes: [2616](#) (if approved)  
Intended status: Standards Track  
Expires: September 15, 2011

R. Fielding, Ed.  
Adobe  
J. Gettys  
Alcatel-Lucent  
J. Mogul  
HP  
H. Frystyk  
Microsoft  
L. Masinter  
Adobe  
P. Leach  
Microsoft  
T. Berners-Lee  
W3C/MIT  
Y. Lafon, Ed.  
W3C  
M. Nottingham, Ed.  
  
J. Reschke, Ed.  
greenbytes  
March 14, 2011

HTTP/1.1, part 6: Caching  
draft-ietf-httpbis-p6-cache-13

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](mailto:ietf-http-wg@w3.org)). The current issues list is at <http://tools.ietf.org/wg/httpbis/trac/report/3> 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.14](#).

Status of This Memo

Fielding, et al.

Expires September 15, 2011

[Page 1]

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 15, 2011.

## Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">5</a>
<a href="#">1.1.</a>	Purpose . . . . .	<a href="#">5</a>
<a href="#">1.2.</a>	Terminology . . . . .	<a href="#">5</a>
<a href="#">1.3.</a>	Requirements . . . . .	<a href="#">7</a>

1.4.	Syntax Notation	7
1.4.1.	Core Rules	7
1.4.2.	ABNF Rules defined in other Parts of the Specification	7
2.	Cache Operation	8
2.1.	Response Cacheability	8
2.1.1.	Storing Partial and Incomplete Responses	9
2.2.	Constructing Responses from Caches	9
2.3.	Freshness Model	10
2.3.1.	Calculating Freshness Lifetime	11
2.3.2.	Calculating Age	12
2.3.3.	Serving Stale Responses	13
2.4.	Validation Model	14
2.5.	Request Methods that Invalidate	15
2.6.	Shared Caching of Authenticated Responses	15
2.7.	Caching Negotiated Responses	16
2.8.	Combining Responses	17
3.	Header Field Definitions	17
3.1.	Age	18
3.2.	Cache-Control	18
3.2.1.	Request Cache-Control Directives	19
3.2.2.	Response Cache-Control Directives	21
3.2.3.	Cache Control Extensions	23
3.3.	Expires	24
3.4.	Pragma	25
3.5.	Vary	26
3.6.	Warning	26
4.	History Lists	29
5.	IANA Considerations	29
5.1.	Cache Directive Registry	29
5.2.	Header Field Registration	30
6.	Security Considerations	30
7.	Acknowledgments	31
8.	References	31
8.1.	Normative References	31
8.2.	Informative References	32
Appendix A.	Changes from <a href="#">RFC 2616</a>	32
Appendix B.	Collected ABNF	32
Appendix C.	Change Log (to be removed by RFC Editor before publication)	34
C.1.	Since <a href="#">RFC 2616</a>	34
C.2.	Since <a href="#">draft-ietf-httpbis-p6-cache-00</a>	34
C.3.	Since <a href="#">draft-ietf-httpbis-p6-cache-01</a>	35
C.4.	Since <a href="#">draft-ietf-httpbis-p6-cache-02</a>	35
C.5.	Since <a href="#">draft-ietf-httpbis-p6-cache-03</a>	35
C.6.	Since <a href="#">draft-ietf-httpbis-p6-cache-04</a>	35
C.7.	Since <a href="#">draft-ietf-httpbis-p6-cache-05</a>	36
C.8.	Since <a href="#">draft-ietf-httpbis-p6-cache-06</a>	36

<a href="#">C.9.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-07</a>	. . . . .	<a href="#">36</a>
<a href="#">C.10.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-08</a>	. . . . .	<a href="#">37</a>
<a href="#">C.11.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-09</a>	. . . . .	<a href="#">37</a>
<a href="#">C.12.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-10</a>	. . . . .	<a href="#">38</a>
<a href="#">C.13.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-11</a>	. . . . .	<a href="#">38</a>
<a href="#">C.14.</a>	Since <a href="#">draft-ietf-httpbis-p6-cache-12</a>	. . . . .	<a href="#">38</a>
Index	. . . . .	. . . . .	<a href="#">38</a>

## 1. Introduction

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

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 employ 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](#)). 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](#)).

### 1.2. Terminology

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

cache

A conformant implementation of a HTTP cache. Note that this implies an HTTP/1.1 cache; this specification does not define conformance for HTTP/1.0 caches.

shared cache

A cache that is accessible to more than one user; usually (but not always) deployed as part of an intermediary.

private cache

A cache that is dedicated to a single user.

## 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 might be additional constraints on whether a cache can use the stored copy to satisfy a particular request.

## explicit expiration time

The time at which the origin server intends that a representation 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 a representation.

### [1.3.](#) Requirements

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](#)].

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

This specification uses the ABNF syntax defined in Section 1.2 of [[Part1](#)] (which extends the syntax defined in [[RFC5234](#)] with a list rule). [Appendix B](#) shows the collected ABNF, with the list rule expanded.

The following core rules are included by reference, as defined in [[RFC5234](#)], [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

The core rules below are defined in Section 1.2.2 of [[Part1](#)]:

quoted-string	= <quoted-string, defined in [ <a href="#">Part1</a> ], Section 1.2.2>
token	= <token, defined in [ <a href="#">Part1</a> ], Section 1.2.2>
OWS	= <OWS, defined in [ <a href="#">Part1</a> ], Section 1.2.2>

#### [1.4.2.](#) ABNF Rules defined in other Parts of the Specification

The ABNF rules below are defined in other parts:

field-name	= <field-name, defined in [ <a href="#">Part1</a> ], Section 3.2>
HTTP-date	= <HTTP-date, defined in [ <a href="#">Part1</a> ], Section 6.1>
port	= <port, defined in [ <a href="#">Part1</a> ], Section 2.6>
pseudonym	= <pseudonym, defined in [ <a href="#">Part1</a> ], Section 9.9>
uri-host	= <uri-host, defined in [ <a href="#">Part1</a> ], Section 2.6>

## [2.](#) Cache Operation

### [2.1.](#) Response Cacheability

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

- o The request method is understood by the cache and defined as being cacheable, and
- o the response status code is understood by the cache, and
- o the "no-store" cache directive (see [Section 3.2](#)) does not appear in request or response header fields, and
- o the "private" cache response directive (see [Section 3.2.2](#)) does not appear in the response, if the cache is shared, and
- o the "Authorization" header field (see Section 4.1 of [\[Part7\]](#)) does not appear in the request, if the cache is shared, unless the response explicitly allows it (see [Section 2.6](#)), and
- o the response either:
  - \* contains an Expires header field (see [Section 3.3](#)), or
  - \* contains a max-age response cache directive (see [Section 3.2.2](#)), or
  - \* contains a s-maxage response cache directive and the cache is shared, or
  - \* contains a Cache Control Extension (see [Section 3.2.3](#)) that allows it to be cached, or
  - \* has a status code that can be served with heuristic freshness (see [Section 2.3.1.1](#)).

In this context, a cache has "understood" a request method or a response status code if it recognises it and implements any cache-specific behaviour. In particular, 206 Partial Content responses cannot be cached by an implementation that does not handle partial content (see [Section 2.1.1](#)).

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

A cache that receives an incomplete response (for example, with fewer bytes of data than specified in a Content-Length header field) can store the response, but MUST treat it as a partial response [[Part5](#)]. Partial responses can be combined as described in Section 4 of [[Part5](#)]; 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 header fields MUST NOT store incomplete or partial responses.

### [2.2.](#) Constructing Responses from Caches

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

- o The presented effective request URI (Section 4.3 of [[Part1](#)]) and that of the stored response match, and
- o the request method associated with the stored response allows it to be used for the presented request, and
- o selecting header fields nominated by the stored response (if any) match those presented (see [Section 2.7](#)), and
- o the presented request and stored response are free from directives that would prevent its use (see [Section 3.2](#) and [Section 3.4](#)), and
- o the stored response is either:
  - \* fresh (see [Section 2.3](#)), or
  - \* allowed to be served stale (see [Section 2.3.3](#)), or
  - \* successfully validated (see [Section 2.4](#)).

When a stored response is used to satisfy a request without validation, a cache MUST include a single Age header field ([Section 3.1](#)) in the response with a value equal to the stored response's current\_age; see [Section 2.3.2](#).

A cache MUST write through requests with methods that are unsafe (Section 7.1.1 of [[Part2](#)]) to the origin server; i.e., a cache must not generate a 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](#).

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

A cache that does not have a clock available MUST NOT use stored responses without revalidating them on every use. A cache, especially a shared cache, SHOULD use a mechanism, such as NTP [[RFC1305](#)], to synchronize its clock with a reliable external standard.

### [2.3](#). Freshness Model

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 field ([Section 3.3](#)) or the max-age response cache directive ([Section 3.2.2](#)). Generally, origin servers will assign future explicit expiration times to responses in the belief that the representation 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 to indicate that the response is already stale. Compliant caches will normally validate the cached response before reusing it for subsequent requests (see [Section 2.3.3](#)).

Since origin servers do not always provide explicit expiration times, a cache MAY assign a heuristic expiration time when an explicit time is not specified, employing algorithms that use other header field values (such as the Last-Modified time) to estimate a plausible expiration time. This 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:

$$\text{response\_is\_fresh} = (\text{freshness\_lifetime} > \text{current\_age})$$

The freshness\_lifetime is defined in [Section 2.3.1](#); the current\_age is defined in [Section 2.3.2](#).

Additionally, clients might 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](#).

[[ISSUE-no-req-for-directives: 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](#) for an explanation of the difference between caches and history mechanisms.

### [2.3.1](#). Calculating Freshness Lifetime

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

- o If the cache is shared and the `s-maxage` response cache directive ([Section 3.2.2](#)) is present, use its value, or
- o If the `max-age` response cache directive ([Section 3.2.2](#)) is present, use its value, or
- o If the `Expires` response header field ([Section 3.3](#)) is present, use its value minus the value of the `Date` response header field, or
- o Otherwise, no explicit expiration time is present in the response. A heuristic freshness lifetime might be applicable; see [Section 2.3.1.1](#).

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

If no explicit expiration time is present in a stored response that has a status code whose definition allows heuristic freshness to be used (including the following in Section 8 of [\[Part2\]](#): 200, 203, 206, 300, 301 and 410), a cache MAY calculate a heuristic expiration time. A cache MUST NOT use heuristics to determine freshness for responses with status codes that do not explicitly allow it.

When a heuristic is used to calculate freshness lifetime, a cache SHOULD attach a `Warning` header field 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 field (Section 6.6 of [Part4]), a cache SHOULD NOT use a heuristic expiration value that is more than some fraction of the interval since that time. A typical setting of this fraction might be 10%.

Note: [RFC 2616](#) ([RFC2616], Section 13.9) required that caches do not calculate heuristic freshness for URIs with query components (i.e., those containing '?'). In practice, this has not been widely implemented. Therefore, servers are encouraged to send explicit directives (e.g., Cache-Control: no-cache) if they wish to preclude caching.

### [2.3.2](#). Calculating Age

HTTP/1.1 uses the Age header field 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 following data is used for the age calculation:

age\_value

The term "age\_value" denotes the value of the Age header field ([Section 3.1](#)), in a form appropriate for arithmetic operation; or 0, if not available.

date\_value

HTTP/1.1 requires origin servers to send a Date header field, if possible, with every response, giving the time at which the response was generated. The term "date\_value" denotes the value of the Date header field, in a form appropriate for arithmetic operations. See Section 9.3 of [\[Part1\]](#) for the definition of the Date header field, and for requirements regarding responses without it.

now

The term "now" means "the current value of the clock at the host performing the calculation". A cache SHOULD use NTP ([\[RFC1305\]](#)) or some similar protocol to synchronize its clocks to a globally accurate time standard.

request\_time

The current value of the clock at the host at the time the request resulting in the stored response was made.

response\_time

The current value of the clock at the host at the time the response was received.

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

1. the "apparent\_age": response\_time 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. the "corrected\_age\_value", if all of the caches along the response path implement HTTP/1.1. A cache **MUST** interpret this value relative to the time the request was initiated, not the time that the response was received.

```
apparent_age = max(0, response_time - date_value);
```

```
response_delay = response_time - request_time;  
corrected_age_value = age_value + response_delay;
```

These are combined as

```
corrected_initial_age = max(apparent_age, corrected_age_value);
```

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.

```
resident_time = now - response_time;  
current_age = corrected_initial_age + resident_time;
```

### 2.3.3. Serving Stale Responses

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](#).

A cache **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](#)).

A cache SHOULD NOT return stale responses unless it is 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](#)).

A cache SHOULD append a Warning header field with the 110 warn-code (see [Section 3.6](#)) to stale responses. Likewise, a cache SHOULD add the 112 warn-code to 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 header fields). A cache SHOULD NOT attempt to validate a response simply because that response became stale in transit.

#### [2.4.](#) Validation Model

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 2.7](#)), it can use the conditional request mechanism [[Part4](#)] 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, a cache SHOULD add an If-Modified-Since header field whose value is that of the Last-Modified header field from the selected (see [Section 2.7](#)) stored response, if available.

Additionally, a cache SHOULD add an If-None-Match header field whose value is that of the ETag header field(s) from all responses stored for the requested URI, if present. However, if any of the stored responses contains only partial content, the cache SHOULD NOT include its entity-tag 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 2.8](#).

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, a cache SHOULD use the full response to satisfy

the request and MAY replace the stored response.

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](#)).

## [2.5](#). Request Methods that Invalidate

Because unsafe request methods (Section 7.1.1 of [\[Part2\]](#)) have the potential for changing state on the origin server, intervening caches can use them to keep their contents up-to-date.

A cache MUST invalidate the effective Request URI (Section 4.3 of [\[Part1\]](#)) as well as the URI(s) in the Location and Content-Location header fields (if present) when the following request methods are received:

- o PUT
- o DELETE
- o POST

However, a cache MUST NOT invalidate a URI from a Location or Content-Location header field if the host part of that URI differs from the host part in the effective request URI (Section 4.3 of [\[Part1\]](#)). This helps prevent denial of service attacks.

A cache that passes through requests with methods it does not understand SHOULD invalidate the effective request URI (Section 4.3 of [\[Part1\]](#)).

Here, "invalidate" means that the cache will either remove all stored responses related to the effective 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.

## [2.6](#). Shared Caching of Authenticated Responses

A shared cache MUST NOT use a cached response to a request with an Authorization header field (Section 4.1 of [\[Part7\]](#)) to satisfy any subsequent request unless a cache directive that allows such

responses to be stored is present in the response.

In this specification, the following Cache-Control response directives ([Section 3.2.2](#)) have such an effect: must-revalidate, public, s-maxage.

Note that cached responses that contain the "must-revalidate" and/or "s-maxage" response directives are not allowed to be served stale ([Section 2.3.3](#)) by shared caches. In particular, a response with either "max-age=0, must-revalidate" or "s-maxage=0" cannot be used to satisfy a subsequent request without revalidating it on the origin server.

## [2.7](#). Caching Negotiated Responses

When a cache receives a request that can be satisfied by a stored response that has a Vary header field ([Section 3.5](#)), it MUST NOT use that response unless all of the selecting header fields nominated by the Vary header field match in both the original request (i.e., that associated with the stored response), and the presented request.

The selecting header fields from two requests are defined to match if and only if those in the first request can be transformed to those in the second request by applying any of the following:

- o adding or removing whitespace, where allowed in the header field's syntax
- o combining multiple header fields with the same field name (see [Section 3.2](#) of [\[Part1\]](#))
- o normalizing both header field values in a way that is known to have identical semantics, according to the header field's specification (e.g., re-ordering field values when order is not significant; case-normalization, where values are defined to be case-insensitive)

If (after any normalization that might take place) 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 header fields 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](#).

### [2.8](#). Combining Responses

When a cache receives a 304 (Not Modified) response or a 206 (Partial Content) response (in this section, the "new" response"), it needs to create an updated response by combining the stored response with the new one, so that the updated response can be used to satisfy the request, and potentially update the cached response.

If the new response contains an ETag, it identifies the stored response to use. [[TODO-mention-CL: might need language about Content-Location here]][[TODO-select-for-combine: Shouldn't this be the selected response?]]

When the new response's status code is 206 (partial content), a cache MUST NOT combine it with the old response if either response does not have a validator, and MUST NOT combine it with the old response when those validators do not match with the strong comparison function (see Section 4 of [\[Part4\]](#)).

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

- o a cache MUST delete any stored Warning header fields with warn-code 1xx (see [Section 3.6](#)).
- o a cache MUST retain any stored Warning header fields with warn-code 2xx.
- o a cache MUST use other header fields provided in the new response to replace all instances of the corresponding header fields from the stored response.

A cache MUST use the updated response header fields to replace those of the stored response (unless the stored response is removed). In the case of a 206 response, a cache MAY store the combined representation.

## [3](#). Header Field Definitions

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

### 3.1. Age

The "Age" 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](#).

```
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 field with a field-value of 2147483648 ( $2^{31}$ ). Recipients parsing the Age header field-value 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 might not implement the Age header field.

### 3.2. Cache-Control

The "Cache-Control" header field is used to specify directives for 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.

A cache MUST obey the requirements of the Cache-Control directives defined in this section. See [Section 3.2.3](#) for information about how Cache-Control directives defined elsewhere are handled.

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

A proxy, whether or not it implements a cache, MUST pass cache directives through in forwarded messages, 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

```
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 cache MUST NOT use a stored response 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 private 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 might be vulnerable to eavesdropping.

Note that if a request containing this directive is satisfied from a cache, the no-store request directive does not apply to the already stored response.

#### 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 the max-stale request directive is also present, 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.

#### 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 intermediary (whether or not it implements a cache) **MUST NOT** change the Content-Encoding, Content-Range or Content-Type request header fields, nor the request representation.

#### 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 code. If a group of caches is being operated as a unified system with good internal connectivity, a member cache **MAY** forward such a request within that group of caches.

### 3.2.2. Response Cache-Control Directives

```
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 a response whose associated request contains an 'Authentication' header MAY be stored (see [Section 2.6](#)).

#### 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 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 header fields. That is, a shared cache MUST NOT store the specified field-names(s), whereas it MAY store the remainder of the response message.

Note: This usage of the word private only controls where the response can be stored; it 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 received; 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 a cache from using it to satisfy a request without contacting it, 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 header fields. That is, a cache **MUST NOT** send the specified field-name(s) 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 received; 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 private 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 might be vulnerable to eavesdropping.

#### must-revalidate

The must-revalidate response directive indicates that once it has become stale, a cache **MUST NOT** use the response 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 a cache **MUST** obey the must-revalidate directive; in particular, if a cache cannot reach the origin server for any reason, it **MUST** generate a 504 (Gateway Timeout) response.

A server **SHOULD** send the must-revalidate directive if and only if failure to validate a request on the representation 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 private 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 field. The s-maxage directive also implies the semantics of the proxy-revalidate response directive.

#### no-transform

The no-transform response directive indicates that an intermediary (regardless of whether it implements a cache) MUST NOT change the Content-Encoding, Content-Range or Content-Type response header fields, nor the response representation.

### [3.2.3.](#) Cache Control Extensions

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 private 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.

A cache MUST be ignore unrecognized cache directives; 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).

The HTTP Cache Directive Registry defines the name space for the cache directives.

A registration MUST include the following fields:

- o Cache Directive Name
- o Pointer to specification text

Values to be added to this name space are subject to IETF review ([\[RFC5226\]](#), [Section 4.1](#)).

The registry itself is maintained at <http://www.iana.org/assignments/http-cache-directives>.

### 3.3. Expires

The "Expires" header field gives the date/time after which the response is considered stale. See [Section 2.3](#) 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](#)]; a sender MUST use the [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](#)), that directive overrides the Expires field. Likewise, the s-maxage directive overrides Expires in shared caches.

A server SHOULD NOT send Expires dates more than one year in the future.

A cache MUST treat other invalid date formats, especially including the value "0", as in the past (i.e., "already expired").

### [3.4.](#) Pragma

The "Pragma" 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, a cache SHOULD forward the request toward the origin server even if it has a stored copy of what is being requested. This pragma directive has the same semantics as the no-cache response directive (see [Section 3.2.2](#)) and is defined here for backward compatibility with HTTP/1.0. A client SHOULD include both header fields when a no-cache request is sent to a server not known to be HTTP/1.1 compliant. A cache 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 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

The "Vary" header field conveys the set of 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 satisfy a given request; see [Section 2.7](#). 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 2.7](#).

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 header fields.

A server 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 header fields (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. A proxy MUST NOT generate the "\*" value.

The field-names given are not limited to the set of standard header fields defined by this specification. Field names are case-insensitive.

### 3.6. Warning

The "Warning" 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 payload 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, distinguishes these responses from true failures.

Warning header fields 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 field, 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 header fields SHOULD order them with this user agent behavior in mind. New Warning header fields SHOULD be added after any existing Warning headers fields.

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:

- o 1xx Warnings describe the freshness or validation status of the response, and so MUST be deleted by a cache after validation. They can only be generated by a cache when validating a cached entry, and MUST NOT be generated in any other situation.
- o 2xx Warnings describe some aspect of the representation that is not rectified by a validation (for example, a lossy compression of the representation) and MUST NOT be deleted by a cache 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 header fields 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 field in the message.

If a system 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 field 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

A cache SHOULD include this whenever the returned response is stale.

#### 111 Revalidation failed

A cache SHOULD include this when returning a stale response because an attempt to validate the response failed, due to an inability to reach the server.

#### 112 Disconnected operation

A cache SHOULD include this if it is intentionally disconnected from the rest of the network for a period of time.

#### 113 Heuristic expiration

A cache SHOULD include this if it 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 a proxy if it applies any transformation to the representation, such as changing the content-coding, media-type, or modifying the representation data, 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

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

The freshness model ([Section 2.3](#)) does not necessarily apply to history mechanisms. I.e., a history mechanism can display a previous representation even if it has expired.

This does not prohibit the history mechanism from telling the user that a view might be stale, or from honoring cache directives (e.g., Cache-Control: no-store).

### 5. IANA Considerations

#### 5.1. Cache Directive Registry

The registration procedure for HTTP Cache Directives is defined by [Section 3.2.3](#) of this document.

The HTTP Cache Directive Registry shall be created at [<http://www.iana.org/assignments/http-cache-directives>](http://www.iana.org/assignments/http-cache-directives) and be populated with the registrations below:

Cache Directive	Reference
max-age	<a href="#">Section 3.2.1</a> , <a href="#">Section 3.2.2</a>
max-stale	<a href="#">Section 3.2.1</a>
min-fresh	<a href="#">Section 3.2.1</a>
must-revalidate	<a href="#">Section 3.2.2</a>
no-cache	<a href="#">Section 3.2.1</a> , <a href="#">Section 3.2.2</a>
no-store	<a href="#">Section 3.2.1</a> , <a href="#">Section 3.2.2</a>
no-transform	<a href="#">Section 3.2.1</a> , <a href="#">Section 3.2.2</a>
only-if-cached	<a href="#">Section 3.2.1</a>
private	<a href="#">Section 3.2.2</a>
proxy-revalidate	<a href="#">Section 3.2.2</a>
public	<a href="#">Section 3.2.2</a>
s-maxage	<a href="#">Section 3.2.2</a>
stale-if-error	<a href="#">[RFC5861]</a> , <a href="#">Section 4</a>
stale-while-revalidate	<a href="#">[RFC5861]</a> , <a href="#">Section 3</a>

## 5.2. Header Field Registration

The Message Header Field Registry located at <http://www.iana.org/assignments/message-headers/message-header-index.html> shall be updated with the permanent registrations below (see [\[RFC3864\]](#)):

Header Field Name	Protocol	Status	Reference
Age	http	standard	<a href="#">Section 3.1</a>
Cache-Control	http	standard	<a href="#">Section 3.2</a>
Expires	http	standard	<a href="#">Section 3.3</a>
Pragma	http	standard	<a href="#">Section 3.4</a>
Vary	http	standard	<a href="#">Section 3.5</a>
Warning	http	standard	<a href="#">Section 3.6</a>

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

## 6. Security Considerations

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 need to be protected as sensitive information.

## 7. Acknowledgments

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

### 8.1. Normative References

- [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-13](#) (work in progress), March 2011.
- [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-13](#) (work in progress), March 2011.
- [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-13](#) (work in progress), March 2011.
- [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-13](#) (work in progress), March 2011.
- [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-13](#) (work in progress), March 2011.
- [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

- [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.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [RFC5861] Nottingham, M., "HTTP Cache-Control Extensions for Stale Content", [RFC 5861](#), April 2010.

## Appendix A. Changes from [RFC 2616](#)

Make the specified age calculation algorithm less conservative.  
([Section 2.3.2](#))

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

Clarify denial of service attack avoidance requirement.  
([Section 2.5](#))

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

## 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)

[C.1](#). Since [RFC 2616](#)

Extracted relevant partitions from [[RFC2616](#)].

[C.2](#). Since [draft-ietf-httpbis-p6-cache-00](#)

Closed issues:

- o <http://tools.ietf.org/wg/httpbis/trac/ticket/9>: "Trailer"  
(<http://purl.org/NET/http-errata#trailer-hop>)
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/12>: "Invalidation  
after Update or Delete"  
(<http://purl.org/NET/http-errata#invalidupd>)
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/35>: "Normative and  
Informative references"
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/48>: "Date reference  
typo"
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/49>: "Connection  
header text"
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/65>: "Informative  
references"
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/66>: "ISO-8859-1  
Reference"
- o <http://tools.ietf.org/wg/httpbis/trac/ticket/86>: "Normative up-  
to-date references"

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

Other changes:

- o 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](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/82>>: "rel\_path not used"

Other changes:

- o Get rid of duplicate BNF rule names ("host" -> "uri-host") (work in progress on <<http://tools.ietf.org/wg/httpbis/trac/ticket/36>>)
- o Add explicit references to BNF syntax and rules imported from other parts of the specification.

### C.4. Since [draft-ietf-httpbis-p6-cache-02](#)

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

- o Reference [RFC 3984](#), and update header field registrations for header fields defined in this document.

### C.5. Since [draft-ietf-httpbis-p6-cache-03](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/106>>: "Vary header classification"

### C.6. Since [draft-ietf-httpbis-p6-cache-04](#)

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

- o Use "/" instead of "|" for alternatives.
- o Introduce new ABNF rules for "bad" whitespace ("BWS"), optional whitespace ("OWS") and required whitespace ("RWS").

- o Rewrite ABNFs to spell out whitespace rules, factor out header field value format definitions.

C.7. Since [draft-ietf-httpbis-p6-cache-05](#)

This is a total rewrite of this part of the specification.

Affected issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/54>>: "Definition of 1xx Warn-Codes"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/60>>: "Placement of 13.5.1 and 13.5.2"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/138>>: "The role of Warning and Semantic Transparency in Caching"
- o <<http://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>>):

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

C.8. Since [draft-ietf-httpbis-p6-cache-06](#)

Closed issues:

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

Affected issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/37>>: "Vary and non-existent headers"

C.9. Since [draft-ietf-httpbis-p6-cache-07](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/54>>: "Definition of 1xx Warn-Codes"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/167>>: "Content-Location on 304 responses"

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/169>>: "private and no-cache CC directives with headers"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/187>>: "[RFC2047](#) and warn-text"

C.10. Since [draft-ietf-httpbis-p6-cache-08](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/147>>: "serving negotiated responses from cache: header-specific canonicalization"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/197>>: "Effect of CC directives on history lists"

Affected issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/199>>: Status codes and caching

Partly resolved issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/60>>: "Placement of 13.5.1 and 13.5.2"

C.11. Since [draft-ietf-httpbis-p6-cache-09](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/29>>: "Age calculation"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/168>>: "Clarify differences between / requirements for request and response CC directives"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/174>>: "Caching authenticated responses"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/208>>: "IANA registry for cache-control directives"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/211>>: "Heuristic caching of URLs with query components"

Partly resolved issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/196>>: "Term for the requested resource's URI"

C.12. Since [draft-ietf-httpbis-p6-cache-10](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/109>>: "Clarify entity / representation / variant terminology"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/220>>: "consider removing the 'changes from 2068' sections"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/223>>: "Allowing heuristic caching for new status codes"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/223>>: "Allowing heuristic caching for new status codes"
- o Clean up TODOs and prose in "Combining Responses."

C.13. Since [draft-ietf-httpbis-p6-cache-11](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/204>>: "Text about clock requirement for caches belongs in p6"

C.14. Since [draft-ietf-httpbis-p6-cache-12](#)

Closed issues:

- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/224>>: "Header Classification"
- o <<http://tools.ietf.org/wg/httpbis/trac/ticket/268>>: "Clarify 'public'"

## Index

### A

age 6  
Age header field 18

### C

cache 5  
Cache Directives  
max-age 20, 23

- max-stale 20
- min-fresh 20
- must-revalidate 22
- no-cache 19, 21
- no-store 19, 22
- no-transform 20, 23
- only-if-cached 20
- private 21
- proxy-revalidate 23
- public 21
- s-maxage 23
- Cache-Control header field 18
- cacheable 5

## E

- Expires header field 24
- explicit expiration time 6

## F

- first-hand 6
- fresh 6
- freshness lifetime 6

## G

- Grammar
  - Age 18
  - Age-v 18
  - Cache-Control 19
  - Cache-Control-v 19
  - cache-extension 19
  - cache-request-directive 19
  - cache-response-directive 21
  - delta-seconds 18
  - Expires 25
  - Expires-v 25
  - extension-pragma 25
  - Pragma 25
  - pragma-directive 25
  - Pragma-v 25
  - Vary 26
  - Vary-v 26
  - warn-agent 27
  - warn-code 27
  - warn-date 27
  - warn-text 27
  - Warning 27
  - Warning-v 27
  - warning-value 27

## H

Header Fields  
  Age 18  
  Cache-Control 18  
  Expires 24  
  Pragma 25  
  Vary 26  
  Warning 26  
  heuristic expiration time 6

## M

max-age  
  Cache Directive 20, 23  
max-stale  
  Cache Directive 20  
min-fresh  
  Cache Directive 20  
must-revalidate  
  Cache Directive 22

## N

no-cache  
  Cache Directive 19, 21  
no-store  
  Cache Directive 19, 22  
no-transform  
  Cache Directive 20, 23

## O

only-if-cached  
  Cache Directive 20

## P

Pragma header field 25  
private  
  Cache Directive 21  
private cache 5  
proxy-revalidate  
  Cache Directive 23  
public  
  Cache Directive 21

## S

s-maxage  
  Cache Directive 23  
shared cache 5  
stale 6

V

validator 6

Vary header field 26

W

Warning header field 26

## Authors' Addresses

Roy T. Fielding (editor)  
Adobe Systems Incorporated  
345 Park Ave  
San Jose, CA 95110  
USA

EMail: [fielding@gbiv.com](mailto:fielding@gbiv.com)  
URI: <http://roy.gbiv.com/>

Jim Gettys  
Alcatel-Lucent Bell Labs  
21 Oak Knoll Road  
Carlisle, MA 01741  
USA

EMail: [jg@freedesktop.org](mailto:jg@freedesktop.org)  
URI: <http://gettys.wordpress.com/>

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](mailto:JeffMogul@acm.org)

Henrik Frystyk Nielsen  
Microsoft Corporation  
1 Microsoft Way  
Redmond, WA 98052  
USA

EMail: [henrikn@microsoft.com](mailto:henrikn@microsoft.com)

Larry Masinter  
Adobe Systems Incorporated  
345 Park Ave  
San Jose, CA 95110  
USA

EEmail: LMM@acm.org  
URI: <http://larry.masinter.net/>

Paul J. Leach  
Microsoft Corporation  
1 Microsoft Way  
Redmond, WA 98052

EEmail: 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

EEmail: 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

EEmail: ylafon@w3.org  
URI: <http://www.raubacapeu.net/people/yves/>

Mark Nottingham (editor)

EEmail: 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](mailto:julian.reschke@greenbytes.de)  
URI: <http://greenbytes.de/tech/webdav/>