Independent Submission
Request for Comments: 5861
Category: Informational
ISSN: 2070-1721

M. Nottingham
Yahoo! Inc.
May 2010

HTTP Cache-Control Extensions for Stale Content

Abstract

This document defines two independent HTTP Cache-Control extensions that allow control over the use of stale responses by caches.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5861.

Copyright Notice

Table of Contents

1. Introduction 2
2. Notational Conventions 2
3. The stale-while-revalidate Cache-Control Extension 2
  3.1. Example 3
4. The stale-if-error Cache-Control Extension 3
  4.1. Example 4
5. Security Considerations 5
6. Normative References 5
Appendix A. Acknowledgements 6

1. Introduction

HTTP [RFC2616] requires that caches "respond to a request with the most up-to-date response held... that is appropriate to the request," although "in carefully considered circumstances" a stale response is allowed to be returned. This document defines two independent Cache-Control extensions that allow for such control, stale-if-error and stale-while-revalidate.

The stale-if-error HTTP Cache-Control extension allows a cache to return a stale response when an error -- e.g., a 500 Internal Server Error, a network segment, or DNS failure -- is encountered, rather than returning a "hard" error. This improves availability.

The stale-while-revalidate HTTP Cache-Control extension allows a cache to immediately return a stale response while it revalidates it in the background, thereby hiding latency (both in the network and on the server) from clients.

2. Notational Conventions

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 RFC 2119 [RFC2119].

This specification uses the augmented Backus-Naur Form of RFC 2616 [RFC2616], and it includes the delta-seconds rule from that specification.

3. The stale-while-revalidate Cache-Control Extension

When present in an HTTP response, the stale-while-revalidate Cache-Control extension indicates that caches MAY serve the response in which it appears after it becomes stale, up to the indicated number of seconds.

```
stale-while-revalidate = "stale-while-revalidate" "=" delta-seconds
```

If a cached response is served stale due to the presence of this extension, the cache SHOULD attempt to revalidate it while still serving stale responses (i.e., without blocking).

Note that "stale" implies that the response will have a non-zero Age header and a warning header, as per HTTP's requirements.

If delta-seconds passes without the cached entity being revalidated, it SHOULD NOT continue to be served stale, absent other information.

3.1. Example

A response containing:

```
Cache-Control: max-age=600, stale-while-revalidate=30
```

indicates that it is fresh for 600 seconds, and it may continue to be served stale for up to an additional 30 seconds while an asynchronous validation is attempted. If validation is inconclusive, or if there is not traffic that triggers it, after 30 seconds the stale-while-revalidate function will cease to operate, and the cached response will be "truly" stale (i.e., the next request will block and be handled normally).

Generally, servers will want to set the combination of max-age and stale-while-revalidate to the longest total potential freshness lifetime that they can tolerate. For example, with both set to 600, the server must be able to tolerate the response being served from cache for up to 20 minutes.

Since asynchronous validation will only happen if a request occurs after the response has become stale, but before the end of the stale-while-revalidate window, the size of that window and the likelihood of a request during it determines how likely it is that all requests will be served without delay. If the window is too small, or traffic is too sparse, some requests will fall outside of it, and block until the server can validate the cached response.

4. The stale-if-error Cache-Control Extension

The stale-if-error Cache-Control extension indicates that when an error is encountered, a cached stale response MAY be used to satisfy the request, regardless of other freshness information.

```
stale-if-error = "stale-if-error" "=" delta-seconds
```

When used as a request Cache-Control extension, its scope of application is the request it appears in; when used as a response Cache-Control extension, its scope is any request applicable to the cached response in which it occurs.

Its value indicates the upper limit to staleness; when the cached response is more stale than the indicated amount, the cached response SHOULD NOT be used to satisfy the request, absent other information.

In this context, an error is any situation that would result in a 500, 502, 503, or 504 HTTP response status code being returned.

Note that this directive does not affect freshness; stale cached responses that are used SHOULD still be visibly stale when sent (i.e., have a non-zero Age header and a warning header, as per HTTP's requirements).

4.1. Example

A response containing:

```
HTTP/1.1 200 OK
Cache-Control: max-age=600, stale-if-error=1200
Content-Type: text/plain

success
```

indicates that it is fresh for 600 seconds, and that it may be used if an error is encountered after becoming stale for an additional 1200 seconds.

Thus, if the cache attempts to validate 900 seconds afterwards and encounters:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain

failure
```

the successful response can be returned instead:

```
HTTP/1.1 200 OK
Cache-Control: max-age=600, stale-if-error=1200
Age: 900
Content-Type: text/plain

success
```

After the age is greater than 1800 seconds (i.e., it has been stale for 1200 seconds), the cache must write the error message through.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain

failure
```

5. Security Considerations

The stale-while-revalidate extension provides origin servers with a mechanism for dictating that stale content should be served from caches under certain circumstances, with the expectation that the cached response will be revalidated in the background. It is suggested that such validation be predicated upon an incoming request, to avoid the possibility of an amplification attack (as can be seen in some other pre-fetching and automatic refresh mechanisms). Cache implementers should keep this in mind when deciding the circumstances under which they will generate a request that is not directly initiated by a user or client.

The stale-if-error provides origin servers and clients a mechanism for dictating that stale content should be served from caches under certain circumstances, and does not pose additional security considerations over those of RFC 2616, which also allows stale content to be served.

6. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

Appendix A. Acknowledgements

Thanks to Ben Drees, John Nienart, Henrik Nordstrom, Evan Torrie, and Chris Westin for their suggestions. The author takes all responsibility for errors and omissions.

Author's Address

Mark Nottingham
Yahoo! Inc.

EMail: mnot@yahoo-inc.com
URI: http://www.mnot.net/