

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 10, 2008

M. Nottingham
Yahoo! Inc.
May 9, 2008

The stale-while-revalidate HTTP Cache-Control Extension
draft-nottingham-http-stale-while-revalidate-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

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

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on November 10, 2008.

Abstract

The stale-while-revalidate HTTP response Cache-Control extension allows servers to instruct caches to serve stale responses while validating them, to avoid latency in some situations.

Internet-Draft

stale-while-revalidate

May 2008

Table of Contents

1.	Introduction	3
2.	Notational Conventions	3
3.	The stale-while-revalidate Cache-Control Extension	3
4.	Example	3
5.	Security Considerations	4
6.	IANA Considerations	4
7.	Normative References	4
Appendix A.	Acknowledgements	5
	Author's Address	5
	Intellectual Property and Copyright Statements	6

1. Introduction

The potential for latency (due to the network as well as server processing) introduced by cache validation in HTTP [[RFC2616](#)] is often undesirable; while subsequent requests can be served from the cache quickly, the request that triggers validation sees degraded service.

In some situations, it may be useful to avoid this latency, at the cost of serving slightly stale responses. the stale-while-revalidate HTTP response Cache-Control extension allows caches to do this.

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 [RFC2616](#) [[RFC2616](#)], and 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 it appears in 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 MUST NOT continue to be served stale, absent other information.

[4.](#) Example

A response containing:

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

Nottingham

Expires November 10, 2008

[Page 3]

Internet-Draft

stale-while-revalidate

May 2008

indicates that it is fresh for 600 seconds, and it may continue to be served stale for up to 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 too sparse, some requests will fall outside of it, and block until the server can validate the cached response.

[5.](#) Security Considerations

This document 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.

6. IANA Considerations

This document has no actions for IANA.

7. 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.,

Nottingham

Expires November 10, 2008

[Page 4]

Internet-Draft

stale-while-revalidate

May 2008

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/>

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.