The stale-while-revalidate HTTP Cache-Control Extension
draft-nottingham-http-stale-while-revalidate-00

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/1id-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 August 23, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

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

1. Introduction ................................................. 3
2. Notational Conventions ...................................... 3
3. The stale-while-revalidate Cache-Control Extension .......... 3
4. Example ..................................................... 4
5. Security Considerations ..................................... 4
6. 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 origin servers to instruct 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 BCP 14 [RFC2119], as scoped to those conformance targets.

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 shared caches MAY serve the response it appears in after it becomes stale, up to the indicated number of seconds.

\[
\text{stale-while-revalidate} = \text{"stale-while-revalidate" } \text{"=} \text{ delta-seconds}
\]

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.

Private caches MUST ignore this Cache-Control extension.
4. 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 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 mechanism 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. Normative References

Appendix A. Acknowledgements

Thanks to 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/