Workgroup: Building Blocks for HTTP APIs
Internet-Draft:
draft-thomson-httpapi-date-requests-00
Published: 9 February 2022
Intended Status: Standards Track
Expires: 13 August 2022
Authors: M. Thomson
Mozilla

Using The Date Header Field In HTTP Requests

Abstract

HTTP clients rarely make use of the Date header field when making requests. This document describes considerations for using the Date header field in requests. A method is described for correcting erroneous in Date request header fields that might arise from differences in client and server clocks. The risks of applying that correction technique are discussed.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/http-request-date/draft-thomson-httpapi-date-requests.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-httpapi-date-requests/.

Discussion of this document takes place on the Building Blocks for HTTP APIs Working Group mailing list (mailto:httpapi@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/httpapi/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/http-request-date.

Status of This Memo

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 https://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 13 August 2022.

Table of Contents

1. Introduction

Many HTTP requests are timeless. That is, the contents of the request are not bound to a specific point in time. Thus, the use of the HTTP Date header field in requests is rare; see Section 6.6.1 of [HTTP].

However, in some contexts, it is important that a request only be valid over a small period of time. One such context is when requests are signed [SIGN], where including a time in a request might prevent a signed request from being reused at another time. Similarly, some uses of OHTTP [OHTTP] might depend on the same sort of replay

protection. It is possible to make anti-replay protections at servers more efficient if requests from either far in the past or into the future can be rejected.

This document describes some considerations for using the Date request header field in Section 3. A new type of problem report [PROBLEM] is defined in Section 4 for use in rejecting requests with a missing or incorrect Date request header field.

Section 5 explores the consequences of using Date header field in requests when client and server clocks do not agree. A method for recovering from differences in clocks is described in Section 5.1. Section 5.2 describes the privacy considerations that apply to this technique.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Date in HTTP Requests

Most HTTP clients have no need to use the Date header field in requests. This only changes if it is important that the request not be considered valid at another time. As requests are - by default - trivially copied, stored, and modified by any entity that can read them, the addition of a Date header field is unlikely to be useful in many cases.

Signed HTTP requests are one example of where requests might be available to entities that are not permitted to alter their contents. Adding a Date request header field - and signing it - ensures that the request cannot be used at a very different time to what was intended.

OHTTP [OHTTP] is another example of where capture and replay of a request might be undesirable. Here, a partially trusted intermediary, an oblivious proxy resource, receives encapsulated HTTP requests. Though this entity cannot read or modify these messages, it is able to delay or replay them. The inclusion of a Date header field in these requests might be used to limit the time over which delay or replay is possible.

In both cases, the inclusion of a Date request header field might be part of an anti-replay strategy at a server. A simple anti-replay scheme starts by choosing a window of time anchored at the current time. Requests with timestamps that fall within this period are

remembered and rejected if they appear again; requests with timestamps outside of this window are rejected. This scheme works for any monotonic value (see for example Section 3.4.3 of [RFC4303]) and allows for efficient rejection of duplicate requests with minimal state.

4. Date Not Acceptable Problem Type

A server can send a 400-series status code in response to a request where the Date request header field is either absent or indicates a time that is not acceptable to the server. Including content of type "application/problem+json" (or "application/problem+xml"), as defined in [PROBLEM], in that response allows the server to provide more information about the error.

This document defines a problem type of "https://iana.org/assignments/http-problem-types#date" for indicating that the Date request header field is missing or incorrect. Figure 1 shows an example response in HTTP/1.1 format.

```
HTTP/1.1 400 Bad Request
Date: Mon, 07 Feb 2022 00:28:05 GMT
Content-Type: application/problem+json
Content-Length: 128

{"type":"https://iana.org/assignments/http-problem-types#date",
"title": "date field in request outside of acceptable range"}
```

Figure 1: Example Response

A server **MUST** include a Date response header field in any responses that use this problem detail type.

In processing a Date header field in a request, a server **MUST** allow for delays in transmitting the request, retransmissions performed by transport protocols, plus any processing that might occur in the client and any intermediaries, and those parts of the server prior to processing the field. Additionally, the Date header field is only capable of expressing time with a resolution of one second. These factors could mean that the value a server receives could be some time in the past.

Differences between client and server clocks are likely to be a source of most disagreements between the server time and the time expressed in Date request header field. Section 5 will explore this problem in more detail and offer some means of handling these disagreements.

5. Clock Skew

Perfect synchronization of client and server clocks is an ideal state that generally only exists in tightly controlled settings. In practice, despite good availability of time services like NTP [NTP] Internet-connected endpoints often disagree about the time (see for example Section 7.1 of [CLOCKSKEW]).

The prevalence of clock skew could justify servers being more tolerant of a larger range of values for the Date request header field. This includes accepting times that are a short duration into the future in addition to times in the past.

For a server that uses the Date request header field to limit the state kept for anti-replay purposes, the amount of state might be all that determines the range of values it accepts.

5.1. Date Correction

Even when a server is tolerant of small clock errors, a valid request from a client can be rejected if the client clock is outside of the range of times that a server will accept. A server might also reject a request when the client makes a request without a Date header field.

A client can recover from a failure that caused by a bad clock by adjusting the time and re-attempting the request.

For a fresh response (see Section 4.2 of [CACHING]), the client can re-attempt the request, copying the Date header field from the response into its new request. If the response is stale, the client can add the age of the response to determine the time to use in a re-attempt; see Section 5.3 for more.

In addition to adjusting for response age, the client can adjust the time it uses based on the elapsed time since it estimates when the response was generated. Note however that if the client retries a request immediately, any additional increment is likely to be less than the one second resolution of the Date header field under most network conditions.

5.2. Limitations of Date Correction

Clients **MUST NOT** accept the time provided by an arbitrary HTTP server as the basis for system-wide time. Even if the client code in question were able to set the time, altering the system clock in this way exposes clients to attack. The source of system time information needs to be trustworthy as the current time is a critical input to security-relevant decisions, such as whether to accept a server certificate [RFC6125].

Use of date correction allows requests that use the correction to be correlated. Limitations on use of date corrections is necessary to ensure privacy. An immediate retry of an identical request with an update Date header field is safe in that it only provides the server with the ability to match the retry to the original request.

Anything other than an immediate retry requires careful consideration of the privacy implications. Use of the same date correction for other requests can be used to link those requests to the same client. Using the same date correction is equivalent to connection reuse, cookies, TLS session tickets, or other state a client might carry between requests. Linking requests might be acceptable, but in general only where other forms of linkage already exist.

Clients **MUST NOT** use the time correction from one server when making requests of another server. Using the same date correction across different servers might be used by servers to link client identities and to exchange information via a channel provided by the client.

For clients that maintain per-server state, the specific date correction that is used for each server **MUST** be cleared when removing other state for that server to prevent re-identification. For instance, a web browser that remembers a date correction would forget that correction when removing cookies and other state.

5.3. Intermediaries and Date Corrections

Some intermediaries, in particular those acting as reverse proxies or gateways, will rewrite the Date header field in responses. This applies especially to responses served from cache, but this might also apply to those that are forwarded directly from an origin server.

For responses that are forwarded by an intermediary, changes to the Date response header field will not change how the client corrects its clock. Errors only occur if the clock at the intermediary differs significantly from the clock at the origin server or if the intermediary updates the Date response header field without also adjusting or removing the Age header field on a stale response.

Servers that condition their responses on the Date header field **SHOULD** either ensure that intermediaries do not cache responses (by including a Cache-Control directive of no-store) or designate the response as conditional on the value of the Date request header field (by including the token "date" in a Vary header field).

6. Security Considerations

Including a Date header field in requests reveals information about the client clock. This might be used to identify clients with vulnerability to attacks that depend on incorrect clocks.

Section 5.2 contains a discussion of the security and privacy concerns associated with date correction.

7. IANA Considerations

IANA are requested to create a new entry in the "HTTP Problem Type" registry established by [PROBLEM].

Type URI: https://iana.org/assignments/http-problem-types#date

Title: Date Not Acceptable

Recommended HTTP Status Code: 400

Reference: Section 4 of this document

8. References

8.1. Normative References

[CACHING] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Caching", Work in Progress, Internet-Draft, draft-ietf-httpbis-cache-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cache-19>.

[PROBLEM] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", Work in Progress, Internet-Draft, draft-ietf-httpapi-rfc7807bis-01, 13 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpapi-rfc7807bis-01>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

8.2. Informative References

[CLOCKSKEW]

Acer, M., Stark, E., Felt, A., Fahl, S., Bhargava, R., Dev, B., Braithwaite, M., Sleevi, R., and P. Tabriz, "Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors", Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, DOI 10.1145/3133956.3134007, October 2017, <https://doi.org/10.1145/3133956.3134007>.

[HTTP] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19>.

[NTP] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/rfc/rfc5905>.

[OHTTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-00, 25 November 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-ohai-ohttp-00>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/rfc/rfc4303>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/rfc/rfc6125>.

[SIGN] Backman, A., Richer, J., and M. Sporny, "HTTP Message Signatures", Work in Progress, Internet-Draft, draft-ietf-httpbis-message-signatures-08, 28 January 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-message-signatures-08>.

Acknowledgments

This document is a result of discussions about how to provide anti-replay protection for OHTTP in which Mark Nottingham, Eric Rescorla, and Chris Wood were instrumental.

Author's Address

Martin Thomson

Mozilla

Email: mt@lowentropy.net