The Priority HTTP Header Field
draft-kazuho-httpbis-priority-00

Abstract

   This document describes the Priority HTTP header field. This header
   field can be used by endpoints to specify the absolute precedence of
   an HTTP response in an HTTP-version-independent way.

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 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 January 9, 2020.

Table of Contents

1. Introduction

It is common for an HTTP ([RFC7230]) resource representation to have
relationships to one or more other resources. Clients will often
discover these relationships while processing a retrieved
representation, leading to further retrieval requests. Meanwhile,
the nature of the relationship determines whether the client is
blocked from continuing to process locally available resources. For
example, visual rendering of an HTML document could be blocked by the
retrieval of a CSS file that the document refers to. In contrast,
inline images do not block rendering and get drawn progressively as
the chunks of the images arrive.

To provide meaningful representation of a document at the earliest
moment, it is important for an HTTP server to prioritize the HTTP
responses, or the chunks of those HTTP responses, that it sends.

HTTP/2 ([RFC7540]) provides such a prioritization scheme. A client
sends a series of PRIORITY frames to communicate to the server a
"priority tree"; this represents the client's preferred ordering and
weighted distribution of the bandwidth among the HTTP responses.
However, the design has shortcomings:

o Its complexity has led to varying levels of support by HTTP/2
   clients and servers.

o It is hard to coordinate with server-driven prioritization. For
   example, a server, with knowledge of the document structure, might
   want to prioritize the delivery of images that are critical to
   user experience above other images, but below the CSS files. But
   with the HTTP/2 prioritization scheme, it is impossible for the
   server to determine how such images should be prioritized against
   other responses that use the client-driven prioritization tree,
   because every client builds the HTTP/2 prioritization tree in a
   different way.

o It does not define a method that can be used by a server to
   express the priority of a response. Without such a method,
   intermediaries cannot coordinate client-driven and server-driven
   priorities.

o The design cannot be ported cleanly to HTTP/3
   ([I-D.ietf-quic-http]). One of the primary goals of HTTP/3 is to
   minimize head-of-line blocking. Transmitting the evolving
   representation of a "prioritization tree" from the client to the
   server requires head-of-line blocking.

Based on these observations, this document defines the Priority HTTP
header field that can be used by both the client and the server to
specify the precedence of HTTP responses in a standardized,
extensible, protocol-version- independent, end-to-end format. This
header-based prioritization scheme can act as a substitute for the
HTTP/2 frame-based prioritization scheme (see Section 4).

1.1. 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 [RFC2119].

The terms sh-token and sh-boolean are imported from
[I-D.ietf-httpbis-header-structure].

Example HTTP requests and responses use the HTTP/2-style formatting
from [RFC7540].

2. The Priority HTTP Header Field

The Priority HTTP header field can appear in requests and responses.
A client uses it to specify the priority of the response. A server
uses it to inform the client that the priority was overwritten. An
intermediary can use the Priority information from client requests
and server responses to correct or amend the precedence to suit it
(see Section 3).

The value of the Priority header field is a Structured Headers [I-D.ietf-httpbis-header-structure] Dictionary. Each dictionary member represents a parameter of the Priority header field. This document defines the "urgency" and "progressive" parameters. Values of these parameters MUST always be present. When any of the defined parameters are omitted, or if the Priority header field is not used, their default values SHOULD be applied.

Unknown parameters MUST be ignored.

2.1. urgency

The "urgency" parameter takes one of the following sh-tokens as the value that indicates how an HTTP response affects the usage of other responses:

o "blocking" indicates that the response prevents other responses from being used.

o "document" indicates that the response contains the document that is being processed.

o "non-blocking" indicates that the response does not prevent the client from using the document even though the response is being used or referred to by the document.

The default value is "document".

A server SHOULD transmit HTTP responses in the order of their urgency: "blocking" first, followed by "document", followed by "non-blocking".

The following example shows a request for a CSS file with the urgency set to "blocking":

```
:method = GET
:scheme = https
:authority = example.net
:path = /style.css
priority = urgency=blocking
```

2.2. progressive

The "progressive" parameter takes an sh-boolean as the value that indicates if a response can be processed progressively, i.e. provide some meaningful output as chunks of the response arrive.

The default value of the "progressive" parameter is "0".

A server SHOULD distribute the bandwidth of a connection between
progressive responses that share the same urgency.

A server SHOULD transmit non-progressive responses one by one,
preferably in the order the requests were generated. Doing so
maximizes the chance of the client making progress in using the
composition of the HTTP responses at the earliest moment.

The following example shows a request for a JPEG file with the
urgency parameter set to "non-blocking" and the progressive parameter
set to "1".

```
:method = GET
:scheme = https
:authority = example.net
:path = /image.jpg
priority = urgency=non-blocking, progressive=?1
```

3. Merging Client- and Server-Driven Parameters

It is not always the case that the client has the best view of how
the HTTP responses should be prioritized. For example, whether a
JPEG image should be served progressively by the server depends on
the structure of that image file - a property only known to the
server.

Therefore, a server is permitted to send a "Priority" response header
field. When used, the parameters found in this response header field
override those specified by the client.

For example, when the client sends an HTTP request with

```
:method = GET
:scheme = https
:authority = example.net
:path = /image.jpg
priority = urgency=non-blocking, progressive=?1
```

and the origin responds with

```
:status = 200
content-type = image/jpeg
priority = progressive=?0
```

the intermediary's view of the progressiveness of the response
becomes negative, because the server-provided value overrides that
provided by the client. The urgency is deemed as "non-blocking",
because the server did not specify the parameter.

4. Coexistence with HTTP/2 Priorities

   Standard HTTP/2 ([RFC7540]) endpoints use frame-based prioritization,
   whereby a client sends priority information in dedicated fields
   present in HEADERS and PRIORITY frames. A client might instead
   choose to use header-based prioritization as specified in this
   document.

4.1. The SETTINGS_HEADER_BASED_PRIORITY SETTINGS Parameter

   To improve communication of the client's intended prioritization
   scheme, this document specifies a new HTTP/2 SETTINGS parameter with
   the name "SETTINGS_HEADER_BASED_PRIORITY". The value of the
   parameter MUST be 0 or 1; the initial value is 0. Frame-based
   prioritization is respected when the value is 0, or when the server
   does not recognize the setting.

   An HTTP/2 client that uses header-based priority SHOULD send a
   "SETTINGS_HEADER_BASED_PRIORITY" parameter with a value of 1 when
   connecting to a server.

   An intermediary SHOULD send a "SETTINGS_HEADER_BASED_PRIORITY"
   parameter with a value of 1 for a connection it establishes when, and
   only when, all the requests to be sent over that connection originate
   from a client that utilizes this header-based prioritization scheme.
   Otherwise this settings parameter SHOULD be set to 0.

   A client or intermediary MUST NOT send a
   "SETTINGS_HEADER_BASED_PRIORITY" parameter with the value of 0 after
   previously sending a value of 1.

   A server MUST NOT send a "SETTINGS_HEADER_BASED_PRIORITY" parameter.
   Upon receipt, a client that supports header-based prioritization MUST
   close the connection with a protocol error. Non-supporting clients
   will ignore this extension element (see [RFC7540], Section 5.5).

5. Considerations

5.1. Why use an End-to-End Header Field?

   Contrary to the prioritization scheme of HTTP/2 that uses a hop-by-
   hop frame, the Priority header field is defined as end-to-end.

   The rationale is that the Priority header field transmits how each
   response affects the client's processing of those responses, rather
   than how relatively urgent each response is to others. The way a
   client processes a response is a property associated to that client
   generating that request. Not that of an intermediary. Therefore, it

is an end-to-end property. How these end-to-end properties carried
by the Priority header field affect the prioritization between the
responses that share a connection is a hop-by-hop issue.

Having the Priority header field defined as end-to-end is important
for caching intermediaries. Such intermediaries can cache the value
of the Priority header field along with the response, and utilize the
value of the cached header field when serving the cached response,
only because the header field is defined as end-to-end rather than
hop-by-hop.

It should also be noted that the use of a header field carrying a
textual value makes the prioritization scheme extensible; see the
discussion below.

5.2. Why are there Only Three Levels of Urgency?

One of the aims of this specification is to define a mechanism for
merging client- and server-provided hints for prioritizing the
responses. For that to work, each urgency level needs to have a
well-defined meaning. As an example, a server can assign the highest
precedence among the non-blocking responses to an HTTP response
carrying an icon, because the meaning of "non-blocking" is shared
among the endpoints.

This specification restricts itself to defining just three levels of
urgency, in order to provide sufficient granularity for prioritizing
responses for ordinary web browsing, at minimal complexity.

However, that does not mean that the prioritization scheme would
forever be stuck to the three levels. The design provides
extensibility. If deemed necessary, it would be possible to divide
any of the three urgency levels into sub-levels by defining a new
parameter. As an example, a server could assign an "importance"
parameter to the priority of each image that it provides, so that an
intermediary could prioritize certain images above others. Or, a
graphical user-agent could send a "visible" parameter to indicate if
the resource being requested is within the viewport.

A server can combine the hints provided in the Priority header field
with other information in order to improve the prioritization of
responses. For example, a server that receives requests for a font
[RFC8081] and images with the same urgency might give higher
precedence to the font, so that a visual client can render textual
information at an early moment.

6. Security Considerations

   TBD

7. IANA Considerations

   This specification registers the following entry in the Permanent
   Message Header Field Names registry established by [RFC3864]:

   Header field name: Priority

   Applicable protocol: http

   Status: standard

   Author/change controller: IETF

   Specification document(s): This document

   Related information: n/a

   This specification registers the following entry in the HTTP/2
   Settings registry established by [RFC7540]:

   Name: SETTINGS_HEADER_BASED_PRIORITY:

   Code: 0xTBD

   Initial value: 0

   Specification: This document

8. References

8.1. Normative References

   [I-D.ietf-httpbis-header-structure]
            Nottingham, M. and P. Kamp, "Structured Headers for HTTP",
            draft-ietf-httpbis-header-structure-10 (work in progress),
            April 2019.

   [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/info/rfc2119>.

   [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <https://www.rfc-editor.org/info/rfc7230>.

   [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext
              Transfer Protocol Version 2 (HTTP/2)", RFC 7540,
              DOI 10.17487/RFC7540, May 2015, <https://www.rfc-
              editor.org/info/rfc7540>.

8.2. Informative References

   [I-D.ietf-quic-http]
              Bishop, M., "Hypertext Transfer Protocol Version 3
              (HTTP/3)", draft-ietf-quic-http-20 (work in progress),
              April 2019.

   [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration
              Procedures for Message Header Fields", BCP 90, RFC 3864,
              DOI 10.17487/RFC3864, September 2004, <https://www.rfc-
              editor.org/info/rfc3864>.

   [RFC8081] Lilley, C., "The "font" Top-Level Media Type", RFC 8081,
              DOI 10.17487/RFC8081, February 2017, <https://www.rfc-
              editor.org/info/rfc8081>.

Appendix A. Acknowledgements

   Many thanks to Robin Marx, Patrick Meenan and Ian Swett for their
   feedback.

Authors' Addresses

   Kazuho Oku
   Fastly

   Email: kazuhooku@gmail.com

   Lucas Pardue
   Cloudflare

   Email: lucaspardue.24.7@gmail.com