HTTP-client suggested Push Preference
draft-pot-prefer-push-01

Abstract

"Prefer-Push" is a HTTP header that a client may use to request that a server uses HTTP/2 Push to send related resources as identified by their link relationships.

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 November 3, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

HTTP/2 [RFC7540] allows a server to push request and response pairs to HTTP clients. This can save round-trips between server and client and reduces the total time required for a client to retrieve all requested resources.

This mechanism is completely controlled by the server, and it is up to implementors of services to anticipate what resources a client might need next.

This specification defines a new HTTP header that allows a client to inform a server of resources they will require next based on a link relation type [RFC8288].

2. Rationale

Many HTTP-based services provide some mechanism to embed the HTTP response bodies of resources into other HTTP resource. A common example of this is when a resource is structured as a "collection of resources". Examples of this include:

- The Atom Syndication Format [RFC4287] that encodes "ATOM:entry" XML elements for each subordinate.
- The [HAL] format, which provides an "_embedded" element to embedding bodies of resources in other resources.
- The [JSON-API] format, which provides a "included" property to embed resources.

Embedding resource responses in other resources has two major performance advantages:

1. It reduces the number of roundtrips. A client can make a single HTTP request and get many responses.

2. Generating a related set of resources can often be implemented on a server to be less time consuming than generating each response individually.

These mechanisms also pose an issue. To HTTP clients and intermediaries such as proxies and caches resources are opaque. They are not aware of a concept of embedded resources.

One example where this might fail is if a client receives a resource, embedded in another resource, a cache might not be aware of this.
resource and serve a stale, older version when this resource is requested directly.

To keep the performance advantage of being able to generate a related set of HTTP responses together, HTTP/2 Push could be an alternative to embedding.

HTTP/2 Push allows the server to initiate a request and response pair and send them to the client early if the server thinks it will need them. Another advantage of HTTP/2 push over embedding is that it allows resources of mixed mediatypes to be pushed.

Servers can however not always anticipate which resources a client might want pushed. To avoid guessing, this specification introduces a "Prefer-Push" header that allows a client to inform a server which resources they will need next.

In many REST APIs, subordinate or embedded resources are identified by their link relation. By using the link relation, it will be possible for a client to indicate to a server which links they intend to follow, allowing a server to only push the resources that the client knows it will need.

3. The header format

This format should the "List" Data Type from the Structured Headers specification [I-D.ietf-httpbis-header-structure].

GET /articles HTTP/1.1
Prefer-Push: item, author, "https://example.org/custom-rel"

Each item in the list is a link relationship, as described in Web Links [RFC8288].

4. Handling a Prefer-Push request

When a server receives the "Prefer-Push" header, it can choose to push the related resources. It’s up to the discretion of the implementor to decide which resources to push. A server is also free to ignore push-requests.

If a server chooses to act on an item in the "Prefer-Push" list, the Link Relationship should exist at the target resource. This specification does not require that the link relationships get returned as HTTP "Link" headers. The "Link" may be defined as "<links>" HTML element, or as a JSON property. How the link is serialized is dependent on the media type.
5. Using with "preload" relationship types

[W3C.CR-preload-20171026] defines a "preload" relationship type. This relationship type can be used by an origin to inform a client or intermediate to start fetching a resource, or a proxy to initiate a HTTP/2 push.

A distinct difference between "preload" and "Prefer-Push" is that "preload" can be used by origin servers to inform clients and intermediates to fetch and potentially push resources optimistically, but fundamentally "Prefer-Push" is a completely client-driven mechanism.

These features can co-exist, but a wide adoption of client-driven suggestions for pushes might eventually make "preload" unnecessary as in most cases clients will have a better knowledge of the resources they need.

6. Security considerations

The Prefer-Push mechanism can potentially result in a large number of resources being pushed. This can result in a Denial-of-Service attack.

A server must set reasonable restrictions around the number of pushed resources.

7. IANA considerations

This document defines the "Prefer-Push" HTTP request fields and registers them in the Permanent Message Header Fields registry.

7.1. Prefer-Push

- Header field name: Prefer-Push
- Applicable protocol: HTTP
- Status: standard
- Author/Change controller: IETF
- Specification document(s): Section 7.1 of this document
- Related information: for Client Hints
8. Acknowledgements

9. References

9.1. 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.

Transfer Protocol Version 2 (HTTP/2)", RFC 7540,
DOI 10.17487/RFC7540, May 2015,

[RFC8288] Nottingham, M., "Web Linking", RFC 8288,
DOI 10.17487/RFC8288, October 2017,

[W3C.CR-preload-20171026]
Grigorik, I. and Y. Weiss, "Preload", World Wide Web
Consortium CR CR-preload-20171026, October 2017,

9.2. Informative References

[HAL] Kelly, M., "JSON Hypertext Application Language",
June 2012,


Syndication Format", RFC 4287, DOI 10.17487/RFC4287,

Appendix A. Example

A server serves a document with a JSON-based media-type. The
following example document might represent a list of articles:
HTTP/1.1 200 OK
Content-Type: application/vnd.example.links+json

{
    "links": [
        { "rel": "item", "href": "/article/1" },
        { "rel": "item", "href": "/article/2" },
        { "rel": "item", "href": "/article/3" },
        { "rel": "item", "href": "/article/4" },
        { "rel": "item", "href": "/article/5" }
    ],
    "total": 5,
}

A "Prefer-Push"-enabled client knows it will want to receive the full representations of all articles. When the client receives the list of articles via a "GET" request, it can indicate this preference with the "Prefer-Push" header:

GET /article HTTP/1.1
Accept: application/vnd.example.links+json
Prefer-Push: item

Upon receiving this request, server may immediately generate the request and response pairs for every "item" link in the collection and initiate push streams for each.

Appendix B. Changelog

B.1. Changes since -00
   o Added an abstract
   o Updated rationale section significantly.

Author’s Address

Evert Pot

Email: me@evertpot.com
URI: https://evertpot.com/