Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 5, 2019

E. Pot

December 02, 2018

# HTTP-client suggested Push Preference
## draft-pot-prefer-push-00

Abstract

TODO

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 June 5, 2019.

## 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 peformance 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 that generating each response individually.

These mechanism also poses an issue. HTTP clients and intermediaries are not aware of these embedded resources, because there was never a true HTTP request.

By leveraging HTTP/2 push instead of format-specific embedding mechanisms, it's possible for services to push subordinate resources as soon as possible, generate HTTP responses as a "set" all while

still taking advantage of existing HTTP infrastructure. Another advantage of HTTP/2 push over embedding it that it allows resources of mixed mediatypes to be pushed.

In many REST apis, sub-ordiniate 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 intent to follow, allowing a server to only push the resources that the client knows it will need.

## 3. The header format

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

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

## 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.

[RFC8288] defines Web Links as an abstract concept that can be specified in a variety of ways. It defines the HTTP "Link" header as a specific serialization. Like [RFC8288], this specification is not dependent on the serialization of the Web Link.

## 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.

As such, these features can co-exist.

## 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 amount of pushes it sends. In the case of N-Depth pushes, servers SHOULD also set restrictions around the depth it supports.

## 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-09 (work in progress), December 2018.

[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>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[W3C.CR-preload-20171026]
Grigorik, I. and Y. Weiss, "Preload", World Wide Web Consortium CR CR-preload-20171026, October 2017, <https://www.w3.org/TR/2017/CR-preload-20171026>.

### 9.2. Informative References

[HAL] Kelly, M., "JSON Hypertext Application Language", June 2012, <https://tools.ietf.org/html/draft-kelly-json-hal-00>.

[JSON-API]
"JSON:API", n.d., <https://jsonapi.org/format/>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <https://www.rfc-editor.org/info/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 recieving this request, server may immediately generate the request and response pairs for every "item" link in the collection and initiate push streams for each.

## Author's Address

Evert Pot

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