Network Working Group
Internet-Draft
Updates: 5789 (if approved)
Intended status: Informational
Expires: September 15, 2014

M. Nottingham
March 14, 2014

The 2NN Patch HTTP Status Code
draft-nottingham-http-patch-status-00

Abstract

This document specifies the 2NN Patch HTTP status code to allow servers to perform partial updates of stored responses in client caches.

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 September 15, 2014.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Notational Conventions 3
2. The 2NN Patch Status Code 3
2.1. The Patched Header Field 5
3. IANA Considerations 5
3.1. 2NN Patch HTTP Status Code 5
3.2. Accept-Patch Header Field 5
3.3. Patched Header Field 5
4. Security Considerations 5
5. References 6
5.1. Normative References 6
5.2. Informative References 6
Appendix A. 2NN Patch and HTTP/2 Server Push 6
Author's Address 8

1. Introduction

[RFC5246] defines the HTTP PATCH method as a means of selectively updating the state of a resource on a server. This document complements that specification by specifying a means for a server to selectively update a stored response on a client - usually, in a cache [I-D.ietf-httpbis-p6-cache].

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].

This document uses the Augmented BNF defined by [RFC5246], and additionally uses the entity-tag rule defined in [I-D.ietf-httpbis-p4-conditional].

2. The 2NN Patch Status Code

The 2NN (Patch) status code allows a response to patch a stored response in a cache, by reusing the patch formats of [RFC5789]. In some sense, it is the complement of the HTTP PATCH request method.

TODO: is this a 2NN or 3xx?

Clients can advertise support for 2NN (Patch), along with the patch formats supported in it, by using the Accept-Patch header field in requests. For example:

```
GET /foo HTTP/1.1
Host: api.example.com
Accept-Patch: application/patch+json
If-None-Match: "abcdef", "ghijkl"
User-Agent: Example/1.0
```

If the server can generate a patch for one of the entity tags provided in If-None-Match, in one of the accepted patch formats, it can generate a 2NN (Patch) response:

```
HTTP/1.1 2NN Patch
Content-Type: application/patch+json
Patched: "ghijkl"
ETag: "mnopqrs"
```

The entity tag carried by the ETag header field is associated with the selected representation - i.e., the stored response after the

patch is applied.

The Patched header field identifies the representation to apply the patch to, as indicated by the entity-tag provided in If-None-Match request header field; see Section 2.1.

Therefore, in the example above, the stored response "ghijkl" is being patched, with the resulting stored response having the entity tag "mnopqrs".

Application of a 2NN (Patch) response happens in a manner very similar to the process for freshening a stored response by applying a 304 (Not Modified), as described in [I-D.ietf-httpbis-p6-cache], Section 4.3.4.

In particular, the stored response to apply a 2NN (Patch) response to is the same; if none is selected, the patch fails, and the client MAY resubmit the request without an Accept-Patch header field, in order to get a full response.

If a stored response is selected, clients MUST update it in the following manner:

- The value of the Content-Length header field MUST be adjusted to reflect the length of the patched response body.
- The ETag header field MUST be replaced (or inserted, if not present) with the value of the Patched header field in the 2NN response (if present).
- Other header fields in the 2NN response MUST update the stored response, in the same manner as described in [I-D.ietf-httpbis-p6-cache], Section 4.3.4. However, the following fields MUST not be updated: Content-Type, Patched.

The 2NN (Patch) status code SHOULD NOT be generated if the request did not include If-None-Match, unless conflicts are handled by the patch format itself (e.g., allowing a patch to append to an array), or externally.

Intermediaries MAY append the Accept-Patch header field to requests, or append new values to it, if they will process 2NN responses for the patch format(s) they add. Likewise, intermediaries MAY generate 2NN (Patch) responses under the conditions specified here.

The 2NN status code is not cacheable by default, and is not a representation of any identified resource.

2.1. The Patched Header Field

The Patched header field identifies the stored representation that a patch is to be applied to in a 2NN (Patch) response.

```
Patched = entity-tag
```

3. IANA Considerations

3.1. 2NN Patch HTTP Status Code

This document defines the 2NN (Patch) HTTP status code, as per [I-D.ietf-httpbis-p2-semantics].

- Status Code (3 digits): TBD
- Short Description: Patch
- Pointer to specification text: Section 2

3.2. Accept-Patch Header Field

This document updates [RFC5789] to allow the Accept-Patch HTTP header field to be used in requests, which ought to be reflected in the registry.

3.3. Patched Header Field

This document defines a new HTTP header, field, "Patched", to be registered in the Permanent Message Header Registry, as per [RFC3864].

- Header field name: Patched
- Applicable protocol: http
- Status: standard
- Author/Change controller: IETF
- Specification document(s): [this document]
- Related information:

4. Security Considerations

2NN (Patch) can be brittle when the application of a patch fails, because the client has no way to report the failure of a patch to the server. This assymetry might be exploited by an attacker, but can be mitigated by judicious use of strong ETags.

Some patch formats might have additional security considerations.

5. References

5.1. Normative References

[I-D.ietf-httpbis-p4-conditional]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", draft-ietf-httpbis-p4-conditional-26 (work in progress), February 2014.

[I-D.ietf-httpbis-p6-cache]
Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", draft-ietf-httpbis-p6-cache-26 (work in progress), February 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

5.2. Informative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-10 (work in progress), February 2014.

[I-D.ietf-httpbis-p2-semantics]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", draft-ietf-httpbis-p2-semantics-26 (work in progress), February 2014.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

Appendix A. 2NN Patch and HTTP/2 Server Push

In HTTP/2 [I-D.ietf-httpbis-http2], it is possible to "push" a request/response pair into a client's cache. 2NN (Patch) can be used with this mechanism to perform partial updates on stored responses.

For example, if a cache has this response stored for "http://example.com/list":

```
200 OK
Content-Type: application/json
Cache-Control: max-age=3600
ETag: "aaa"

{ "items": ["a"]}
```

A HTTP/2 server could partially update it by sending the request/response pair (using pseudo-HTTP/1 syntax for purposes of illustration):

```
GET /list
Host: example.com
If-None-Match: "aaa"
Accept-Patch: application/patch+json

2NN Patch
Content-Type: application/patch+json
ETag: "aab"
Patched: "aaa"

[
    { "op": "add", "path": "/items/1", "value": "b" }
]
```

Once the patch is applied, the stored response is now:

```
200 OK
Content-Type: application/json
Cache-Control: max-age=3600
ETag: "aab"

{ "items": ["a", "b"]}
```

Note that this approach requires a server pushing partial responses to know the stored response's ETag, since the client cache will silently ignore the push if it does not match that provided in "Patched". Likewise, clients that are not conformant to this specification will silently drop such pushes, since the status code is not recognised (as per [I-D.ietf-httpbis-p6-cache]).

However, it is possible to do some partial updates without strong consistency. For example, if the stored response is as above, and the server simply wishes to append an value to an array, without regard for the current content of the array (because, presumably,

ordering of its content is not important), it can push:

```
GET /list
Host: example.com
Accept-Patch: application/patch+json

2NN Patch
Content-Type: application/patch+json

[
    { "op": "add", "path": "/items/-", "value": "b" }
]
```

Here, the resulting document would be as above, but since entity tags are not provided, the operation will succeed as long as the patch application succeeds.

Author's Address

Mark Nottingham

Email: mnot@mnot.net
URI: http://www.mnot.net/