Individual Submission
Internet-Draft
Expires: December 3, 2007

L. Dusseault
OSAF
J. Snell
June 2007

# PATCH Method for HTTP
# draft-dusseault-http-patch-08

## Status of this Memo



## Copyright Notice



## Abstract

Several applications extending HTTP require a feature to do partial resource modification. Existing HTTP functionality only allows a complete replacement of a document. This proposal adds a new HTTP method, PATCH, to modify an existing HTTP resource.

Table of Contents

1. Introduction . . . . . 3
2. Mechanisms . . . . . 3
2.1. PATCH Method . . . . . 3
2.1.1. Example . . . . . 5
2.2. Error handling . . . . . 5
2.3. Advertising Support in OPTIONS . . . . . 7
2.3.1. Example . . . . . 7
3. IANA Considerations . . . . . 8
3.1. The 'Accept-Patch' Response Header . . . . . 8
4. Security Considerations . . . . . 8
5. References . . . . . 9
5.1. Normative References . . . . . 9
5.2. Informative References . . . . . 9
Appendix A. Acknowledgements . . . . . 9
Appendix B. Changes . . . . . 9
B.1. Changes from -00 . . . . . 9
B.2. Changes from -01 . . . . . 9
B.3. Changes from -02 . . . . . 10
B.4. Changes from -03 . . . . . 10
B.5. Changes from -04 . . . . . 10
B.6. Changes from -05 . . . . . 10
B.7. Changes from -06 . . . . . 11
B.8. Changes from -07 . . . . . 11
Appendix C. Notes to RFC Editor . . . . . 12
Authors' Addresses . . . . . 12
Intellectual Property and Copyright Statements . . . . . 13

## 1. Introduction

This specification defines the new HTTP 1.1 [RFC2616] method PATCH that is used to apply partial modifications to a HTTP resource.

A new method is necessary to improve interoperability and prevent errors. The PUT method is already defined to overwrite a resource with a complete new body, and can not be reused to do partial changes. Otherwise, proxies and caches and even clients and servers may get confused as to the result of the operation.

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119].

## 2. Mechanisms

### 2.1. PATCH Method

The PATCH method requests that a set of changes described in the request entity be applied to the resource identified by the Request-URI. The set of changes is represented in a format called a "patch document" identified by a media type. PATCH is neither safe or idempotent as defined by [RFC2616] Section 9.1.

The difference between the PUT and PATCH requests is reflected in the way the server processes the enclosed entity to modify the resource identified by the Request-URI. In a PUT request, the enclosed entity is considered to be a modified version of the resource stored on the origin server and the client is requesting that stored version be replaced. With PATCH, however, the enclosed entity contains a set of instructions describing how a resource currently residing on the origin server should be modified to produce a new version. The changes described by the entity MAY result in the creation of one or more new resources on the server, however it is not intended that the body of the PATCH request be used as the content of such resources.

The server MUST always apply the entire set of changes atomically and never provide (e.g. in response to a GET during this operation) a partially-modified representation. If the entire patch document cannot be successfully applied then the server MUST fail the entire request, applying none of the changes. The determination of what constitutes a successful PATCH can vary depending on the patch document and the type of resource being modified. The actual method for determining how to apply the patch document to the resource is defined entirely by the origin server. See Error Handling in section 2.2 for details on status codes and possible error conditions.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable, unless the response includes appropriate Cache-Control or Expires header fields. However, the 303 (See Other) response can be used to direct the user agent to retrieve a cacheable resource.

Collisions from multiple requests are more dangerous than PUT collisions, because a patch document that is not operating from a known base point may corrupt the resource. Clients wishing to apply a patch document to a known entity can first acquire the strong ETag of the resource to be modified, and use that Etag in the If-Match header on the PATCH request to verify that the resource is still unchanged. If a strong ETag is not available for a given resource, the client can use If-Unmodified-Since as a less-reliable safeguard.

It is RECOMMENDED that Servers provide strong ETags for all resources for which PATCH is supported.

A PATCH response with a 2xx status code indicates that the PATCH request was a success. When the server responds with a status code of 200 OK, it MUST include a representation of the modified resource. A 200 response whose entity payload is empty indicates that the result of the PATCH request is an empty resource. If the server chooses not to return a representation of the modified resource, it can use 204 No Content. With either a 200 or 204 response, the server MAY include appropriate entity headers applied to the modified resource to allow the client to verify the success of the operation.

The server MUST NOT ignore any Content-* (e.g. Content-Range) headers that it does not understand or implement and MUST return a 501 (Not Implemented) response in such cases.

If the Request-URI identifies a resource with multiple alternate representations, the server can choose to respond in a variety of ways. For instance, the server can decide which representation to alter and might even be able to change them all consistently depending on the patch format. A particular patch document might be able to identify specific representations to modify or might be capable of describing changes to multiple representations. If the server cannot choose a representation, it can reject the request with an error or the server can choose to redirect the request (e.g. using 301 Moved Permanently or 302 Found), in which case the user agent makes its own decision regarding whether or not to proceed with the request.

Clients are advised to take caution when sending multiple PATCH requests, or sequences of requests that include PATCH, over a

pipelined connection as there are no guarantees that pipelined requests will be processed by the server in the same order in which the client sends them. Such sequences of requests can be made safer by using conditional request mechanisms such as If-Match. See [RFC2616] Section 8.1.2.2 for additional details regarding pipelining and non-idempotent requests.

There is no guarantee that a resource can be modified with PATCH. Further, it is expected that different patch document formats will be appropriate for different types of resources and that no single format will be appropriate for all types of resources. Therefore, there is no single default patch document format that implementations are required to support. Servers MUST ensure that a received patch document is appropriate for the type of resource identified by the Request-URI.

### 2.1.1. Example

Simple PATCH example

```
PATCH /file.txt HTTP/1.1
Host: www.example.com
Content-type: application/example
If-Match: "e0023aa4e"
Content-Length: 100

[description of changes]
```

This example illustrates use of a hypothetical patch document on an existing text file.

Successful PATCH response to existing text file

```
HTTP/1.1 200 OK
ETag: "e0023aa4f"
Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==
Content-Type: text/plain

[modified resource]
```

## 2.2. Error handling

There are several known conditions under which a PATCH request can fail.

Malformed patch document: Can be specified using a 400 Bad Request when the server finds that the patch document provided by the client was improperly formatted. The definition of badly formatted depends on the patch document chosen, but generally if the server finds it cannot handle the patch due to the serialization of the patch document, this response ought to be appropriate.

Unsupported patch document: Specified using a 415 Unsupported Media Type when the client sends a patch document that the server doesn't support for the resource identified by the Request-URI. Such a response SHOULD include an Accept-Patch response header as described in Section 2.3 to notify the client what patch document formats are supported.

Unprocessable request: Can be specified with a 422 Unprocessable Entity [RFC4918] when the server understands the patch document and the syntax of the patch document appears valid, but the server is incapable of processing the request. There are a number of situations that could lead to such a result, for example:

* The client attempted to apply a patch document to an empty resource, but the patch document chosen cannot be applied to an empty resource.
* The client attempted to apply a structural modification and the structures assumed to exist did not exist (e.g. a patch which specifies changing element 'foo' to element 'bar' but element 'foo' doesn't exist).
* The client attempted to modify a resource in a way that would cause the resource to become invalid. For instance, a modification to a well-formed XML document that would cause it to no longer be well-formed.
* The client attempted to modify a resource that has multiple representations but the server was unable to choose which representation to modify.

Conflicting modification: Specified with a 412 Precondition Failed when a client uses either the If-Match or If-Unmodified-Since request headers and attempts to apply a patch document to a resource whose state has changed since the patch was created. If the server detects a possible conflicting modification and neither the If-Match or If-Unmodified-Since request headers are used, the server can return a 409 Conflict response.

Concurrent modification: When a server receives multiple concurrent requests to modify a resource, those requests SHOULD be queued and processed in the order in which they are received. If a server is incapable of queuing concurrent requests, all subsequent requests SHOULD be rejected using a 409 Conflict response until the first modification request is complete.

Other HTTP status codes can also be used under the appropriate circumstances.

The entity body of error responses SHOULD contain enough information to communicate the nature of the error to the client. The content-type of the response entity can vary across implementations.

## 2.3. Advertising Support in OPTIONS

A server can advertise its support for the PATCH method by adding it to the listing of allowed methods in the "Allow" OPTIONS response header defined in HTTP/1.1.

Clients also need to know whether the server supports specific patch document formats, so this specification introduces a new response header "Accept-Patch" used to specify the patch document formats accepted by the server. "Accept-Patch" MUST appear in the OPTIONS response for any resource that supports the use of the PATCH method. The presence of the "Accept-Patch" header in response to any method is an implicit indication that PATCH is allowed on the resource identified by the Request-URI.

Accept-Patch = "Accept-Patch" ":" #( media-range )

The Accept-Patch header specifies a listing of media ranges as defined by [RFC2616], Section 14.1. Note that, unlike the HTTP Accept request header, the Accept-Patch header does not use quality factors.

### 2.3.1. Example

Example: OPTIONS request and response for specific resource

```
[request]

OPTIONS /example/buddies.xml HTTP/1.1
Host: www.example.com

[response]

HTTP/1.1 200 OK
Allow: GET, PUT, POST, OPTIONS, HEAD, TRACE, DELETE, PATCH
Accept-Patch: application/example, text/example
```

The examples show a server that supports PATCH generally using two hypothetical patch documents.

## 3. IANA Considerations

### 3.1. The 'Accept-Patch' Response Header

The 'Accept-Patch' response header should be added to the permanent registry (see [RFC3864]).

Header field name: Accept-Patch

Applicable Protocol: HTTP

Status: standard

Author/Change controller: IETF

Specification document: this specification

## 4. Security Considerations

The security considerations for PATCH are nearly identical to the security considerations for PUT. In addition, one might be concerned that a document that is patched might be more likely to be corrupted, but that concern can be addressed through the use of mechanisms such as conditional requests using ETags and the If-Match request header.

Sometimes an HTTP intermediary might try to detect viruses being sent via HTTP by checking the body of the PUT/POST request or GET response. The PATCH method complicates such watch-keeping because neither the source document nor the patch document might be a virus, yet the result could be. This security consideration is not materially different from those already introduced by byte-range downloads, downloading patch documents, uploading zipped (compressed) files and so on.

Individual patch documents will have their own specific security considerations that will likely vary depending on the types of resources being patched. The considerations for patched binary resources, for instance, will be different than those for patched XML documents.

## 5. References

### 5.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

### 5.2. Informative References

[RFC4918] Dusseault, L., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

## Appendix A. Acknowledgements

PATCH is not a new concept, it first appeared in HTTP in drafts of version 1.1 written by Roy Fielding and Henrik Frystyk.

Thanks to Adam Roach, Chris Sharp, Julian Reschke, Geoff Clemm, Scott Lawrence, Jeffrey Mogul, Roy Fielding, Greg Stein, Jim Luther, Alex Rousskov, Jamie Lokier, Joe Hildebrand, Mark Nottingham and Michael Balloni for review and advice on this document.

## Appendix B. Changes

### B.1. Changes from -00

OPTIONS support: removed "Patch" header definition and used Allow and new "Accept-Patch" headers instead.

Supported delta encodings: removed vcdiff and diffe as these do not have defined MIME types and did not seem to be strongly desired.

PATCH method definition: Clarified cache behavior.

### B.2. Changes from -01

Removed references to XCAP - not yet a RFC.

Fixed use of MIME types (this "fix" now obsolete)

Explained how to use MOVE or COPY in conjunction with PATCH, to create a new resource based on an existing resource in a different location.

### B.3. Changes from -02

Clarified that MOVE and COPY are really independent of PATCH.

Clarified when an ETag must change, and when Last-Modified must be used.

Clarified what server should do if both Content-Type and IM headers appear in PATCH request.

Filled in missing reference to DeltaV and ACL RFCs.

Stopped using 501 Unsupported for unsupported delta encodings.

Clarified what a static resource is.

Refixed use of MIME types for patch formats.

Limited the scope of some restrictions to apply only to usage of required diff format.

### B.4. Changes from -03

Various typographical, terminology consistency, and other minor clarifications or fixes.

### B.5. Changes from -04

Moved paragraphs on ACL and RFC3229 interoperability to new section.

Added security considerations.

Added IANA considerations, registration of new namespace, and discontinued use of "DAV:" namespace for new elements.

Added example of error response.

### B.6. Changes from -05

Due to various concerns it didn't seem likely the application/gdiff registration could go through so switching to vcdiff as required diff format, and to RFC3229's approach to specifying diff formats, including use of the IM header.

Clarified what header server MUST use to return MD5 hash.

Reverted to using 501 Unsupported for unsupported delta encodings.

### B.7. Changes from -06

The reliance on RFC 3229 defined patch documents has been factored out in favor of delta encodings identified by MIME media type.

The required use of DeltaV-based error reporting has been removed in favor of using basic HTTP status codes to report error conditions.

The Accept-Patch response header has been redefined as a listing of media-ranges, similar to the Accept request header.

Added James Snell as a co-author.

### B.8. Changes from -07

Terminology change from "delta encoding" to "patch document"

Added clarification on the safety and idempotency of PATCH

Updated the caching rules of PATCH responses

200 responses MUST include a representation of the modified resource. 204 responses are used to indicate successful response without returning a representation.

Suggest using 422 Unprocessable Entity to indicate that a properly formatted patch document cannot be processed

Clarify the use of 412 and 409 to indicate concurrent and conflicting resource modifications.

Added registration for the Accept-Patch header.

Relaxed the requirements for the use of If-Match and If-Unmodified-Since.

Add language that clarifies the difference between PUT and PATCH.

Add language that clarifies the issues with PATCH and Content Negotiation.

Use of Accept-Patch on any response implies that PATCH is supported.

Add language advising caution when pipelining PATCH requests.

## Appendix C. Notes to RFC Editor

The RFC Editor should remove this section and the Changes section.

## Authors' Addresses

Lisa Dusseault
Open Source Application Foundation
2064 Edgewood Dr.
Palo Alto, CA 94303
US

Email: lisa@osafoundation.org

James M Snell

Phone:
Email: jasnell@gmail.com
URI: http://www.snellspace.com

## Full Copyright Statement



## Intellectual Property



## Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).