Internet Engineering Task Force (IETF)
Request for Comments: 7396
Obsoletes: 7386
Category: Standards Track
ISSN: 2070-1721

P. Hoffman
VPN Consortium
J. Snell

October 2014

JSON Merge Patch

Abstract

This specification defines the JSON merge patch format and processing rules. The merge patch format is primarily intended for use with the HTTP PATCH method as a means of describing a set of modifications to a target resource's content.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7396.

Copyright Notice

Copyright (c) 2014 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 (http://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.

Table of Contents

1. Introduction

This specification defines the JSON merge patch document format, processing rules, and associated MIME media type identifier. The merge patch format is primarily intended for use with the HTTP PATCH method [RFC5789] as a means of describing a set of modifications to a target resource's content.

A JSON merge patch document describes changes to be made to a target JSON document using a syntax that closely mimics the document being modified. Recipients of a merge patch document determine the exact set of changes being requested by comparing the content of the provided patch against the current content of the target document. If the provided merge patch contains members that do not appear within the target, those members are added. If the target does contain the member, the value is replaced. Null values in the merge patch are given special meaning to indicate the removal of existing values in the target.

For example, given the following original JSON document:

```
{
  "a": "b",
  "c": {
    "d": "e",
    "f": "g"
  }
}
```

Changing the value of "a" and removing "f" can be achieved by sending:

```
PATCH /target HTTP/1.1
Host: example.org
Content-Type: application/merge-patch+json

{
  "a":"z",
  "c": {
    "f": null
  }
}
```

When applied to the target resource, the value of the "a" member is replaced with "z" and "f" is removed, leaving the remaining content untouched.

This design means that merge patch documents are suitable for describing modifications to JSON documents that primarily use objects for their structure and do not make use of explicit null values. The merge patch format is not appropriate for all JSON syntaxes.

2. Processing Merge Patch Documents

JSON merge patch documents describe, by example, a set of changes that are to be made to a target resource. Recipients of merge patch documents are responsible for comparing the merge patch with the current content of the target resource to determine the specific set of change operations to be applied to the target.

To apply the merge patch document to a target resource, the system realizes the effect of the following function, described in pseudocode. For this description, the function is called MergePatch, and it takes two arguments: the target resource document and the merge patch document. The Target argument can be any JSON value or undefined. The Patch argument can be any JSON value.

```
define MergePatch(Target, Patch):
  if Patch is an Object:
    if Target is not an Object:
      Target = {} # Ignore the contents and set it to an empty Object
    for each Name/Value pair in Patch:
      if Value is null:
        if Name exists in Target:
          remove the Name/Value pair from Target
      else:
        Target[Name] = MergePatch(Target[Name], Value)
    return Target
  else:
    return Patch
```

There are a few things to note about the function. If the patch is anything other than an object, the result will always be to replace the entire target with the entire patch. Also, it is not possible to patch part of a target that is not an object, such as to replace just some of the values in an array.

The MergePatch operation is defined to operate at the level of data items, not at the level of textual representation. There is no expectation that the MergePatch operation will preserve features at the textual-representation level such as white space, member ordering, number precision beyond what is available in the target's implementation, and so forth. In addition, even if the target implementation allows multiple name/value pairs with the same name, the result of the MergePatch operation on such objects is not defined.

3. Example

Given the following example JSON document:

```
{
  "title": "Goodbye!",
  "author" : {
    "givenName" : "John",
    "familyName" : "Doe"
  },
  "tags":[ "example", "sample" ],
  "content": "This will be unchanged"
}
```

A user agent wishing to change the value of the "title" member from "Goodbye!" to the value "Hello!", add a new "phoneNumber" member, remove the "familyName" member from the "author" object, and replace the "tags" array so that it doesn't include the word "sample" would send the following request:

```
PATCH /my/resource HTTP/1.1
Host: example.org
Content-Type: application/merge-patch+json

{
  "title": "Hello!",
  "phoneNumber": "+01-123-456-7890",
  "author": {
    "familyName": null
  },
  "tags": [ "example" ]
}
```

The resulting JSON document would be:

```
{
  "title": "Hello!",
  "author" : {
    "givenName" : "John"
  },
  "tags": [ "example" ],
  "content": "This will be unchanged",
  "phoneNumber": "+01-123-456-7890"
}
```

4. IANA Considerations

This specification registers the following additional MIME media types:

Type name: application

Subtype name: merge-patch+json

Required parameters: None

Optional parameters: None

Encoding considerations: Resources that use the "application/merge-patch+json" media type are required to conform to the "application/json" media type and are therefore subject to the same encoding considerations specified in Section 8 of [RFC7159].

Security considerations: As defined in this specification

Published specification: This specification.

Applications that use this media type: None currently known.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh file type code(s): TEXT

Person & email address to contact for further information: IESG

Intended usage: COMMON

Restrictions on usage: None

Author: James M. Snell <jasnell@gmail.com>

Change controller: IESG

5. Security Considerations

The "application/merge-patch+json" media type allows user agents to indicate their intention for the server to determine the specific set of change operations to be applied to a target resource. As such, it is the server's responsibility to determine the appropriateness of any given change as well as the user agent's authorization to request such changes. How such determinations are made is considered out of the scope of this specification.

All of the security considerations discussed in Section 5 of [RFC5789] apply to all uses of the HTTP PATCH method with the "application/merge-patch+json" media type.

6. References

6.1. Normative References

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

6.2. Informative References

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010, <http://www.rfc-editor.org/info/rfc5789>.

Appendix A. Example Test Cases

```
ORIGINAL        PATCH            RESULT
-----------------------------------------
{"a":"b"}       {"a":"c"}       {"a":"c"}

{"a":"b"}       {"b":"c"}       {"a":"b",
                                 "b":"c"}

{"a":"b"}       {"a":null}      {}

{"a":"b",       {"a":null}      {"b":"c"}
 "b":"c"}

{"a":["b"]}     {"a":"c"}       {"a":"c"}

{"a":"c"}       {"a":["b"]}     {"a":["b"]}

{"a": {         {"a": {         {"a": {
  "b": "c"}       "b": "d",       "b": "d"
}                 "c": null}      }
                }               }

{"a": [         {"a": [1]}      {"a": [1]}
  {"b":"c"}
 ]
}

["a","b"]       ["c","d"]       ["c","d"]

{"a":"b"}       ["c"]           ["c"]

{"a":"foo"}     null            null

{"a":"foo"}     "bar"           "bar"

{"e":null}      {"a":1}         {"e":null,
                                 "a":1}

[1,2]           {"a":"b",       {"a":"b"}
                 "c":null}

{}              {"a":            {"a":
                 {"bb":           {"bb":
                  {"ccc":          {}}}
                   null}}}
```

Acknowledgments

Many people contributed significant ideas to this document. These people include, but are not limited to, James Manger, Matt Miller, Carsten Bormann, Bjoern Hoehrmann, Pete Resnick, and Richard Barnes.

Authors' Addresses

Paul Hoffman
VPN Consortium

EMail: paul.hoffman@vpnc.org

James M. Snell

EMail: jasnell@gmail.com