Applications Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2015

P. Hoffman
VPN Consortium
J. Snell

June 30, 2014

JSON Merge Patch
draft-ietf-appsawg-json-merge-patch-03

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 subset of target resource's content.

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 January 1, 2015.

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 subset of 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.

A JSON merge patch document can only be a JSON array or a JSON object.

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.

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. 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.

The following rules MUST be applied to determine what changes are to be made. Only one of the two rules will be applied.

1. If the roots of either the merge patch or target resource documents are JSON Arrays, the target resource is replaced, in whole, by the merge patch document.

2. If the roots of both the merge patch and the target resource documents are Objects, iterate through each member of merge patch object and determine the following:

   A. If an equivalent member in the target resource is currently undefined, and the given value in the merge patch is not

null, the member/value pair is added to the target. If an equivalent member in the target resource is currently undefined, and the given value in the merge patch is null, no action is taken.

B. If an equivalent member in the target resource is any type, and the given value in the merge patch is null, the existing member is removed.

C. If an equivalent member in the target resource is any type, and the given value in the merge patch is any type other than an Object or null, the existing value is replaced with the given value.

D. If an equivalent member in the target resource is any type other than an Object, and the given value in the merge patch is an Object, the existing value is replaced with the given value.

E. If an equivalent member in the target resource is an Object, and the given value in the merge patch is an Object, then recursively apply Rule #2 to the two objects.

F. Any member currently defined in the target resource that does not explicitly appear within the merge patch remains untouched and unmodified.

It is not an error for the merge patch document to attempt to remove a member from the target resource that does not currently exist. This is because the final modified state of the target will still accurately reflect the merge documents original intent.

Once the set of intended modifications is derived from the merge patch document, the recipient is free to determine the appropriateness of the modification based on it's own understanding of the target resource. If the recipient is unable to apply any individual modification described by the merge patch document, it MUST NOT apply any of the changes and MUST stop processing the modification. For example, if a patch would remove a member of an object that is semantically required by a processor, the processor MUST not apply any of the patches.

3. Example

For 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" from the "author" object, and remove the word sample from the "tags" Array, would send the following request:

```
PATCH /my/resource HTTP/1.1
Host: example.org
Content-Type: application/merge-patch+json; charset="UTF-8"

{
  "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:

[[NOTE: There were a few notes that the charset media type parameter is unacceptable for a +json media type.]]

Type name: application

Subtype name: merge-patch+json

Required parameters: None

Optional parameters: "charset" : Specifies the character set encoding. If not specified, a default of "UTF-8" is assumed.

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 6 [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: James M Snell <jasnell@gmail.com>

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

6. Acknowledgements

Many people contributed significant ideas to this document. These people include, but are not limited to, James Manger, Matt Miller, Carsten Bormann, and Bjoern Hoehrmann. [[NOTE: If you contributed and we missed your name, please let us know.]]

7. Normative References

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

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

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

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"]

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

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

   {"a":"foo"}     null            Invalid Patch

   {"a":"foo"}     "bar"           Invalid Patch
```

Appendix B. Example JavaScript Implementation

[[NOTE: This example refers to a previous version of this draft. It needs to be updated to reflect the current draft.]]

The following example implementation is provided as is, without warranty. It is provided in the public domain. Note that this

example is provided strictly for illustrative purposes and has not been optimized for performance or reliability in any way.

```
// Apply the patch to the original, return the
// modified object... this will mutate the
// passed in object in place as well...
function apply(orig, patch) {
  if (patch == null)
    return orig;
  else if (patch instanceof Array)
    orig = purge_nulls(patch);
  else if (is_primitive(patch))
    orig = patch;
  else if (patch instanceof Object) {
    if (orig instanceof Array) {
      orig = purge_nulls(patch);
    } else {
      for (m in patch) {
        if (orig.hasOwnProperty(m)) {
          if (patch[m] == null)
            delete orig[m];
          else {
            if (is_primitive(patch[m]))
              orig[m] = patch[m];
            else {
              if (orig[m] instanceof Array)
                orig[m] = purge_nulls(patch[m]);
              else
                orig[m] = apply(orig[m],patch[m]);
            }
          }
        } else if (patch[m] != null)
          orig[m] = purge_nulls(patch[m]);
      }
    }
  }
  return orig;
}

function is_primitive(val) {
  var m = typeof val;
  return m == 'string'  ||
         m == 'number'  ||
         m == 'boolean';
}

function purge_nulls(obj) {
  var ret = obj;
```

```
    if (!is_primitive(obj)) {
      if (obj instanceof Array) {
        var ret = [];
        for (m in obj)
          if (obj[m] != null)
            ret.push(purge_nulls(obj[m]));
      } else if (obj instanceof Object) {
        var ret = {};
        for (m in obj) {
          if (obj[m])
            ret[m] = purge_nulls(obj[m]);
        }
      }
    }
    return ret;
  }

  // Define the original object...
  var orig = {
    "a": "b",
    "c": {
      "d": [1,2,3],
      "e": {
        "f": 1
      }
    }
  }

  // Define the patch...
  var patch = {
    "c": {
      "d": [1,2],
      "e": {
        "f": null
      }
    }
  }

  // Apply the patch...
  var modified = apply(orig,patch);
```

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

James M Snell

Email: jasnell@gmail.com