Applications Area Working Group   J.M. Snell
Internet-Draft   May 20, 2013
Intended status: Standards Track
Expires: November 21, 2013

The application/json-merge-patch Media Type
draft-ietf-appsawg-json-merge-patch-00

Abstract

This specification defines the application/json-merge-patch media type and it's intended use with the HTTP PATCH method defined by RFC 5789.

Status of This Memo

Copyright Notice

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.

For example, given the following original JSON document:

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

A change to the value of the "a" member can be described simply as:

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

{"a": "z"}
```

When applied to the target resource, only the value of the "a" member will be modified, leaving the remaining content untouched.

The Merge Patch format generally supports two types of changes: removing and setting JSON object members. JSON arrays are treated the same as JSON primitives: the whole value can be replaced, but not partially modified. The JSON null value is given a special meaning to indicate the removal of an existing value. These constraints allow Merge Patch to use a format that closely mimics the document being modified. The constraints mean Merge Patch is suitable for patching 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. The "application/json-merge-patch" Media Type

The "application/json-merge-patch" Media Type is used to identify JSON documents that describe, by example, a set of changes that are to be made to a target resource. When used within an HTTP PATCH request, it is the responsibility of the server receiving and processing the request to inspect the payload entity and determine the specific set of operations that are to be performed to modify the target resource.

For example, given the following example JSON document:

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

If the intent is to change the value of the "title" member to from "Goodbye!" to the value "Hello!", add a new "phoneNumber" member, remove the "familyName" member from the "author" object, and remove the word sample from the "tags" Array, the user-agent would send the following request:

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

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

Upon receiving the request, the server is responsible for inspecting the payload and determining, based on it's own understanding of the target resource media type and the underlying data model of the target resource, what specific operations will be applied to modify the resource.

A server receiving this patch request MUST apply the following rules to determine the specific set of change operations to be performed:

1. If either the root of the JSON data provided in the payload or the root of the target resource are JSON Arrays, the target resource is to be replaced, in whole, by the provided data. Any object member or array element contained in the provided data whose value is explicitly null is to be treated as if the member is undefined and MUST NOT appear within the result.

2. If the root of the JSON data provided in the payload is an Object, for each distinct member specified within that object:

   * If the member is currently undefined within the target resource, and the given value is not null, the member and the value are to be added to the target.

   * If the value is explicitly set to null and that member is currently defined within the target resource, the existing member is removed.

   * If the value is either a non-null JSON primitive or an Array and that member is currently defined within the target resource, the existing value for that member is to be replaced with that provided.

   * If the value is a JSON object and that member is currently defined for the target resource and the existing value is a

JSON primitive or Array, the existing value is to be replaced in whole by the object provided. Any object member or array element contained in the provided data whose value is explicitly null is to be treated as if the member was undefined and MUST NOT appear within the result.

* If the value is a JSON object and that member is currently defined within the target resource and the existing value is also a JSON object, then recursively apply Rule #2 to each object.

* Any member currently defined within the target resource that does not explicitly appear within the patch is to remain untouched and unmodified.

Applying these rules to the previous example, the set of specific change operations derived from the request are:

o Change the existing value of the "title" member from "Goodbye!" to "Hello!",

o Add the "phoneNumber" member with a value of "+01-123-456-7890",

o Remove the "familyName" member from the current object value associated with the "author" member, and

o Change the existing value of the "tags" member from ["example","sample"] to ["example"].

The resulting JSON document would be similar to the following (the specific ordering of members within JSON documents is insigificant):

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

Once the set of intended modifications is derived from the request, the server is free to determine the appropriateness of the modification based on it's own understanding of the target resource. For instance, in the previous example, it is possible that the "familyName" member might be required within the target resource and

cannot be removed. Note that in such cases, per [RFC5789], Section 2, the server is REQUIRED to reject the entire PATCH request using an HTTP error response code appropriate to the error condition.

If the request attempts to remove a member from the target resource that does not currently exist, the server SHOULD NOT consider the request to be in error. The requested removal operation is simply to be ignored by the server as the final modified state of the target resource will still accurately reflect the user-agent's original intent.

3. IANA Considerations

This specification registers the following additional MIME Media Types:

Type name: application

Subtype name: json-merge-patch

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/json-merge-patch" 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 [RFC4627].

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

4. Security Considerations

The "application/json-merge-patch" 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/json-merge-patch" Media Type.

5. Normative References

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

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

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

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}

     {"a":"foo"}     {"b": [         {"a":"foo",
                       3,             "b": [3, {}]
                       null,         }
                       {"x":null}
                      ]
                     }

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

Appendix B. Example JavaScript Implementation

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

```
   // 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);
```

Author's Address

James M Snell

Email: jasnell@gmail.com