RFC 7396 : JSON Merge Patch

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 novembre 2014

Date de publication du RFC : Octobre 2014

https://www.bortzmeyer.org/7396.html

La commande HTTP `PATCH` permet d'envoyer à un serveur HTTP un document représentant les différences entre l'état d'une ressource sur le serveur et l'état souhaité par le client HTTP. Cette liste de différences peut prendre plusieurs formats et ce RFC en spécifie un nouveau, "*JSON merge patch*", un format de "*patch*" conçu pour JSON.

À noter que ce RFC remplace le RFC 7386 [1], publié deux semaines avant, mais qui comportait une erreur technique.

Le format normalisé par ce RFC n'est en fait pas spécifique à HTTP et pourra même servir avec d'autres protocoles. Mais la commande `PATCH` du RFC 5789 sera sans doute sa principale utilisation. Une ressource au format "*JSON merge patch*" est un objet JSON elle-même. Pour chaque clé, si la ressource cible a également cette clé, la valeur est remplacée par celle du "*patch*". Si elle ne l'a pas, on ajoute le couple clé/valeur. Et si la valeur dans le "*patch*" est `null`, on supprime le couple clé/valeur de la ressource cible. Ainsi, si on a ce document JSON sur le serveur :

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

et qu'on envoie le "*patch*" suivant en HTTP (notez le type MIME de notre nouveau format) :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7386.txt`

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

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

On obtiendra :

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

(La valeur de `a` a été changée, et le couple indexé par `c / f` a été supprimé.)

Ce format, centré sur le résultat plutôt que sur les opérations, suit donc des principes assez différents de son homologue XML du RFC 5261.

On notera donc que tous les contenus JSON ne sont pas "*patchables*" ou, en tout cas, pas de manière propre et facile, avec ce format. Par exemple, si des `null` sont effectivement utilisés, ou bien si la structure du texte JSON n'est pas celle d'un objet. Mais, bon, ce format est très simple, est lui-même en JSON, et le RFC est très court et facile à comprendre (ce qui n'aurait pas été le cas si on avait voulu tout prévoir), on ne peut pas tout avoir.

La section 2 du RFC précise les règles à suivre lors du traitement des "*patches*". Elle est rédigée en pseudo-code (c'est une erreur dans l'indentation de ce pseudo-code qui avait rendu nécessaire le remplacement du RFC 7386 par notre RFC) et est assez simple pour être citée ici :

```
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
```

Parmi les points à noter, le fait qu'un "*patch*" qui n'est pas un objet JSON (par exemple un tableau) va toujours remplacer l'intégralité de la ressource cible, ou le fait qu'on ne peut pas modifier une partie d'une ressource cible qui n'est pas elle-même un objet (il faut la changer complètement).

Le "*patch*" va agir sur les valeurs, pas sur leur représentation. Ainsi, on n'a aucune garantie qu'il préservera l'indentation du texte JSON ou la précision des nombres. De même, si la ressource cible tire profit des faiblesses de la norme JSON, elle peut ne pas sortir intacte : par exemple, si la ressource cible a plusieurs membres qui ont la même clé (ce qui n'est pas formellement interdit en JSON mais donne des résultats imprévisibles).

Un exemple plus détaillé de "*patch*" JSON se trouve en section 3. On part de ce document :

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

Et on veut changer le titre, ajouter un numéro de téléphone, retirer le nom de famille de l'auteur, et retirer l'élément `sample` du tableau `tags`. On envoie cette requête :

```
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" ]
}
```

Et on obtient ce document :

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

Notez qu'il a fallu remplacer complètement le tableau `tags` : il n'y a pas de mécanisme pour retirer juste un élément du tableau (comme expliqué au début). Des tas d'exemples figurent dans l'annexe A, si vous voulez écrire une suite de tests.

Le type MIME `application/merge-patch+json` figure désormais dans le registre IANA <`https://www.iana.org/assignments/media-types/media-types.xhtml#application`> (cf. section 4 du RFC), aux côtés de l'équivalent XML décrit dans le RFC 7351, `application/xml-patch+xml`.