RFC 7493 : The I-JSON Message Format

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2015

Date de publication du RFC : Mars 2015

https://www.bortzmeyer.org/7493.html

Le format JSON, normalisé dans le RFC 8259 [1], est aujourd'hui très répandu dans l'Internet et largement utilisé. Il a pourtant des faiblesses, notamment une définition un peu floue sur certains points, qui laisse planer un doute sur l'interopérabilité : si j'envoie dans le fichier un objet JSON où deux membres portent le même nom, le destinataire pourra-t-il le traiter ? Ce nouveau RFC résout le problème en créant un profil, une restriction de JSON, nommée I-JSON, qui n'a pas ces flous et garantit donc normalement l'interopérabilité.

On peut penser qu'il aurait mieux valu préciser la norme JSON (aujourd'hui dans le RFC 8259) pour éliminer les ambiguités. Mais cela n'est pas possible sans casser la compatibilité. Si, par exemple, on interdisait dans la norme les objets dont deux membres ont le même nom, on rendrait illégaux des textes JSON existants, qui profitaient du flou de la norme. Cela a été jugé inacceptable, d'où ce choix d'un profil de JSON. Ceux qui veulent continuer à profiter du vague de la norme restent au JSON traditionnel, ceux qui veulent le maximum d'interopérabilité envoient désormais uniquement du I-JSON (ce qui veut dire « Internet JSON » et qui, curieusement, n'a pas de type MIME à lui, on continue à utiliser `application/json`). Ne pas casser l'existant était d'autant plus nécessaire que bien des analyseurs JSON mettent directement en correspondance les objets JSON avec des structures de données équivalentes dans leur langage de programmation, et qu'il ne fallait donc pas leur imposer une vérification préalable.

Donc, définition du format (section 2). Un texte I-JSON est un texte JSON, puisque I-JSON est un profil, un sous-ensemble de JSON. Il est forcément encodé en UTF-8 et ne doit pas utiliser les "*surrogates*" (seizets d'indirection) d'Unicode, ni des points de code qui sont des non-caractères <http://www.unicode.org/faq/private_use.html#nonchar1>.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8259.txt

Il y a deux grands problèmes d'interopérabilité avec JSON : la représentation des nombres (voir la section 6 du RFC 8259), et les noms de membres dupliqués dans les objets (cf. la section 4 du RFC 8259). La section 2.2 traite les nombres : un texte I-JSON ne devrait pas (sauf raison vitale) utiliser des nombres flottants qui ne sont pas représentables en IEEE 754-2008 binary64, soit en raison de leur taille (1E400...), soit à cause de leur précision (3.14159265358979323846264338327 9). Pour les entiers, il ne faut pas espérer que les nombres en dehors de l'intervalle [-(2**53)+1 , (2**53)-1] soient acceptés.

Pour les noms de membres, la section 2.3 est claire : les noms dupliqués sont interdits. Donc, cet objet est du JSON légal, quoique dangereux, alors qu'il n'est pas du I-JSON, en raison du `Title` dupliqué :

```
"Image": {
            "Width":  800,
            "Height": 600,
            "Title": "View from 15th Floor",
            "Title": "Nice view"
}
```

D'autre part, I-JSON dit clairement que l'ordre des membres d'un objet ne compte pas. Ces deux objets sont identiques :

```
"Image": {
            "Width":  800,
            "Height": 600}

"Image": {
            "Height": 600,
            "Width":  800}
```

La section 3 de notre RFC précise bien qu'une mise en œuvre de I-JSON est tout à fait autorisée à rejeter ou ignorer un texte JSON qui n'est pas du bel I-JSON bien propre. Les protocoles Internet qui utilisent I-JSON sont invités à prévoir un moyen de signaler ce genre d'erreurs à l'envoyeur.

La section 4 du RFC couvre quelques choix supplémentaires. D'abord, comme dans l'ancien RFC 4627, et de manière différente de ce que permet le RFC 8259, un texte I-JSON doit avoir un objet ou un tableau au niveau supérieur. Ce texte est du JSON valable (depuis le RFC 7159) mais pas du I-JSON :

```
"Ma belle chaîne n'est pas acceptée car elle n'est pas objet ou tableau"
```

Il est recommandé que les protocoles qui utilisent I-JSON précisent que le type de plus haut niveau d'un texte soit un objet et que les membres inconnus de cet objet soient ignorés. Il existe en effet deux politiques dans les protocoles pour traiter les éléments inconnus : "*Must-Ignore*" et "*Must-Understand*". La première indique que le receveur d'un message doit en ignorer les membres qu'il ne connait pas et continuer comme s'ils étaient absents. La deuxième indique qu'il faut connaitre tous les membres présents, et donc rejeter un message comportant des membres inconnus. La politique "*Must-Ignore*" permet l'extensibilité du protocole : une nouvelle version permet d'ajouter des membres à un objet, sans craindre que les anciens récepteurs ne plantent.

I-JSON ajoute aussi des contraintes aux types de données de base. Ainsi, il impose que les dates et heures soient au format du RFC 3339, avec en outre uniquement des lettres en majuscules et des secondes indiquées explicitement :

```
"creation": "2015-03-23T09:13:00Z"
```

Et une dernière contrainte sur les données, le binaire devra être encodé en base64url (section 5 du RFC 4648). Ce point fut l'un des plus chaudement disputés dans le groupe de travail.

Je n'ai pas trouvé d'outil qui testerait qu'un texte JSON est conforme à cette nouvelle norme I-JSON. Si quelqu'un connait...

I-JSON est expliqué par l'auteur du RFC sur son blog <https://www.tbray.org/ongoing/When/201x/2015/03/23/i-json>.