RFC 5137 : ASCII Escaping of Unicode Characters

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 24 février 2008

Date de publication du RFC : Février 2008

https://www.bortzmeyer.org/5137.html

Idéalement, aujourd'hui, tous les protocoles Internet devraient être complètement **internationalisés** et notamment devraient accepter que les données soient exprimées dans un des encodages classiques d'Unicode, comme UTF-8. Mais ce n'est pas encore le cas. Ce RFC propose une approche pour utiliser des caractères Unicode dans les derniers protocoles dinosauriens qui n'acceptent normalement que l'ASCII.

La section 1.1 explique bien que ce RFC ne s'applique pas aux protocoles qui acceptent déjà un ou plusieurs encodages Unicode, ce qu'ils feront tous dans le futur, espérons-le. Mais, en attendant ce jour, ce RFC s'applique aux vieux protocoles. Actuellement, ceux-ci acceptent presque tous une forme d'**échappement** qui leur permet de transporter plus ou moins d'Unicode. L'idée est de mettre un peu d'ordre dans ces formes, sans aller jusqu'à les standardiser complètement (ce que tentaient de faire les premières versions de l'"*Internet-Draft*" qui a donné naissance à ce RFC).

Le cas de textes parlant de caractères Unicode (« Les citations en français doivent commencer par un demi-cadratin, le U+2013 ») n'est pas non plus couvert : de tels textes, conçus pour les humains et non pour les programmes, doivent utiliser la notation Unicode traditionnelle U+nnnn comme ci-dessus (section 3).

Notre RFC recommande donc finalement deux formes pour les caractères Unicode dans les fichiers conçus pour être lus par des programmes. En prenant l'exemple du [Caractère Unicode non montré [1]] cyrillique (U+0418, grand I), les deux formes recommandées sont :

— `\u'0418'` (section 5.1), une forme qui a été conçue spécialement pour ce RFC,
— `&#x418;` (section 5.2), la forme traditionnelles de XML et HTML.

1. Car trop difficile à faire afficher par LaTeX

Pourquoi ces deux formes ? Comme l'explique la section 2, il existe de nombreuses solutions. Un premier principe qui a guidé celle choisie est qu'il fallait encoder des **points de code** Unicode (les index dans la table Unicode comme le 418 hexadécimal ci-dessus) et surtout pas des octets d'un encodage particulier comme le font, malheureusement, les URI (section 2.1 du RFC 3986 [2]).

Un deuxième principe est d'utiliser des délimiteurs explicites, pour éviter toute ambiguïté, sans forcer les caractères à être représentés par un nombre fixe de chiffres (section 4).

Plusieurs formes couramment recontrées dans la nature sont discutées dans le RFC et rejetées (section 6 et annexe A) :

— Celle de Java, `\u0418` qui viole le premier principe (c'est un sur-encodage d'UTF-16, ce qui n'est pas gênant pour notre grand I cyrillique, qui fait partie du Plan Multilingue de Base d'Unicode, mais complique les choses pour les caractères en dehors de ce plan),
— Celle de C, `\u0418`, mais qui utilise un grand U et huit chiffres pour les caractères en dehors du Plan Multilingue de Base, une astuce bien dans la ligne de ce langage,
— Celle de Perl, `\x{418}`, qui avait personnellement ma préférence, notamment par ses délimiteurs symétriques, mais qui a été écartée car certains craignaient une ambiguïté possible due à la possibilité en Perl d'utiliser d'autres encodages.

Enfin, on notera que le RFC ne spécifie pas d'échappement standard si on veut écrire, par exemple `\u'0418'` sans qu'il soit interprété comme un caractère Unicode. Le truc classique du `\\` est recommandé mais pas imposé.

2. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3986.txt`