RFC 5895 : Mapping Characters in IDNA2008

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 septembre 2010

Date de publication du RFC : Septembre 2010

https://www.bortzmeyer.org/5895.html

Dans l'ancienne version de la norme IDN, en 2003, une étape obligatoire et uniforme de **canonicalisation** (ou **normalisation**) précédait le traitement des noms de domaine en Unicode. La nouvelle version d'IDN <https://www.bortzmeyer.org/idnabis.html>, sortie en 2010, a supprimé cette étape. Désormais, chaque application est libre de canonicaliser comme elle le souhaite les noms entrés par l'utilisateur. Il n'est toutefois pas interdit de donner des conseils aux programmeurs mais il n'a pas été possible, dans le groupe de travail idnabis <http://tools.ietf.org/wg/idnabis>, de trouver un consensus sur ces conseils. Il y aura donc un ou plusieurs RFC « pour information seulement » sur ce sujet. Celui-ci est le premier.

La section 1 du RFC rappelle ce problème de normalisation. Celle-ci se produit uniquement dans les applications, et n'a pas d'influence sur ce qu'on voit dans le réseau. Sur le câble, on voit passer uniquement des noms « punycodés » comme xn--acadmie-franaise-npb1a.fr. La norme IDNA bis (RFC 5890 [1] et suivants) expose comment traduire un nom de domaine en Unicode (comme académie-française.fr) en cette forme punycodée (et retour). Le RFC 5891 impose que le nom Unicode soit déjà en forme normale C et n'accepte qu'un nombre restreint de caractères (par exemple les majuscules sont exclues). Imaginons qu'un utilisateur francophone travaille dans un environnnement Latin-1 et saisisse Académie-Française.fr. Ce nom est illégal pour IDNA bis et, pire, il n'est pas en Unicode. Il va donc falloir le traduire en Unicode, et le normaliser. Ici, c'est trivial, on remplace les octets de Latin-1 par ceux d'un encodage Unicode et on remplace chaque majuscule par sa minuscule équivalente. On a fait une **normalisation**. Mais, dans certains cas, l'opération est bien plus complexe. En français, il est difficile de trouver un exemple significatif. Mais d'autres langues posent des défis plus sérieux. Un exemple classique est celui du turc où la minuscule du grand I sans point (I) est le petit i sans point ([Caractère Unicode non montré [2]]) pas le petit i avec point (i). Mettre I en minuscule dépend de la "*locale*".

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5890.txt
2. Car trop difficile à faire afficher par LaTeX

Il est donc impossible de trouver une normalisation qui couvre proprement tous les cas, ne serait-ce que parce qu'elle dépend de la langue (alors que les RFC définissent des normes mondiales). Le choix de IDNA bis avait donc été d'abandonner le principe d'une normalisation standard (qui était définie dans le RFC 3491). Il y a donc désormais plusieurs normalisations possibles. Toutes devraient suivre un principe simple « surprendre l'utilisateur le moins possible ». Plus facile à dire qu'à faire... Notre RFC 5895 précise d'ailleurs modestement qu'il ne donne pas une solution complète. Mais, au moins, la normalisation est laissée au logiciel qui est le mieux placé pour connaître le contexte, le logiciel qui est directement en contact avec l'utilisateur.

À propos de la différence entre interface utilisateur et protocole réseau, la section 1.1 contient une intéressante discussion à ce sujet. Normalement, l'IETF ne s'occupe que des protocoles. L'interface utilisateur est certes une chose importante mais elle nécessite des compétences différentes, qui ne sont pas forcément très présentes à l'IETF. Résultat, beaucoup de programmeurs d'applications réseau sous-estiment la difficulté de réaliser une bonne interface utilisateur. Ainsi, des phrases qu'on trouve dans des normes IDN comme « l'utilisateur rentre un nom de domaine en Unicode » expédie en moins d'une ligne un processus très complexe, partant d'une décision de l'utilisateur de choisir tel symbole plutôt que tel autre, puis de le rentrer dans l'ordinateur par des moyens très divers (rien qu'avec un clavier, il existe d'innombrables manières de taper un sinogramme, par exemple ; et le clavier n'est pas le seul périphérique d'entrée), le tout suivi par le processus d'interprétation des entrées par l'ordinateur (très simple pour l'ASCII mais bien plus complexe avec d'autres écritures).

IDNA bis a supprimé complètement la partie « interface utilisateur ». Notre RFC 5895, sans la rétablir dans le protocole, expose des idées pour la gérer. Cette idée est résumée dans la section 1.2 : ce RFC 5895 propose des extensions à IDNA bis pour canonicaliser les noms d'une manière raisonnable dans la plupart des cas. Par définition, la normalisation de ce RFC ne conviendra donc pas à tout le monde, d'où son caractère « pour information » seulement.

La section 2 décrit la procédure de normalisation sous forme d'une suite d'étapes :

— Passer les caractères en minuscule, en utilisant uniquement la base Unicode (propriétés `Lowercase_Mapping` dans le fichier `SpecialCasing.txt` puis `Simple_Lowercase_Mapping` dans la table principale).
— Les caractères « pleine-chasse » et « demi-chasse » sont remplacés par leur équivalent (défini dans la table principale Unicode), le seul qu'accepte IDNA bis. Cette étape est utile car les mécanismes de saisie utilisés en Asie ne permettent pas toujours facilement de saisir la forme qu'attend le RFC 5891.
— Les caractères sont normalisés en NFC. C'est la forme qu'attend IDNA bis. Il est raisonnable que l'application fasse cette normalisation car l'utilisateur n'en a pas toujours la possibilité. Si vous tapez sur la touche « à » de votre clavier, savez-vous si l'application enregistrera U+00E0 (petit a avec accent grave) ou bien U+0061 U+0300 (petit a suivi de l'accent grave combinant) ? NFC ne laissera subsister que la première forme.
— Pour de très bonnes raisons, les règles d'IDNA bis ne s'appliquent qu'à un composant de nom de domaine isolé. Ainsi, dans `www.académie-française.fr`, `académie-française` est traité sans considération pour le composant précédent (`www`) ou pour le suivant (`fr`). Il y a de très bonnes raisons pour cela (même si certains amateurs de régulation auraient voulu faire des règles inter-composants) mais la canonicalisation, qui dispose du nom complet peut ajouter d'autres étapes notamment considérer comme séparateur d'autres caractères que le point de ASCII. Le [Caractère Unicode non montré] (U+3002) est recommandé.

Si j'appliquais cet algorithme, aurais-je une normalisation raisonnable ? La section 3 étudie ce point et répond clairement **non**. L'algorithme décrit en section 2 est uniquement un point de départ pour le programmeur et ne devrait pas être utilisé « sec ». La bonne démarche pour le courageux développeur est donc de bien lire la section 1 pour comprendre l'ampleur du problème, ensuite de partir de l'algorithme de la section 2 et de l'adapter.