RFC 5892 : The Unicode code points and IDNA

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 août 2010

Date de publication du RFC : Août 2010

https://www.bortzmeyer.org/5892.html

Dans l'ensemble <https://www.bortzmeyer.org/idnabis.html> des RFC sur la version 2 des IDN (appelée IDNAbis ou IDNA2008), ce document normalise les propriétés Unicode utilisées pour calculer la catégorie à laquelle appartient un caractère donné, catégorie qui déterminera s'il est autorisé dans un nom de domaine en Unicode. (Il a depuis été partiellement mis à jour par le RFC 8753[1].)

Le concept de catégorie est défini dans la section 1 (avertissement : le vocabulaire de IDNAbis est très flou et on trouve « catégorie » ou « propriété » pour ce concept, des termes d'autant plus malheureux qu'ils existent aussi dans Unicode avec un sens différent ; j'ai donc plutôt utilisé « statut »). Ce RFC 5892 contient les tables complètes de tous les caractères Unicode et de leur catégorie. Mais ces tables ne sont présentes qu'à titre d'information : IDNAbis est en effet indépendant de la version d'Unicode utilisée et la vraie norme est l'algorithme utilisé pour créer les tables, algorithme qu'il faut faire tourner pour chaque version d'Unicode, pour trouver la table effective.

IDNAbis repose sur un principe d'exclusion par défaut. Tout caractère est interdit, sauf s'il est autorisé (cf. RFC 4690). Cette autorisation dépend de ses propriétés Unicode, propriétés qui font partie de la norme Unicode. Par exemple, le U+00E9 (petit e accent aigu) est dans la catégorie Unicode `Ll` (lettres minuscules, presque toutes autorisées).

Le fait d'être dans la bonne catégorie des tables ne suffit pas : dans certains cas, IDNAbis met des contraintes aux **combinaisons** de caractères.

Quelles sont les catégories (ou statuts) possibles ?
— `PROTOCOL VALID` dit `PVALID`, les caractères acceptés.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc8753.txt`

— CONTEXTUAL RULE REQUIRED, pour des caractères acceptés sous condition, par exemple sur leur position dans le composant du nom de domaine. Il est abrégé en CONTEXTJ (caractères qui contrôlent l'attachement de deux mots comme le U+200D, "*Zero-width joiner*") ou CONTEXTO (les autres).
— DISALLOWED, les caractères interdits.
— UNASSIGNED, les points de code pas encore affectés dans la norme Unicode, interdits aujourd'hui mais qui, selon leurs propriétés, pourraient devenir autorisés dans le futur.

Le classement d'un caractère dans une de ces catégories dépend de l'algorithme décrit plus loin. Une liste d'exceptions maintenue manuellement (section 2.6) permet d'ajuster le résultat, une autre liste manuelle permet de maintenir la stabilité (d'empêcher un caractère de changer de catégorie).

La section 2 décrit les catégories utilisées pour classer les caractères (à ne pas confondre avec les catégories IDNA, comme PVALID, décrites plus haut) et les propriétés utilisées (elles ne sont pas mutuellement exclusives) :

— Lettres & Chiffres (section 2.1) : le caractère a une catégorie Unicode dans l'ensemble {Ll, Lu, Lo, Nd, Lm, Mn, Mc}. Par exemple Ll désigne les lettres minuscules, ainsi le êta grec, [Caractère Unicode non montré [2]] (U+0371) est dans cette catégorie (sa définition dans la base Unicode étant 0371;GREEK SMALL LETTER HETA;Ll;0;L;;;;;N;;;0370;;0370).
— Instables (section 2.2) : le caractère ne survit pas à une normalisation NFKC avec un changement de casse. La plupart de ces caractères ne seront pas autorisés.
— Propriétés qu'on peut ignorer (section 2.3) : ces caractères ont des propriétés qui font qu'on peut les ignorer pour les noms de domaines (par exemple les caractères d'espacement, propriété Unicode White_Space).
— Blocs qu'on peut ignorer (section 2.4) : des blocs entiers des tables Unicode peuvent être ignorés comme le bloc des symboles utilisés en musique qui va de U+1D100 ([Caractère Unicode non montré]) à U+1D1FF.
— LDH ("*Letters-Digits-Hyphens*"), les caractères ASCII traditionnellement utilisés dans les noms de machine (section 2.5).
— Des exceptions, définies manuellement, pour le cas où les propriétés Unicode ne donnent pas le bon résultat (section 2.6). Ainsi, après de **très** longues discussions dans le groupe de travail, le [Caractère Unicode non montré] allemand, le [Caractère Unicode non montré] grec et le [Caractère Unicode non montré] tibétain (ce dernier, le "*tsheg*", a été l'objet du débat le plus aveugle puisqu'aucun expert de cette écriture n'était présent) ont été manuellement autorisés (alors qu'ils auraient été interdits en appliquant l'algorithme). Par contre, les chiffres indo-arabes, qui auraient été autorisés inconditionnellement, sont maintenant autorisés uniquement dans certains contextes (voir le RFC 5564 pour une discussion sur ces chiffres).
— Compatibilité (section 2.7) : cette catégorie est actuellement vide. Mais, dans les futures versions d'Unicode, des changements des propriétés pourraient faire passer des caractères de PVALID à DISALLOWED ou le contraire. Ils seront alors mis dans cette catégorie pour conserver leur ancien statut.
— Contrôle du collage entre caractères (section 2.8). Cette catégorie regroupe les caractères « gluons » qui servent à coller des caractères ou des mots qui seraient normalement séparés. Certaines écritures (par exemple les indiennes) en font un grand usage.
— Vieil Hangul (section 2.9), une catégorie très "*ad hoc*" pour des caractères utilisés en Corée.
— Et enfin, la dernière catégorie, celle des caractères non actuellement affectés (section 2.10) mais qui pourraient l'être dans les futures versions d'Unicode.

Bien, on a dix catégories. Comment les utilise t-on pour déterminer si un caractère est acceptable ou pas ? C'est l'objet de la section 3, qui indique cet algorithme en pseudo-code. Je n'en donne qu'une partie ici :

2. Car trop difficile à faire afficher par LaTeX

```
SI le caractère est dans les Exceptions, sa propriété IDNA est donnée
par la table Exceptions
...
SINON SI le caractère est dans LDH, alors il est PVALID
...
SINON SI le caractère est dans les Blocs Ignorables alors il est
DISALLOWED
...
SINON SI le caractère est dans les Lettres & Chiffres, alors il
est PVALID

SINON SI (cas par défaut) il est DISALLOWED
```

La section 4 insiste sur le fait que la liste des caractères faisant foi est celle calculée par l'algorithme ci-dessus. La liste fournie dans ce RFC 5892, en annexe B, n'est là que pour information (en effet, chaque nouvelle version d'Unicode modifiera les tables).

On a vu que le sort d'un caractère dont le statut est `CONTEXT` nécessite de regarder une table. Celle-ci est définie dans la section 5.2 et sa syntaxe figure dans l'annexe A. Elle est hébergée à l'IANA `<https://www.iana.org/assignments/idna-tables/idna-tables.xml#idna-tables-context>`.

Enfin, même si elle n'a pas de caractère normatif (cf. RFC 8753), la plus grande partie du RFC est formée par l'annexe B, qui donne l'état actuel des tables de caractères avec leur statut.

Comme indiqué plus haut, les tables figurant dans le RFC ne sont pas normatives, seul l'algorithme l'est. En pratique, les tables ont été produites par un programme écrit par l'auteur du RFC qui ne le distribue pas (malgré plusieurs demandes). J'ai refait une mise en œuvre **incomplète** (manque de temps) de l'algorithme qu'on peut récupérer en (en ligne sur `https://www.bortzmeyer.org/files/create-idnabis-tables.py`).