RFC 7940 : Representing Label Generation Rulesets Using XML

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 août 2016

Date de publication du RFC : Août 2016

https://www.bortzmeyer.org/7940.html

Lorsqu'on a des identificateurs en Unicode (par exemple pour les noms de domaine internationalisés), on souhaite souvent limiter le nombre de caractères autorisés, et avoir des possibilités de variante (par exemple, dire que deux chaînes de caractères Unicode sont en fait équivalentes sémantiquement). Il faut donc rédiger ces règles, et un langage formel est certainement utile pour cela. C'est ce que propose ce nouveau RFC, qui normalise un langage XML pour décrire les règles d'enregistrement de ces identificateurs Unicode.

Le terme utilisé par le RFC est LGR, pour "*Label Generation Ruleset*". "*Label*" est le composant de base d'un nom de domaine, et "*Ruleset*" fait référence à l'ensemble des règles décrites en XML et qui permettront de savoir si un composant est valide, et quelles transformations éventuellement effectuer avant de l'enregistrer. C'est un concept qui vient de l'ICANN, qui a toujours présenté les IDN comme une chose dangereuse (bien à tort <https://www.bortzmeyer.org/idn-et-phishing.html>), nécessitant des "*variant tables*" ou autres algorithmes compliqués. Donc, si vous ne travaillez pas avec l'ICANN, il n'est pas nécessaire de connaitre ce RFC.

Le langage présenté dans ce RFC est assez complexe, mais le problème est complexe. Ceci dit, dans les cas « simples » (alphabet latin), les règles LGR seront également simples. Si on veut juste dire que é, è et ë sont autorisés dans les noms, et avec e comme variante, c'est trivial à faire, à peine plus compliqué que les tables du RFC 4290 [1].

Notez que l'algorithme décrit par un LGR ne suffit pas comme règle d'enregistrements des noms de domaine : il existe d'autres règles (comme la priorité aux entreprises titulaires de propriété intellectuelle) qui ne sont pas exprimables par ces algorithmes.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4290.txt

Avant ce RFC, les règles concernant les IDN étaient publiées en langue naturelle, ou parfois dans le langage formel des RFC 3743 et/ou RFC 4290. Le langage décrit ici a vocation à les remplacer. Il est structuré, donc lisible par un programme, et permet de dériver automatiquement les programmes de vérification d'un nom (par exemple ceux qui sont lancés par un serveur EPP recevant un ordre `<domain:create>`, RFC 5731, section 3.2.1).

Le format est relativement simple si on a une politique simple mais permet le cas échéant de mettre en œuvre des politiques très complexes (qui sont utiles pour certaines écritures asiatiques, par exemple). Il est conçu pour les IDN mais peut être utilisé si on a à enregistrer des identificateurs Unicode dans un autre contexte.

Le cahier des charges du format des LGR figure en section 2 du RFC :
— Doit pouvoir être programmé de manière raisonnablement directe (je n'ai pas essayé...)
— Doit pouvoir être validé (l'ICANN, qui exige l'envoi d'une politique IDN pour les TLD qu'elle régule, peut ainsi tester automatiquement la validité de celles-ci). Notez aussi qu'un LGR sera développé `<https://www.icann.org/en/resources/idn/draft-lgr-procedure-07dec12-en.pdf>` pour la racine du DNS.
— Doit pouvoir exprimer la notion de variantes (équivalence entre deux caractères, ou chaînes de caractère, par exemple entre les sinogrammes traditionnels et les simplifiés).
— Doit pouvoir exprimer la notion de contexte (un caractère n'est valable que dans un certain contexte par exemple le sigma final grec, U+03F1 ([Caractère Unicode non montré [2]]) qui ne peut apparaître qu'en fin de mot).
— Etc

Un point important de ce RFC est qu'il décrit **un format, pas une politique** : le RFC ne dit pas si le caractère Unicode U+00E8 (è) doit être considéré comme une variante du U+0065 (e) : il fournit juste le moyen d'exprimer cette équivalence, si on le décide.

La section 4 du RFC spécifie le format exact. (Le schéma formel - en Relax NG - est dans l'annexe D.) Un ensemble de règles LGR est mise dans un élément XML `<lgr>`. Il contient quelques métadonnées, dont la date et la langue ou l'écriture pour lesquels ce LGR est prévu. Comme l'ICANN, dans ses politiques IDN, mélange très souvent langue et écriture (par exemple en parlant de « domaines multilingues », ce qui est absurde), il faut préciser que la plupart des politiques portent sur des écritures (un nom de domaine n'a pas de langue : `coca-cola.com` est-il de l'anglais ?) La valeur de l'élément XML `<language>` est une étiquette du RFC 5646, avec la langue marquée comme « non définie » (`und` pour "*undeterminate*"). Un exemple pour une politique consacrée à l'alphabet cyrillique : `<language>und-Cyrl</langua`

Les règles elle-mêmes sont en section 5. La plus simple est celle qui autorise un caractère donné, ici le tiret (`cp` = "*Code Point*" Unicode, ici U+002D) :

```
<char cp="002D"/>
```

On peut aussi autoriser un intervalle, ici tous les chiffres arabes :

```
<range first-cp="0030" last-cp="0039"/>
```

2. Car trop difficile à faire afficher par LaTeX

Comme on le voit, les caractères sont identifiés par leur point de code Unicode, écrit sans le « U+ » initial. Bien que XML permettre de représenter nativement tous les caractères Unicode, cette possibilité n'est pas exploitée par ce RFC (cf. Unicode Standard Annex #42 <http://unicode.org/reports/tr42/>).

Voici maintenant un exemple de règle qui précise que le point médian U+00B7 n'est autorisé qu'entre deux l (c'est le cas en catalan mais notez qu'en français, il est aussi utilisé dans une perspective féministe). Cela se fait en indiquant une séquence :

```
<char cp="006C 00B7 006C" comment="Catalan middle dot"/>
```

Et voici une règle contextuelle plus complexe. On veut n'autoriser le gluon de largeur nulle U+200D qu'après un vir[Caractère Unicode non montré]ma. Si une règle `follows-virama` (non montrée ici) a été définie, on peut écrire :

```
  <char cp="200D" when="follows-virama" />
```

Et les variantes ? Voici d'abord comment exprimer que le é est une simple variante du e (non pas que je recommande cette politique : c'est juste un exemple) :

```
 <char cp="00E8">
          <var cp="0065"/>
 </char>
```

Et une variante où un caractère (œ) est mis en correspondance avec deux caractères (oe) :

```
 <char cp="00F6">
          <var cp="006F 0065"/>
 </char>
```

Ces règles portaient sur des caractères qui formaient une partie d'un composant d'un nom. Mais il y a aussi des règles qui portent sur la totalité du composant (section 6 du RFC). Pour cela, d'abord un petit mot sur la notion de classe. On définit une classe de caractères à partir de propriétés Unicode. Ici, on a la classe des vir[Caractère Unicode non montré]ma, définie à partir de la propriété Unicode CCC (”*Canonical Combining Class*”) :

```
<class name="virama" property="ccc:9" />
```

On peut ensuite combiner les classes par intersection, union, etc.

Les règles permettent aussi d'utiliser des opérateurs booléens. Voici par exemple la règle traditionnelle des noms de machines (et pas des noms de domaine <https://www.bortzmeyer.org/host-vs-domain.html>, contrairement aux bêtises qu'on lit souvent), autorisant lettres ASCII, chiffres et tiret. Elle utilise <choice>, qui fait un OU logique entre les termes :

```
<rule name="ldh">
   <choice count="1+">
       <class by-ref="letter"/>
       <class by-ref="digit"/>
       <char cp="002D" comment="literal HYPHEN"/>
   </choice>
</rule>
```

Des exemples plus réalistes et plus détaillés figurent dans l'annexe A.

Le document XML complet a le type MIME application/lgr+xml.

Quelques petits mots sur la sécurité (section 12 du RFC). Le RFC part du préjugé (qui est faux <https://www.bortzmeyer.org/idn-et-phishing.html>) comme quoi les noms en Unicode poseraient des problèmes de sécurité. La section 12 note ensuite (si on accepte cette prémisse) que les règles de ce RFC ne résolvent pas complètement le problème (qui est largement imaginaire). Il reste possible de faire des noms « trompeurs » même avec les règles les plus strictes. Outre question de sécurité à garder en tête : les règles de ce RFC peuvent permettre de concevoir des politiques LGR très complexes, qui peuvent épuiser les ressources de la machine qui tenterait de les évaluer.

Si vous voulez pratiquer avec vos propres LGR, l'ICANN a fait développer un outil qui est accessible en ligne <https://lgrtool.icann.org/> (lisez la documentation <https://www.icann.org/en/system/files/files/lgr-toolset-user-guide-07oct16-en.pdf> pour connaitre le mot de passe) ou bien en source sur GitHub : lgr-core <https://github.com/icann/lgr-core> et lgr-django <https://github.com/icann/lgr-django> pour l'interface Web en Django.