RFC 6991 : Common YANG Data Types

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 juillet 2013

Date de publication du RFC : Juillet 2013

https://www.bortzmeyer.org/6991.html

Le langage de modélisation YANG, normalisé dans le RFC 6020[1], est notamment utilisé dans le protocole de gestion à distance NETCONF (RFC 6241). C'est en YANG que les utilisateurs de NETCONF décrivent les données à manipuler via NETCONF. Ce RFC, successeur du RFC 6021, normalise une bibliothèque de types de données d'usage général, dérivés des types de base de YANG.

Ces types de base sont décrits dans la section 9 du RFC 6020. On y trouve les classiques de l'informatique, "*boolean*", entiers, "*string*", etc. Deux modules YANG sont décrits dans ce nouveau RFC, `ietf-yang-types` pour des types non limités à l'Internet et `ietf-inet-types` pour des types spécifiques à TCP/IP.

Je ne vais pas ici reprendre la liste de tous ces nouveaux types, juste donner quelques exemples de ce qu'apporte ce RFC. En section 3, les types généraux, on trouve par exemple, `counter32`. C'est un compteur, c'est à dire une variable qui ne fait que grimper, jusqu'au moment d'atteindre une valeur maximale (elle repart alors de zéro). Comme son nom l'indique, il est stocké sur 32 bits. Par contre, "*gauge32*" peut monter et descendre.

Le type "*yang-identifier*", un nouveau venu, est une chaîne de caractères ASCII (ne commençant pas par `xml`). Il est défini en YANG par une expression rationnelle :

```
typedef yang-identifier {
       type string {
         length "1..max";
         pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*';
         pattern '.|..|[^xX].*|.[^mM].*|..[^lL].*';
         ...
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6020.txt`

"*date-and-time*", lui, existait déjà dans le RFC 6021. C'est une date au format du RFC 3339 qui est proche du format traditionnel des schémas XML (mais, par contre, est différent de celui utilisé dans le SMI). Au contraire de "*date-and-time*", qui est formaté, "*timeticks*" est simplement un nombre de secondes depuis l'"*epoch*" (non spécifiée : chaque schéma qui utilisera ce type indiquera quelle est l'"*epoch*").

Il y a aussi des types plus orientés réseau comme "*mac-address*" pour les adresses MAC :

```
typedef mac-address {
       type string {
         pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
       }
```

Et d'autres qui empruntent à des identificateurs existants comme "*uuid*" (RFC 4122).

En section 4, les types spécifiques aux protocoles TCP/IP, on a entre autres "*ip-version*" qui permet d'indiquer si on parle d'IPv4 ou d'IPv6 :

```
typedef ip-version {
       type enumeration {
         enum unknown {
           value "0";
           description
            "An unknown or unspecified version of the Internet
             protocol.";
         }
         enum ipv4 {
           value "1";
           description
            "The IPv4 protocol as defined in RFC 791.";
         }
         enum ipv6 {
           value "2";
           description
            "The IPv6 protocol as defined in RFC 2460.";
         }
       }
...
```

Il y a bien sûr les numéros de port :

```
typedef port-number {
       type uint16 {
         range "0..65535";
       }
...
```

Et de système autonome :

```
typedef as-number {
       type uint32;
...
```

Et évidemment les adresses IP :

```
typedef ip-address {
       type union {
         type inet:ipv4-address;
         type inet:ipv6-address;
       }
...
typedef ipv6-address {
       type string {
         pattern '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
               + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
               + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
               + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
               + '(%[\p{N}\p{L}]+)?';
         pattern '(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*))|'
               + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?)'
               + '(%.+)?';
       }
...
```

Un exercice amusant : essayez de retrouver dans quel RFC est défini l'expression rationnelle officielle pour les adresses IPv6... (Avertissement : c'est difficile.)

Enfin, ce RFC donne aussi une définition du type "*domain-name*" pour les noms de domaine :

```
typedef domain-name {
       type string {
         pattern
           '((([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.)*'
         + '([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.?)'
         + '|\.';
         length "1..253";
       }
...
```

C'est une définition très contestable car, limitée à l'ASCII, elle oblige à représenter les IDN par un "*A-label*" (`café.fr` doit être écrit `xn--caf-dma.fr`, RFC 5890). Pire, le texte d'explication qui accompagne le module est plutôt embrouillé quant à la la définition des noms de domaine par rapport à celle des noms de machines (plus restrictive).

Par rapport au RFC 6021, qui contenait la bibliothèque originale, ce RFC ajoute quelques types ("*yang-identifier*", "*uuid*", etc, la liste complète est dans l'annexe A).