RFC 7950 : The YANG 1.1 Data Modeling Language

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 octobre 2016

Date de publication du RFC : Août 2016

https://www.bortzmeyer.org/7950.html

Le protocole standard Netconf (normalisé dans le RFC 6241 [1]) permet de configurer un équipement réseau (par exemple un commutateur) à distance. Netconf fonctionne par des RPC dont les paramètres sont des actions à faire effectuer par l'équipement configuré, ou bien les nouvelles valeurs que peut prendre telle ou telle des variables de configuration de cet équipement. Mais comment savoir quelles actions sont possibles, quelles variables existent, et quelles valeurs elles peuvent prendre ? Jusqu'à présent, cela pouvait se spécifier uniquement dans une documentation en langue naturelle fournie avec l'équipement. Désormais, il est possible de spécifier ces informations dans un langage formel, YANG. La première version de YANG était normalisée dans RFC 6020, ce nouveau RFC normalise la nouvelle version, la 1.1, qui a peu de changements, mais certains cassent la compatibilité ascendante.

Ce RFC 7950 est très détaillé, plus de deux cents pages. Et je n'ai pas personnellement d'expérience pratique avec YANG. Donc, je ne donne ici qu'un très bref résumé. Un tel survol se trouve également dans la section 4 du RFC : YANG modélise les données (configuration **et** état) qui seront utilisées par Netconf. Ces données sont représentées sous forme arborescente. YANG est modulaire (section 5.1 du RFC), un module YANG pouvant se référer à d'autres modules. YANG définit un ensemble de types pour décrire les données (section 9 et RFC 6991). Il permet également d'indiquer les contraintes que doivent respecter les données. YANG, langage de haut niveau, ne décrit pas l'encodage utilisé sur le câble.

Notez que YANG peut être utilisé avec d'autres protocoles que Netconf, comme RESTCONF (décrit dans le RFC 8040).

YANG a donc bien des points communs avec le SMI des RFC 2578 et RFC 2579. Avant Netconf, beaucoup de gens pensaient que toute la gestion des équipements réseau se ferait en SNMP, en s'appuyant

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6241.txt

sur ce modèle SMI. Si, pour la lecture des variables, SNMP s'est largement imposé, force est de constater que, pour l'écriture de variables et pour les actions, SNMP reste très peu utilisé, au profit de toute une galaxie de mécanismes privés (Web, REST, SSH + CLI, etc), galaxie que Netconf vise à remplacer. Une MIB du SMI peut donc être traduite en YANG, l'inverse n'étant pas vrai (YANG étant plus riche).

La syntaxe de YANG utilise des groupes emboîtés, délimités par des accolades. Mais une syntaxe équivalente, en XML, existe, sous le nom de Yin. Tout module YANG peut être traduit en Yin sans perte et réciproquement (voir la section 13 pour plus de détails sur Yin).

Donc, un engin donné, routeur ou autre équipement qu'on veut gérer, est décrit par des modules YANG. Lorsqu'un serveur Netconf à bord dudit engin met en œuvre un module YANG, cela veut dire qu'il permet de modifier, via Netconf, les variables décrites dans le module (le serveur typique met en œuvre plusieurs modules). Voici le début d'un module possible :

```
     // Only an example, not a real module.
     module acme-system {
         namespace "http://acme.example.com/system";
         prefix "acme";

         organization "ACME Inc.";
         contact "joe@acme.example";
         description
             "The module for entities implementing the ACME system.";

         revision 2010-08-05 {
             description "Initial revision.";
         }
...
```

On l'a dit, YANG est arborescent. Les feuilles de l'arbre (section 4.2.2.1 du RFC) contiennent une valeur particulière, par exemple, ici, le nom de l'engin géré :

```
       leaf host-name {
           type string;
           description "Hostname for this system";
       }
```

Ici, `leaf` est un mot-clé de YANG qui indique une feuille de l'arbre (plus de nœuds en dessous), `host-name` est le nom que l'auteur du module a donné à une variable, de type « chaîne de caractères ». Lorsqu'un serveur Netconf enverra cette information à un client (ou réciproquement), elle sera encodée en XML ainsi (Netconf utilise XML pour l'encodage des messages mais d'autres encodages sont possibles, cf. RFC 7951) :

```
       <host-name>my-router.example.com</host-name>
```

Donc, pour résumer, YANG modélise ce qu'on **peut** lire ou modifier, Netconf permet de le lire ou de le modifier effectivement.

Par contre, si un nœud de l'arbre YANG n'est pas une feuille, il est désigné par le mot-clé `container`. Par exemple, il y a ici deux `containers` emboîtés et une feuille :

```
container system {
    container login {
        leaf message {
            type string;
            description
                "Message given at start of login session";
        }
    }
}
```

Lorsque Netconf utilise cette donnée, cela ressemblera, sur le câble, à ceci :

```
<system>
  <login>
    <message>Good morning</message>
  </login>
</system>
```

YANG dispose d'un certain nombre de types pour représenter les données (section 4.2.4 et RFC 6991), mais on peut aussi créer ses types (sections 4.2.5 et 7.3) par exemple ainsi :

```
typedef percent {
    type uint8 {
        range "0 .. 100";
    }
    description "Percentage";
}

leaf completed {
    type percent;
}
```

On a ajouté un intervalle de validité au type prédéfini `uint8`. Autre exemple, en indiquant une valeur par défaut, et en dérivant d'un type défini dans le module `inet` :

```
typedef listen-ipv4-address {
    type inet:ipv4-address;
    default "0.0.0.0";
}
```

YANG a bien d'autres possibilités, décrites en détail dans les sections suivantes. Par exemple, dans un monde idéal, tous les engins mettant en œuvre un module YANG donné géreraient la totalité des variables du module. Mais, comme ce n'est pas forcément le cas, YANG permet des déviations (sections 5.6.3 et 7.20.3). Prenons l'exemple du RFC, un routeur BGP qui suit un module YANG BGP. Le module ne donne pas de limite au nombre de pairs BGP mais un routeur bas de gamme pourrait avoir une limite, disons à 16 pairs. Un client Netconf qui tenterait de configurer un dix-septième pair recevrait donc une erreur. Le mot-clé YANG `deviation` permettrait audit client de savoir à l'avance en quoi ce routeur particulier dévie du modèle BGP général. Le client Netconf n'aurait donc pas à essayer pour voir, il pourrait savoir à l'avance que l'opération de configuration du dix-septième pair ne marchera pas.

La syntaxe formelle de YANG est décrite en section 6. Elle ressemble à celle de langages de programmation comme C ou à celle de SMIng du RFC 3780 (RFC qui n'a pas eu de succès). Cette syntaxe favorise la lisibilité par des humains, le cahier des charges étant de privilégier les lecteurs, pas les auteurs de modules, ni les programmeurs d'outils YANG. À noter que, comme dans toutes les normes modernes, YANG n'est pas limité à l'ASCII et peut utiliser tout Unicode.

Bien que YANG n'utilise pas XML, il réutilise un langage de ce monde, XPath (sections 6.4 et 7.5.3). XPath sert à indiquer les dépendances entre nœuds de l'arbre.

YANG permet en effet de définir des contraintes (section 8) que doivent respecter les variables, avec la directive `must`. Par exemple :

```
    must "ifType != 'ethernet' or " +
         "(ifType = 'ethernet' and ifMTU = 1500)" {
        error-message "An ethernet MTU must be 1500";
    }
```

Voici un exemple de requête Netconf complète, correspondant à une variable YANG. Soit un équipement muni d'un serveur SSH et d'un serveur Netconf pour sa configuration. Disons que le serveur Netconf met en œuvre la variable YANG `port`, définie ainsi :

```
leaf port {
    type inet:port-number;
    default 22;
    description "The port which the SSH server listens to"
}
```

La requête Netconf `<edit-config>` (RFC 6241, section 7.2) qui configure le serveur SSH pour écouter sur le port 2022 serait :

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <system xmlns="http://example.com/schema/config">
        <services>
          <ssh>
            <port>2022</port>
          </ssh>
        </services>
      </system>
    </config>
  </edit-config>
</rpc>
```

Le choix de YANG comme langage standard pour la description des capacités d'un serveur Netconf ne s'était pas fait sans mal. Plusieurs concurrents avaient été envisagés notamment Relax NG, un choix logique puisque Netconf utilise XML. Un langage de description de schémas comme Relax NG semblait donc un choix raisonnable. Parmi les discussions à ce sujet, citons par exemple le débat qui avait eu lieu sur la liste du secteur Applications de l'IETF <http://www.ietf.org/mail-archive/web/discuss/current/msg00987.html>. Les raisons du choix de YANG, telles que vues par les concepteurs de YANG, sont décrites sur le site officiel du projet <http://www.yang-central.org/twiki/bin/view/Main/ComparingYang> mais je trouve cette comparaison très unilatérale.

Un bon tutoriel Netconf, couvrant également YANG, est disponible en <http://www.aims-conference.org/issnsm-2008/06-netconf-yang.pdf>.

Quelles sont les mises en œuvre de YANG ? Il en existe une liste sur le site officiel <http://www.yang-central.org/twiki/bin/view/Main/YangTools>. Voyons par exemple l'outil pyang <http://code.google.com/p/pyang/>, qui sert à valider des schémas YANG (y compris de la nouvelle version 1.1 décrite dans ce RFC) et à les convertir dans d'autres formats. Il ne semble pas trop maintenu mais, bon, il marche. Il peut produire du XSD et du RelaxNG - enfin du DSDL mais c'est presque pareil. Voici un exemple de test d'un schéma invalide (`leaf` a été tapé `laf`) :

```
% pyang test.yang
test.yang:11: error: unexpected keyword "laf"
```

Et, si on corrige :

```
% pyang test.yang
%
```

Maintenant, convertissons en Yin :

```
% cat test.yang
 module acme-foo {
         namespace "http://acme.example.com/foo";
         prefix "acfoo";

         list interface {
             key "name";
             leaf name {
                 type string;
             }

             leaf mtu {
                 type uint32;
                 description "The MTU of the interface.";
             }
         }
     }

% pyang -fyin test.yang
<?xml version="1.0" encoding="UTF-8"?>
<module name="acme-foo"
        xmlns="urn:ietf:params:xml:ns:yang:yin:1"
        xmlns:acfoo="http://acme.example.com/foo">
  <namespace uri="http://acme.example.com/foo"/>
```

```
  <prefix value="acfoo"/>
  <list name="interface">
    <key value="name"/>
    <leaf name="name">
      <type name="string"/>
    </leaf>
    <leaf name="mtu">
      <type name="uint32"/>
      <description>
        <text>The MTU of the interface.</text>
      </description>
    </leaf>
  </list>
</module>
```

Et voici une conversion du même code en DSL :

```
% pyang -fdsdl test.yang
<?xml version='1.0' encoding='UTF-8'?>
<grammar datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
 ns="http://acme.example.com/foo"
 xmlns="http://relaxng.org/ns/structure/1.0"
 xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
 xmlns:acfoo="http://acme.example.com/foo"
 xmlns:dc="http://purl.org/dc/terms"
 xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl"
 xmlns:nm="urn:ietf:params:xml:ns:netmod:dsdl-attrib:1"
 xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <dc:source>YANG module 'acme-foo' (automatic translation)</dc:source>
<start>
   <zeroOrMore>
      <element name="interface" nm:key="name">
  <element name="name">
     <data type="string"/>
  </element>
  <optional>
     <element name="mtu"><a:documentation>The MTU of the interface.</a:documentation>
<data type="unsignedInt"/>
     </element>
  </optional>
       </element>
   </zeroOrMore>
</start>
</grammar>
```

Outre pyang, il y a bien entendu même un mode Emacs, `yang-mode`.

Le site officiel du projet, `<http://www.yang-central.org/>`, contient beaucoup d'autre information sur YANG.

Notez que l'ancien YANG, 1.0, décrit dans le RFC 6020, n'est pas abandonné. L'ancien RFC reste d'actualité pour décrire la version 1.0, qui restera en usage un certain temps. Les principaux changements apportés par la version 1.1 de YANG sont décrits dans la section 1.1 du RFC. La liste est très longue, mais la plupart ne sont que des points de détail. Parmi les changements qui peuvent rendre illégaux des modèles YANG qui étaient légaux avant, il y a par exemple le changement d'interprétation des échappements dans les chaînes de caractères, ou bien le fait qu'une chaîne de caractères qui n'est pas encadrée par des apostrophes ou des guillemets n'a plus le droit de contenir des apostrophes ou guillemets (section 6.1.3). De même, les clés (identificateurs uniques) ne peuvent plus être conditionnelles (instructions `when` ou `if-feature`) ce qui est logique, mais rend également invalide certains anciens modèles YANG.