RFC 1155 : Structure and identification of management information for TCP/IP-based internets

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 18 février 2010

Date de publication du RFC : Mai 1990

<https://www.bortzmeyer.org/1155.html>

Ce RFC a été pendant longtemps la référence pour la description d'informations gérées sur un réseau. Le SMI est le cadre général, dans ce cadre on écrit des MIB, on les interroge avec SNMP (RFC 1157¹) et voilà la gestion de l'Internet. Aujourd'hui, la dernière norme sur le SMI est la version 2, décrite dans le RFC 2578 mais notre RFC 1155 est toujours en statut standard (numéro 16) et toujours d'actualité puisque toutes les MIB n'ont pas été mises à jour (c'est le cas de la MIB-II, la MIB de base, normalisée dans le RFC 1213, même si elle a connu des évolutions comme celle du RFC 2863).

Notre RFC décrit donc le SMI, le cadre de base de la description d'informations qu'on gère, typiquement avec SNMP (à l'époque où a été fait notre RFC, il était encore obligatoire de citer CMIP, le concurrent ISO qui n'a jamais connu de déploiement significatif). Il y a longtemps que la ligne officielle de l'IAB est que tout protocole Internet soit gérable, et aie donc une MIB, écrite selon les règles du SMI (section 1).

La section 2 donne les principes du SMI : séparation de la description des objets et du protocole d'accès (c'était nécessaire à l'époque, pour les raisons politiques qu'explique bien l'un des auteurs du RFC 1155, Marshall Rose, dans son livre "*The Simple Book*"), et utilisation d'un sous-ensemble de ASN.1 pour décrire les classes d'objets gérés.

Un autre principe important, est celui de la simplicité. Bien qu'on puisse se demander s'il est vraiment atteint, il faut se rappeler que les propositions concurrentes, à l'époque, étaient bien pires. Le RFC note avec modestie qu'on ne sait pas encore bien maîtriser la gestion des réseaux et qu'il faut donc

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc1157.txt>

éviter de tout figer par des normes trop strictes (et, effectivement, en 2010, la question est toujours ouverte, comme le montre le RFC 5706). De même, le SMI se veut extensible, car tout ne peut pas être prévu à l'avance

Et, pour ceux qui s'intéressent à l'histoire, outre l'excellent livre de Rose déjà cité, il y a les RFC 1052 et RFC 1109.

Avec la section 3, on en arrive au concret : qu'est-ce qu'une MIB et qu'est-ce qu'on met dedans ? La MIB, écrite en ASN.1, rassemble des objets qui ont tous un nom, plus exactement un OBJECT IDENTIFIER (OID, section 3.1). Ce nom est une suite de chiffres, alloués hiérarchiquement (ce qui garantit l'unicité). Certains de ces chiffres ont aussi parfois une étiquette en lettres. Ainsi, le nœud 1 est étiqueté `iso` et géré par l'ISO. Tous les objets Internet sont sous le sous-arbre 1.3.6.1 (3 étant alloué par l'ISO à d'autres organisations et 6 étant le DoD qui finançait le projet, 1 revenant à l'IAB). En ASN.1, cela se dit :

```
internet    OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }
```

et les objets officiels sont sous le sous-arbre 1.3.6.1.2. La MIB-II du RFC 1213 est ainsi 1.3.6.1.2.1. Pour prendre un exemple d'objet, `ifOperStatus`, l'état effectif d'une interface réseau (par exemple une carte Ethernet) est `{ ifEntry 8 }`, c'est-à-dire 1.3.6.1.2.1.2.2.1.8 puisque `ifEntry` était 1.3.6.1.2.1.2.2.1 (il faut se rappeler qu'une MIB est arborescente). Chaque interface réseau recevra ensuite un OBJECT IDENTIFIER à elle, 1.3.6.1.2.1.2.2.1.8.1, 1.3.6.1.2.1.2.2.1.8.1, etc. Voici un exemple vu avec `snmpwalk` (`-O n` lui dit d'afficher les OID, pas les étiquettes), sur une machine qui a quatre interfaces réseaux dont deux fonctionnent :

```
% snmpwalk -v 1 -O n -c tressecret 192.0.2.68 1.3.6.1.2.1.2.2.1.8
.1.3.6.1.2.1.2.2.1.8.1 = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.8.2 = INTEGER: up(1)
.1.3.6.1.2.1.2.2.1.8.3 = INTEGER: down(2)
.1.3.6.1.2.1.2.2.1.8.4 = INTEGER: down(2)
```

Sans `-O n`, on aurait eu :

```
% snmpwalk -v 1 -c tressecret 192.0.2.68 1.3.6.1.2.1.2.2.1.8
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.2 = INTEGER: up(1)
IF-MIB::ifOperStatus.3 = INTEGER: down(2)
IF-MIB::ifOperStatus.4 = INTEGER: down(2)
```

Si 1.3.6.1.2 désigne les objets « officiels », 1.3.6.1.3 (section 3.1.3 du RFC) est pour les expérimentations et 1.3.6.1.4 pour les objets privés et 1.3.6.1.4.1 pour les objets spécifiques à une entreprise (section 3.1.4 du RFC). Par exemple, si Netaktiv <<http://www.netaktiv.com/>> a obtenu <<https://www.iana.org/assignments/enterprise-numbers>> le numéro 9319, ses objets seront sous 1.3.6.1.4.1.9319.

L'objet a aussi une syntaxe (section 3, par exemple INTEGER, OCTET STRING, OBJECT IDENTIFIER ou NULL. C'est un sous-ensemble d'ASN.1 qui est utilisé pour bâtir des définitions à partir de ces types primitifs, en les organisant en séquences ou en tables (section 3.2.2). Par exemple, l'état d'une interface réseau, `ifOperStatus`, déjà cité, est défini par le RFC 1213 comme INTEGER. Voici la définition complète :

```

ifOperStatus OBJECT-TYPE
    SYNTAX  INTEGER {
        up(1),          -- ready to pass packets
        down(2),
        testing(3)     -- in some test mode
    }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The current operational state of the interface.
        The testing(3) state indicates that no operational
        packets can be passed."
 ::= { ifEntry 8 }

```

Notre RFC définit aussi des types à partir de ces types primitifs, types qui pourront être utilisés par toutes les MIB. C'est le cas de :

- `IpAddress` (section 3.2.3.2), une `OCTET STRING` de quatre octets (IPv6 n'était pas encore inventé),
- `Counter` (section 3.2.3.3), un entier positif sur 32 bits, croissant de manière monotone modulo sa valeur maximale. Si 32 bits semblaient à l'époque permettre des valeurs énormes, c'est aujourd'hui bien trop petit. Pour une interface Ethernet 10G, le `Counter ifInOctets` peut atteindre sa valeur maximale en seulement trois secondes!
- `Gauge` (section 3.2.3.4) qui, contrairement à `Counter`, peut monter et descendre.

Un objet a enfin un encodage sur le câble (section 3.3) car ASN.1 ne spécifie pas comment les objets doivent être sérialisés en bits. Le SMI utilise BER.

La section 4 donne les informations qui doivent être indiquées lors de la définition d'une classe d'objets (comme `ifOperStatus` plus haut) : OID, bien sûr mais aussi accès (lecture seule ou bien aussi écriture), description en langage naturel, etc (la syntaxe formelle est en section 4.3). Cette section est mieux résumée par un exemple, ici tirée d'une expérience avortée, le RFC 1414 :

```

indentOpSys OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE(0..40))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Indicates the type of operating system in use.
        In addition to identifying an operating system,
        each assignment made for this purpose also
        (implicitly) identifies the textual format and
        maximum size of the corresponding indentUserid and
        indentMisc objects.

        The legal values for the 'indentOpSys' strings
        are those listed in the SYSTEM NAMES section of
        the most recent edition of the ASSIGNED NUMBERS
        RFC [8]."
 ::= { indentEntry 2 }

```

Toutes les définitions de ce RFC, en ASN.1, sont rassemblées dans la section 6.

Tiens, Google m'a retrouvé le premier programme significatif que j'avais fait avec SNMP, en 1993 : `<http://www.cpan.org/scripts/netstuff/addresses.on.the.LAN.info>`. Ne me demandez pas les sources, ils semblent avoir été perdus.