RFC 2578 : Structure of Management Information Version 2 (SMIv2)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 septembre 2010

Date de publication du RFC : Avril 1999

https://www.bortzmeyer.org/2578.html

Le SMI est le cadre général, dans ce cadre on écrit des descriptions des objets gérés, les MIB, on les interroge avec le protocole SNMP. L'essentiel de la gestion technique de l'Internet, en tout cas de la partie qui consiste à interroger les équipements réseau, se fait ainsi, d'où l'importance de ce RFC. Il documente la version 2 de la SMI (SMIv2), la version SMIv1 était décrite dans le RFC 1155[1], qui reste d'actualité pour certaines MIB (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). (À noter que la première description de la SMIv2 était dans le RFC 1902, que notre RFC 2578 remplace.) La structure du RFC 2578 est très différente de celle du RFC 1155 et, à mon avis, moins concrète. Cela rend très difficile de voir les changements importants entre SMIv1 et SMIv2, d'autant plus que le RFC 2578 ne comporte pas de chapitre sur les changements.

Notre RFC décrit donc le SMI, le cadre de base de la description d'informations qu'on gère, typiquement avec SNMP. Il y a longtemps que la ligne officielle de l'IAB est que tout protocole Internet soit gérable, et ait donc une MIB, écrite selon les règles du SMI (section 1). Les sections 1 et 3 rappellent des principes du SMI comme le choix de l'utilisation d'un sous-ensemble de ASN.1 pour décrire les classes d'objets gérés. De même, le SMI se veut extensible, car tout ne peut pas être prévu à l'avance. Les MIB permettent de définir trois sortes de choses, les modules (ainsi le RFC 5813 spécifie le module `forcesMib` pour le protocole ForCES,"*Forwarding and Control Element Separation*", RFC 3746), les objets (le même RFC 5813 crée un objet `forcesLatestProtocolVersionSupported`, dont la valeur est un entier indiquant la version du protocole) et les notifications (comme, toujours dans ForCES, `forcesAssociationEntryUp` qui indique qu'une association entre deux éléments du protocole devient opérationnelle).

Toutes les définitions de ce RFC, en ASN.1, sont rassemblées dans la section 2.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1155.txt`

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 3.5). Ce nom est une suite de chiffres, alloués hiérarchiquement (ce qui garantit l'unicité, cf. section 4). Certains de ces chiffres ont aussi parfois une étiquette en lettres. Ainsi, le nœud 1 est étiquetté `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 financait 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`. 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 4 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. 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 7), par exemple `INTEGER`, `OCTET STRING` (voir la section 7.1.2 pour une discussion subtile sur sa taille maximale), `BITS` (un tableau de bits), `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 7.1.12). 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 7.1.5, et limitée à IPv4, le type `Ipv6Address` est apparu dans le RFC 2465), une `OCTET STRING` de quatre octets (IPv6 n'était pas encore inventé),
— `CounterN` (section 7.1.6 et 7.1.10), un entier positif sur 32 (`Counter32`) ou 64 (`Counter64`) bits, croissant de manière monotone modulo sa valeur maximale.
— `Gauge` (section 3.2.3.4) qui, contrairement à `Counter`, peut monter et descendre.

Un exemple d'utilisation de `BITS` (section 7.1.4) figure dans le RFC 5601 (FCS : somme de contrôle) :

```
pwFcsRetentionStatus OBJECT-TYPE
     SYNTAX   BITS {
              remoteIndicationUnknown     (0),
              remoteRequestFcsRetention   (1),
              fcsRetentionEnabled         (2),
              fcsRetentionDisabled        (3),
              localFcsRetentionCfgErr     (4),
              fcsRetentionFcsSizeMismatch (5)
              }
     MAX-ACCESS    read-only
```

La section 5 du RFC est consacrée spécifiquement aux modules et notamment à leurs métadonnées (organisme à contacter, numéro de version, etc). La section 6 se penche sur les définitions d'objets. L'exemple du RFC est imaginaire mais voici un exemple réel tiré du RFC 5813 (CE = "*Control Engine*", la partie du routeur qui fait tourner les protocoles de routage) :

```
forcesLatestProtocolVersionSupported OBJECT-TYPE
          SYNTAX      ForcesProtocolVersion
          MAX-ACCESS  read-only
          STATUS      current
          DESCRIPTION
                 "The ForCES protocol version supported by the CE.
                  The current protocol version is 1.
                  Note that the CE must also allow interaction
                  with FEs supporting earlier versions."
          ::= { forcesMibObjects 1 }

      forcesAssociations OBJECT IDENTIFIER ::= { forcesMibObjects 2 }
```

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

Les MIB sont souvent utilisées comme base pour créer de nouvelles MIB, par exemple parce que le protocole a évolué et qu'on met à jour la MIB. La section 10 donne d'utiles conseils dans ce cas, notamment l'importance de ne pas **retirer** d'éléments de la MIB car les clients SNMP peuvent être restés à l'ancienne MIB. Les nouveaux éléments ne les gènent pas (ils ne les interrogeront pas) mais un élément supprimé va les prendre en défaut. Dans tous les cas, les OID ne doivent jamais être réaffectés. Si la définition d'un objet change de manière incompatible, c'est un nouvel objet et il doit recevoir un nouvel OID. La section 10.2 liste les changements compatibles : ajouter une nouvelle valeur à une énumération, par exemple (mais pas en retirer).

Une implémentation complète du SMI figure en `<http://www.ibr.cs.tu-bs.de/projects/libsmi/>` (sur une Debian, c'est dans le paquetage `libsmi2-dev`). Sur beaucoup de systèmes d'exploitation, les MIB elles-mêmes sont distribuées et, par exemple, sur Debian ou une Gentoo, elles se trouvent dans `/usr/share/snmp/mibs`.