Groupe de travail Réseau
Request for Comments : 1212
STD 16
mars 1991

M. Rose, Performance Systems International
K. McCloghrie, Hughes LAN Systems
éditeurs
Traduction Claude Brière de L'Isle

Brèves définitions de MIB

Statut du présent mémoire

Le présent mémoire définit un format pour la production des modules de MIB. La présente RFC spécifie un document en cours de normalisation de l'IAB pour la communauté de l'Internet, et appelle des discussions et suggestions pour son amélioration. Prière de se référer à l'édition courante de "normes officielles des protocoles de l'IAB" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Table des Matières

1 Résumé

Le présent mémoire décrit une approche directe pour la production de modules de MIB concis et descriptifs. Il est prévu qu'à l'avenir, tous les modules de MIB soient écrits dans ce format.

2 Perspective historique

Comme il est rapporté dans la RFC 1052, Recommandations de l'IAB pour le développement des normes de gestion de réseau de l'Internet [1], une stratégie en deux phases a été entreprise pour la gestion de réseau des internets fondés sur TCP/IP. À court terme, le protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*), défini dans la RFC 1067, était destiné à la gestion des nœuds dans la communauté de l'Internet. À long terme, l'utilisation du cadre de gestion de réseau OSI devait être examiné. Deux documents ont été produits pour définir les informations de gestion : la RFC 1065, qui définit la structure des informations de gestion (SMI, *Structure of Management Information*), et la RFC 1066, qui définit la base de données d'informations de gestion (MIB, *Management Information Base*). Ces deux documents étaient conçus comme étant compatibles à la fois avec le cadre de gestion de réseau SNMP et OSI.

Cette stratégie a été couronnée de succès à court terme : la technologie de gestion de réseau fondée sur Internet a été mise à la disposition du public en quelques mois par les efforts conjugués de la recherche et des commerciaux. Il en est résulté que des parties de la communauté de l'Internet sont devenues gérables assez rapidement.

Comme il est rapporté dans la RFC 1109, Rapport du second groupe ad hoc de révision de la gestion de réseau [2], les exigences des cadres de la gestion de réseau SNMP et OSI étaient plus différents qu'on ne le pensait. À ce titre, l'exigence de compatibilité entre le SMI/MIB et les deux cadres a été suspendue. Cette action a permis au cadre opérationnel de la

gestion de réseau, fondé sur le SNMP, de répondre aux nouveaux besoins du fonctionnement de la communauté Internet avec la production de la MIB-II.

En mai 1990, le noyau central des documents est passé du statut de "protocole standard" à celui de "recommandé". À ce titre, le cadre standard Internet de la gestion de réseau comporte : la structure et l'identification des informations de gestion pour les internets fondés sur TCP/IP, RFC 1155 [3], qui décrit comment sont définis les objets gérés contenus dans la MIB ; la base d'informations de gestion pour la gestion de réseau des internets fondés sur TCP/IP, RFC 1156 [4] qui décrit les objets gérés contenus dans la MIB, et le protocole simple de gestion de réseau, RFC1157 [5] qui définit le protocole utilisé pour gérer ces objets. Conformément à la directive de l'IAB de produire à court terme des systèmes simples et fonctionnels, la liste des objets gérés définie dans la MIB standard Internet été déduite en prenant ceux de ces éléments qui sont considérés comme essentiels. Cependant, la SMI définissait trois mécanismes d'extensibilité : l'ajout de nouveaux objets standard à travers la définition de nouvelles versions de la MIB ; l'ajout d'objets largement disponibles mais non standard par l'arborescence expérimentale ; et l'ajout d'objets privés par les sous arborescences d'entreprises. De tels objets supplémentaires peuvent non seulement être utilisés par des éléments spécifiques des fabricants, mais aussi pour l'expérimentation en tant que nécessaire pour parfaire la connaissance des autres objets essentiels.

Comme plus d'objets sont définis en utilisant la seconde méthode, l'expérience a montré que les descriptions de MIB résultantes contiennent des informations redondantes. Afin de fournir des descriptions de MIB plus concises, et tout aussi informatives, on suggère une amélioration qui permet à l'auteur d'une MIB de retirer les informations redondantes, tout en conservant le texte descriptif important.

Avant de présenter l'approche, une brève présentation du traitement des colonnes d'objets par le protocole SNMP est nécessaire. Cela explique et illustre mieux les raisons de l'amélioration.

3 Objets de colonne

Le protocole SNMP prend en charge les opérations sur les objets de MIB dont la syntaxe est ObjectSyntax comme défini dans la SMI. Dit de façon informelle, les opérations SNMP s'appliquent exclusivement à des objets scalaires. Cependant, il est pratique pour les développeurs d'applications de gestion d'imposer des structures tabulaires imaginaires sur la collection ordonnée des objets qui constituent la MIB. Chacun de ces tableaux conceptuels contient zéro, une ou plusieurs lignes, et chaque ligne peut contenir un ou plusieurs objets scalaires, appelés objets de colonne. Historiquement, cette conceptualisation a été formalisée en utilisant la macro OBJECT-TYPE pour définir à la fois un objet qui correspond à un tableau et un objet qui correspond à une ligne dans ce tableau. (La clause ACCESS pour de tels objets est "non-accessible", bien sûr.) Cependant, on doit souligner que, au niveau du protocole, les relations entre les objets de colonne dans la même ligne sont une question de convention, pas de protocole.

Noter qu'il y a de bonnes raisons pour que la structure du tableau ne soit pas l'affaire du protocole. Considérons le fonctionnement de Get-Next-PDU de SNMP agissant sur le dernier objet de colonne dans une instance d'une ligne conceptuelle ; il retourne la prochaine colonne de la première ligne conceptuelle ou la première instance d'objet survenant après le tableau. À l'opposé, si les lignes étaient une question de protocole, il aurait retourné une erreur à la place. En ne retournant pas une erreur, un seul échange de PDU informe le gestionnaire non seulement de ce que la fin de la ligne/tableau conceptuel a été atteinte, mais fournit aussi des informations sur la prochaine instance d'objet, accroissant par là la densité des informations de l'échange de PDU.

3.1 Suppression de ligne

Néanmoins, il est très utile de fournir le moyen par lequel une ligne conceptuelle peut être retirée d'un tableau. Dans une MIB-II, ceci est obtenu par la définition, pour chaque ligne conceptuelle, d'un objet de colonne avec une valeur d'entier. Si une station de gestion établit la valeur de cet objet à un certain montant, habituellement appelé "invalide", l'effet est alors l'invalidation de la ligne correspondante dans le tableau. Cependant, c'est une question spécifique de la mise en œuvre de savoir si un agent retire une entrée invalidée du tableau. En conséquence, les stations de gestion doivent être prêtes à recevoir des informations de tableaux de la part d'agents qui correspondent à des entrées qui ne sont pas actuellement utilisées. La bonne interprétation de telles entrées exige un examen de l'objet de colonne qui indique le statut d'utilisation.

3.2 Ajout de ligne

Il est aussi très utile d'avoir une claire compréhension de la façon dont une ligne conceptuelle peut être ajoutée à un tableau. Dans SNMP, au niveau du protocole, une station de gestion produit une opération set SNMP qui contient un

ensemble arbitraire de liens de variables. Dans le cas où un agent détecte qu'un ou plusieurs de ces liens de variables se réfèrent à une instance d'objet qui n'est pas actuellement disponible chez cet agent, il peut, conformément aux règles du protocole SNMP, se comporter conformément à l'un des paradigmes suivants :

(1) Il peut rejeter l'opération set SNMP comme se référant à des instances d'objet non existantes en retournant une réponse avec un champ d'état d'erreur réglé à "noSuchName" *(pas de tel nom)* et le champ d'indice d'erreur réglé de façon à se référer à la première référence vacante.

(2) Il peut accepter l'opération set SNMP comme demandant la création de nouvelles instances d'objet correspondant à chacune des instances d'objet désignées dans les liens de variables. La valeur de chaque nouvelle instance d'objet (potentiellement) créé est spécifiée par le composant "valeur" du lien de variable pertinent. Dans ce cas, si la demande spécifie une valeur pour un objet nouvellement (ou précédemment) créé qui semble inappropriée en raison de la valeur ou de la syntaxe, il rejette alors l'opération set SNMP en répondant par un champ d'état d'erreur réglé à badValue et le champ d'indice d'erreur réglé de façon à se référer au premier lien de variable qui pose problème.

(3) Il peut accepter l'opération set SNMP et créer de nouvelles instances d'objet comme décrit en (2) ci-dessus et, de plus, à sa convenance, créer des instances d'objet supplémentaires pour compléter une ligne dans un tableau conceptuel dont peuvent faire partie les nouvelles instances d'objet spécifiées dans la demande.

Il faut souligner que les trois comportements ci-dessus sont tous en parfaite conformité avec la spécification du protocole SNMP et sont pleinement acceptables, sous réserve de toute restriction qui pourrait être imposée par le contrôle d'accès et/ou les définitions des objets de MIB eux-mêmes.

4 Définition d'objets

La SMI standard Internet emploie une approche à deux niveaux à l'égard de la définition d'objet. Une définition de MIB comporte deux parties : une partie textuelle, dans laquelle les objets sont placés dans des groupes, et un module de MIB, dans lequel les objets sont seulement décrits en termes d'OBJECT-TYPE de macro ASN.1, qui sont définis par la SMI.

Un exemple de la première définition pourrait être :

```
OBJET :      sysLocation { system 6 }
Syntaxe :    DisplayString (TAILLE (0..255))
Définition : La localisation physique de ce nœud (par exemple, "cabine téléphonique, 3ème étage").
Accès :      lecture seule.
Statut :     obligatoire
```

Un exemple de la seconde définition pourrait être :

```
TYPE D'OBJET sysLocation
    SYNTAXE     DisplayString (TAILLE (0..255))
    ACCES       lecture seule
    STATUT      obligatoire
        ::= { system 6 }
```

Pour faire court et réduire les chances de fautes de frappe, il semble utile de combiner les deux définitions. Ceci peut être réalisé en définissant une extension de la macro OBJECT-TYPE :

```
IMPORTS
      ObjectName
        FROM RFC1155-SMI
      DisplayString
        FROM RFC1158-MIB;

      MACRO OBJECT-TYPE ::=
      DEBUT
      TYPE NOTATION ::=                          -- doit se conformer à l'ObjectSyntax de la RFC 1155
                  "SYNTAXE" type(ObjectSyntax)
                  "ACCES" Accès
                  "STATUT" Statut
                  DescrPart
                  ReferPart
                  IndexPart
                  DefValPart
      VALEUR NOTATION ::= valeur (VALEUR ObjectName)
```

```
        Accès ::= "lecture seule"
                    | "lecture-écriture"
                    | "écriture-seule"
                    | "non-accessible"
        Statut ::= "obligatoire"
                    | "facultatif"
                    | "obsolète"
                    | "déconseille"

        DescrPart ::=
                "DESCRIPTION" valeur (description DisplayString)
                    | vide

        ReferPart ::=
                "REFERENCE" valeur (reference DisplayString)
                    | vide

        IndexPart ::=
                "INDEX" "{" IndexTypes "}"
                    | vide
        IndexTypes ::=
                IndexType | IndexTypes "," IndexType
        IndexType ::=
                        -- si indexobject, utiliser la valeur SYNTAXE de l'invocation OBJECT-TYPE correspondante
                valeur (indexobject ObjectName)
                        -- autrement utiliser le type SMI désigné
                        -- doit se conformer au IndexSyntax ci-dessous
                    | type (indextype)

        DefValPart ::=
                "DEFVAL" "{" valeur (defvalue ObjectSyntax) "}"
                    | vide

FIN

        IndexSyntax ::=
            CHOIX {
                nombre
                    ENTIER (0..MAX),
                chaîne
                    CHAINE D'OCTETS,
                objet
                    IDENTIFIANT D'OBJET,
                adresse
                    NetworkAddress,
                ipAddress
                    IpAddress
            }
```

4.1 Transposition de la macro OBJECT-TYPE

Il convient de noter que l'expansion de la macro OBJECT-TYPE est quelque chose qui survient conceptuellement durant la mise en œuvre et non durant le fonctionnement.

4.1.1 Transposition de la clause SYNTAXE

La clause SYNTAXE, qui doit être présente, définit la structure de données abstraites correspondant à ce type d'objet. Le langage ASN.1 [6] est utilisé à cette fin. Cependant, la SMI restreint à dessein la construction ASN.1 qui peut être utilisée. Ces restrictions sont expressément faites dans le but de simplifier.

4.1.2 Transposition de la clause ACCES

La clause ACCES, qui doit être présente, définit le niveau minimum de prise en charge exigé pour ce type d'objet. La prise en charge d'autres types d'accès par les mises en œuvre (par exemple, une mise en œuvre peut choisir de permettre d'écrire une variable marquée en lecture seule) est une question locale. De plus, les "points de vue" spécifiques du protocole (par exemple, ceux impliqués indirectement par une communauté SNMP) peuvent faire d'autres restrictions sur l'accès à une variable.

4.1.3 Transposition de la clause STATUT

La clause STATUT, qui doit être présente, définit la prise en charge exigée de la mise en œuvre pour ce type d'objet.

4.1.4 Transposition de la clause DESCRIPTION

La clause DESCRIPTION, qui n'est pas nécessairement présente, contient une définition textuelle de ce type d'objet qui fournit toutes les définitions de sémantique nécessaires pour la mise en œuvre et devrait incorporer toute information qui autrement serait communiquée dans toute annotation de commentaire ASN.1 associée à l'objet. Noter que, afin de se conformer à la syntaxe ASN.1, toute la valeur de cette clause doit être incluse dans des double guillemets, bien que la valeur puisse être multi lignes.

De plus, noter que si le module de MIB ne contient pas ailleurs une description textuelle du type d'objet, la clause DESCRIPTION doit alors être présente.

4.1.5 Transposition de la clause REFERENCE

La clause REFERENCE, qui n'est pas nécessairement présente, contient une référence croisée textuelle à un objet défini dans un autre module de MIB. C'est utile lors de la "dés-OSIfication" d'une MIB produite par une autre organisation.

4.1.6 Transposition de la clause INDEX

La clause INDEX, qui ne peut être présente que si ce type d'objet correspond à une ligne conceptuelle, définit des informations d'identification d'instance pour ce type d'objet. (Autrefois, chaque définition de MIB contenait une section intitulée "Identification des instances d'OBJET à utiliser avec SNMP". En utilisant la clause INDEX, cette section n'a plus de raison d'être car cette clause capture de façon concise la sémantique précise nécessaire à l'identification d'instance.)

Si la clause INDEX n'est pas présente, et si le type d'objet correspond à un objet non colonnaire, les instances de cet objet sont alors identifiées par l'ajout d'un sous identifiant de zéro au nom de cet objet. De plus, noter que si le module de MIB ne contient pas une description textuelle de la façon dont les informations d'identification d'instance sont déduites pour les objets colonnaires, la clause INDEX doit alors être présente.

Pour définir les informations d'identification d'instance, déterminer quelle ou quelles valeurs d'objet vont distinguer de façon non ambiguë une ligne conceptuelle. La syntaxe de ces objets indique comment former l'identifiant d'instance :

(1) valeur d'entier : un seul sous identifiant prenant la valeur entière (cela ne fonctionne qu'avec un entier non négatif) ;
(2) chaîne de longueur fixée, à valeur de chaîne : 'n' sous identifiants, où 'n' est la longueur de la chaîne (chaque octet de la chaîne est codé dans un sous identifiant distinct) ;
(3) chaîne de longueur variable, à valeur de chaîne : 'n+1' sous identifiants, où 'n' est la longueur de la chaîne (le premier sous identifiant est 'n' lui-même, et ensuite, chaque octet de la chaîne est codé dans un sous identifiant distinct) ;
(4) objet valorisé par un identifiant : 'n+1' sous identifiants, où 'n' est le nombre de sous identifiants dans la valeur (le premier sous identifiant est 'n' lui même, et ensuite est copié chaque sous identifiant dans la valeur) ;
(5) valorisé par NetworkAddress : 'n+1' sous identifiants, où 'n' dépend de la sorte d'adresse qui est codée (le premier sous identifiant indique la sorte d'adresse, la valeur 1 indique une adresse IP) ;
(6) valorisé par IpAddress : 4 sous identifiants, dans la notation a.b.c.d familière.

Noter que si une valeur "indextype" est présente (par exemple, ENTIER plutôt que ifIndex), une clause DESCRIPTION doit alors être présente ; le texte contenu ci-après indique la sémantique de la valeur "indextype".
À titre d'exemple, dans le contexte de MIB-II [7], les clauses INDEX suivantes pourraient être présentes :

Objets sous	**clause INDEX**
ifEntry	{ ifIndex }
atEntry	{ atNetIfIndex, atNetAddress }
ipAddrEntry	{ ipAdEntAddr }
ipRouteEntry	{ ipRouteDest }

ipNetToMediaEntry	{ ipNetToMediaIfIndex, ipNetToMediaNetAddress }
tcpConnEntry	{ tcpConnLocalAddress, tcpConnLocalPort, tcpConnRemoteAddress, tcpConnRemotePort }
udpEntry	{ udpLocalAddress, udpLocalPort }
egpNeighEntry	{ egpNeighAddr }

4.1.7 Transposition de la clause DEFVAL

La clause DEFVAL, qui n'est pas nécessairement présente, définit une valeur par défaut acceptable qui peut être utilisée lors de la création d'une instance d'objet, à la discrétion de l'agent qui agit en conformité avec le troisième paradigme du paragraphe 4.2 ci-dessus.

Durant la création de lignes conceptuelles, si une instance d'objet colonnaire n'est pas présente en tant qu'opérande dans l'opération set SNMP correspondante, la valeur de la clause DEFVAL, si elle est présente, indique alors une valeur par défaut acceptable que l'agent pourrait utiliser.

La valeur de la clause DEFVAL doit, bien sûr, correspondre à la clause SYNTAXE pour l'objet. Noter que si un opérande pour l'opération set SNMP est une instance d'un objet en lecture seule, l'erreur noSuchName sera alors retournée. À ce titre, la clause DEFVAL peut être utilisée pour fournir une valeur par défaut acceptable que l'agent pourrait utiliser.

Il est possible qu'il n'existe de valeur par défaut acceptable pour aucun des objets colonnaires dans une ligne conceptuelle pour laquelle la création de nouvelles instances d'objet est permise. Dans ce cas, les objets spécifiés dans la clause INDEX doivent avoir une valeur de clause ACCES correspondante de lecture-écriture.

À titre d'exemple, considérons les clauses DEFVAL possibles suivantes :

ObjectSyntax	**clause DEFVAL**
ENTIER	1 – même chose pour Counter, Gauge, TimeTicks
CHAINE D'OCTET	'ffffffffffff'h
DisplayString	"toute chaîne ASCII NVT"
IDENTIFIANT D'OBJET	sysDescr
IDENTIFIANT D'OBJET	{ system 2 }
NUL	NUL
NetworkAddress	{ internet 'c0210415'h }
IpAddress	'c0210415'h -- 192.33.4.21

4.1.8 Transposition de la clause OBJECT-TYPE

La valeur d'une invocation de la macro OBJECT-TYPE est le nom de l'objet, qui est un identifiant d'objet.

4.2 Exemple d'utilisation

Considérons comment le tableau ipNetToMediaTable provenant de MIB-II pourrait être entièrement décrit :

-- Tableaux de traduction des adresses IP
-- Les tableaux de traduction des adresses IP contiennent les équivalences d'adresse IP à adresse "physique". Certaines interfaces n'utilisent pas de tableau de traduction pour déterminer les équivalences d'adresse (par exemple DDN-X.25 a une méthode algorithmique) ; si toutes les interfaces sont de ce type, le tableau de traduction d'adresse est vide, c'est à dire qu'il a zéro entrée.

```
ipNetToMediaTable OBJECT-TYPE
   SYNTAXE       SEQUENCE DE IpNetToMediaEntry
   ACCES         non accessible
   STATUT        obligatoire
   DESCRIPTION   "Tableau de traduction d'adresse IP utilisé pour transposer les adresses IP en adresses physiques."
   ::= { ip 22 }

ipNetToMediaEntry OBJECT-TYPE
   SYNTAXE       IpNetToMediaEntry
   ACCES         non accessible
   STATUT        obligatoire
   DESCRIPTION    "Chaque entrée contient une équivalence d'adresse IP à adresse 'physique'."
   INDEX         { ipNetToMediaIfIndex, ipNetToMediaNetAddress }
```

```
    ::= { ipNetToMediaTable 1 }

IpNetToMediaEntry ::=
        SEQUENCE {
          ipNetToMediaIfIndex
            ENTIER,
          ipNetToMediaPhysAddress
            CHAINE D'OCTETS,
          ipNetToMediaNetAddress
            IpAddress,
          ipNetoToMediaType
            ENTIER
        }

ipNetToMediaIfIndex OBJECT-TYPE
    SYNTAXE         ENTIER
    ACCES           lecture-écriture
    STATUT          obligatoire
    DESCRIPTION     "Interface sur laquelle l'équivalence de cette entrée est effective. L'interface identifiée par une valeur
                     particulière de cet indice est la même que celle identifiée par la même valeur de ifIndex."
    ::= { ipNetToMediaEntry 1 }

ipNetToMediaPhysAddress OBJECT-TYPE
    SYNTAXE     CHAINE D'OCTETS
    ACCES       lecture-écriture
    STATUT      obligatoire
    DESCRIPTION     "Adresse 'physique' dépendante du support."
    ::= { ipNetToMediaEntry 2 }

ipNetToMediaNetAddress OBJECT-TYPE
   SYNTAXE      adresse IP
   ACCES        lecture-écriture
    STATUT      obligatoire
    DESCRIPTION  "Adresse IP correspondant à l'adresse 'physique' dépendante du support."
    ::= { ipNetToMediaEntry 3 }

ipNetToMediaType OBJECT-TYPE
    SYNTAXE     ENTIER {
                  autre(1),                     -- aucun des suivants
                  invalid2),                    -- une transposition invalidée
                  dynamic(3),
                  static(4)
                }
    ACCES       lecture-écriture
    STATUT      obligatoire
    DESCRIPTION  "Type de transposition. Le réglage de cet objet à la valeur invalid(2) a pour effet d'invalider l'entrée
                  correspondante dans le tableau ipNetToMediaTable. C'est à dire que cela désassocie effectivement
                  l'interface identifiée avec ladite entrée de la transposition identifiée avec ladite entrée. C'est une
                  question spécifique de la mise en œuvre de savoir si l'agent retire une entrée invalidée du tableau. En
                  conséquence, les stations de gestion doivent être prêtes à recevoir de la part d'agents des informations
                  de tableaux qui correspondent à des entrées qui ne sont pas actuellement utilisées. L'interprétation
                  correcte de telles entrées exige l'examen de l'objet ipNetToMediaType pertinent."
    ::= { ipNetToMediaEntry 4 }
```

5 Appendice : Dés-OSIfication des MIB

Il y a eu récemment une importante quantité de travaux pour prendre les MIB définies par les autres organisations (par exemple, l'IEEE) et les dés-OSIfier pour les utiliser dans le cadre de la gestion de réseau standard Internet. Les étapes à franchir pour ce faire sont directes, bien que fastidieuses. Bien sûr, une expérience de rédaction de modules de MIB à utiliser dans le cadre de la gestion de réseau standard Internet sera d'une aide précieuse.

La première étape est de construire le squelette d'un module de MIB, par exemple,

```
RFC1213-MIB DEFINITIONS ::= DEBUT

    IMPORTE
            experimental, OBJECT-TYPE, Counter
                DE RFC1155-SMI;

                                                    -- contacter l'IANA pour le numéro réel
    IDENTIFIANT D'OBJET racine  ::= { experimental xx }

    FIN
```

L'étape suivante est de catégoriser les objets dans des groupes. Pour les MIB expérimentales, les objets facultatifs sont permis. Cependant, lorsque un module de MIB est placé dans l'espace standard Internet, ces objets facultatifs sont soit retirés, soit placés dans un groupe facultatif, qui, s'il est mis en œuvre, oblige à mettre en œuvre tous les objets du groupe. Pour le premier passage, le plus sage est d'ignorer simplement tout objet facultatif dans la MIB originale : l'expérience montre qu'il est préférable de définir d'abord un module de MIB central, qui ne contient que les objets essentiels ; plus tard, si l'expérience l'exige, d'autres objets peuvent être ajoutés.

On doit souligner que les groupes sont des "unités de conformité" au sein d'une MIB : tout dans un groupe est "obligatoire" et les mises en œuvre traitent tout le groupe ou pas du tout.

5.1 Transposition des objets gérés

Ensuite, pour chaque classe d'objets gérés, déterminer si il peut exister plusieurs instances de cette classe d'objets gérés. Si ce n'est pas le cas, pour chacun de ses attributs, utiliser alors la macro OBJECT-TYPE pour créer une définition équivalente.

Autrement, si plusieurs instances de la classe d'objets gérés peuvent exister, définir alors un tableau conceptuel avec des lignes conceptuelles contenant chacune un objet colonnaire pour chaque attribut de la classe d'objets gérés. Si la classe d'objet gérés est contenue dans l'arborescence d'une autre classe d'objets gérés, l'allocation d'un type d'objet est alors normalement requise pour chacun des "attributs distinctifs" de la classe d'objets gérés contenante. Si ils n'existent pas déjà au sein du module de MIB, ils peuvent alors être ajoutés via la définition d'objets colonnaires supplémentaires dans la ligne conceptuelle correspondant à la classe d'objet gérés contenue.

En définissant une ligne conceptuelle, il est utile de considérer l'optimisation des opérations de gestion de réseau qui vont agir sur ces objets colonnaires. En particulier, le plus sage est d'éviter de définir plus d'objets colonnaires au sein d'une ligne conceptuelle qu'il n'en tient dans une seule PDU. Comme règle d'approximation, une ligne conceptuelle ne devrait pas contenir plus d'approximativement 20 objets. De même, ou comme moyen de rester fidèle à la "ligne directrice des 20 objets", les objets colonnaires devraient être groupés en tableaux conformément au groupage des opérations de gestion de réseau qu'on attend d'eux. À ce titre, le contenu d'une ligne conceptuelle devrait refléter les scénarios d'accès typiques, par exemple, ils devraient être organisés selon des lignes fonctionnelles comme une ligne pour les statistiques et une autre ligne pour les paramètres, ou selon des lignes d'utilisation comme d'objets d'utilisation courante contre d'objets rarement nécessaires.

D'un autre côté, on devrait aussi éviter de définir des lignes conceptuelles où le nombre d'objets colonnaires utilisés comme indices dépasse le nombre d'objets utilisés pour détenir les informations. En particulier, le partage des attributs d'une classe d'objets gérés en de nombreux tableaux conceptuels ne devrait pas être utilisé comme moyen d'obtenir le même degré de souplesse/complexité qu'on trouve souvent dans les MIB avec une multitude d'options.

5.1.1 Transposition en la clause SYNTAXE

Lors de la transposition de la macro OBJECT-type en clause SYNTAXE :

(1) Un objet avec la syntaxe BOOLÉEN devient un ENTIER prenant les valeurs vrai(1) ou faux(2).

(2) Un objet avec la syntaxe ENUMERATED devient un ENTIER, prenant une des valeurs données.

(3) Un objet avec la syntaxe CHAINE BINAIRE ne contenant pas plus de 32 bits devient un ENTIER défini comme une somme ; autrement, si plus de 32 bits sont présents, l'objet devient une CHAINE D'OCTETS, avec les bits numérotés de gauche à droite, dans laquelle les bits de moindre poids du dernier octet peuvent être "réservés pour utilisation future".

(4) Un objet avec une syntaxe de chaîne de caractères devient soit une CHAINE D'OCTETS soit une DisplayString,

selon le répertoire de la chaîne de caractères.

5) Un objet non tabulaire avec une syntaxe complexe, comme REEL ou EXTERNE, doit être décomposé, habituellement en une CHAINE D'OCTETS (si cela a un sens). On doit suivre comme une règle que tout objet ayant une syntaxe compliquée devrait être évité.

(6) Les objets tabulaires doivent être décomposés en lignes d'objets colonnaires.

5.1.2 Transposition en la clause ACCES

C'est direct.

5.1.3 Transposition en la clause STATUT

C'est habituellement direct ; cependant, certaines MIB OSI utilisent le terme "recommandé". Dans ce cas, un choix doit être fait entre "obligatoire" et "facultatif".

5.1.4 Transposition en la clause DESCRIPTION

C'est direct : copier simplement le texte, en s'assurant que tous les doubles guillemets incorporés sont apurés (c'est-à-dire remplacés par des guillemets simples ou retirés).

5.1.5 Transposition en la clause REFERENCE

C'est direct : inclure simplement une référence textuelle à l'objet à transposer, le document qui définit l'objet, et peut-être un numéro de page dans le document.

5.1.6 Transposition en la clause INDEX

Décider comment les identifiants d'instance pour les objets colonnaires seront formés et définir cette clause en conséquence.

5.1.7 Transposition en la clause DEFVAL

Décider si on peut allouer une valeur par défaut significative à l'objet à transposer, et si c'est le cas, définir la clause DEFVAL en conséquence.

5.2 Transposition d'action

Les actions sont modélisées comme des objets en lecture-écriture, dans lesquels écrire une valeur particulière résulte en la réalisation de l'action.

5.2.1 Transposition en la clause SYNTAXE

On utilise habituellement une syntaxe d'ENTIER avec une valeur distinctive fournie pour chaque action à laquelle l'objet donne accès. De plus, il y a habituellement une autre valeur distinctive, qui est celle retournée lorsque l'objet est lu.

5.2.2 Transposition en la clause ACCES

Toujours utiliser lecture-écriture.

5.2.3 Transposition en la clause STATUT

C'est direct.

5.2.4 Transposition en la clause DESCRIPTION

C'est direct : copier simplement le texte, en s'assurant que tous les guillemets doubles incorporés sont apurés (c'est-à-dire, remplacés par des guillemets simples ou retirés).

5.2.5 Transposition en la clause REFERENCE

C'est direct : inclure simplement une référence textuelle à l'action à transposer, le document qui définit l'action, et peut-être un numéro de page dans le document.

6 Remerciements

Le présent document a été produit par le groupe de travail SNMP :

Anne Ambler, Spider
Karl Auerbach, Sun
Fred Baker, ACC
Ken Brinkerhoff
Ron Broersma, NOSC
Jack Brown, US Army
Theodore Brunner, Bellcore
Jeffrey Buffum, HP
John Burress, Wellfleet
Jeffrey D. Case, University of Tennessee at Knoxville
Chris Chiptasso, Spartacus
Paul Ciarfella, DEC
Bob Collet
John Cook, Chipcom
Tracy Cox, Bellcore
James R. Davin, MIT-LCS
Eric Decker, cisco
Kurt Dobbins, Cabletron
Nadya El-Afandi, Network Systems
Gary Ellis, HP
Fred Engle
Mike Erlinger
Mark S. Fedor, PSI
Richard Fox, Synoptics
Karen Frisa, CMU
Chris Gunner, DEC
Fred Harris, University of Tennessee at Knoxville
Ken Hibbard, Xylogics
Ole Jacobsen, Interop
Ken Jones
Satish Joshi, Synoptics
Frank Kastenholz, Racal-Interlan
Shimshon Kaufman, Spartacus
Ken Key, University of Tennessee at Knoxville
Jim Kinder, Fibercom
Alex Koifman, BBN
Christopher Kolb, PSI
Cheryl Krupczak, NCR
Paul Langille, DEC
Peter Lin, Vitalink
John Lunny, TWG
Carl Malamud
Randy Mayhew, University of Tennessee at Knoxville
Keith McCloghrie, Hughes LAN Systems
Donna McMaster, David Systems
Lynn Monsanto, Sun
Dave Perkins, 3COM
Jim Reinstedler, Ungerman Bass
Anil Rijsinghani, DEC
Kathy Rinehart, Arnold AFB
Kary Robertson
Marshall T. Rose, PSI (chair)
L. Michael Sabo, NCSC
Jon Saperia, DEC
Greg Satz, cisco
Martin Schoffstall, PSI
John Seligson
Steve Sherry, Xyplex
Fei Shu, NEC
Sam Sjogren, TGV
Mark Sleeper, Sparta
Lance Sprung
Mike St.Johns
Bob Stewart, Xyplex
Emil Sturniold
Kaj Tesink, Bellcore
Dean Throop, Data General
Bill Townsend, Xylogics
Maurice Turcotte, Racal-Milgo
Kannan Varadhou
Sudhanshu Verma, HP
Bill Versteeg, Network Research Corporation
Warren Vik, Interactive Systems
David Waitzman, BBN
Steve Waldbusser, CMU
Dan Wintringhan
David Wood
Wengyik Yeong, PSI
Jeff Young, Cray Research

7 Références

[1] V. Cerf, "Recommandations de l'IAB pour le développement des normes de gestion de réseau de l'Internet", RFC 1052, NRI, avril 1988.

[2] V. Cerf, "Rapport du second groupe ad hoc de révision de la gestion de réseau", RFC 1109, NRI, août 1989.

[3] M. Rose et K. McCloghrie, "Structure et identification des informations de gestion pour les internets fondés sur TCP/IP", RFC 1155, Performance Systems International, Hughes LAN Systems, mai 1990.

[4] K. McCloghrie et M. Rose, "Base de données d'informations de gestion pour la gestion de réseau des internets fondés sur TCP/IP", RFC 1156, Hughes LAN Systems, Performance Systems International, mai 1990.

[5] J. Case, M. Fedor, M. Schoffstall et J. Davin, "Protocole simple de gestion de réseau", RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, mai 1990.

[6] Systèmes de traitement de l'information – Interconnexion des système ouverts - Spécification de la notation de syntaxe abstraite n° 1 (ASN.1), Organisation mondiale de normalisation, Norme internationale 8824, décembre 1987.

[7] M. Rose, éditeur, "Base de donées d'informations de gestion pour la gestion de réseau des internets fondés sur TCP/IP : MIB-II", RFC 1213, Performance Systems International, mars 1991.

8 Considérations pour la sécurité

Les questions de sécurité ne sont pas abordées dans le présent mémoire.

9 Adresse des auteurs

Marshall T. Rose
Performance Systems International
5201 Great America Parkway
Suite 3106
Santa Clara, CA 95054
téléphone : +1 408 562 6222
mél : mrose@psi.com
X.500 : rose, psi, us

Keith McCloghrie
Hughes LAN Systems
1225 Charleston Road
Mountain View, CA 94043
1225 Charleston Road
Mountain View, CA 94043
téléphone : (415) 966-7934
mél : kzm@hls.com