Groupe de travail Réseau
Request for Comments : 1157
Rendue obsolète : RFC 1098

J. Case, SNMP Research
M. Fedor, Performance Systems International
M. Schoffstall, Performance Systems International
J. Davin, MIT Laboratory for Computer Science

Mai 1990
Traduction Claude Brière de l'Isle, décembre 2006

Protocole simple de gestion de réseau (SNMP)

Table des matières

1. Statut de ce mémo

La présente RFC est une nouvelle version de la RFC 1098, avec une section "Statut de ce mémo" modifiée plus quelques corrections typographiques mineures. Le présent mémo définit un protocole simple par lequel les informations de gestion pour un élément de réseau peuvent être inspectés ou modifiés par des utilisateurs logiquement distants. En particulier, conjointement avec les mémos compagnons qui décrivent la structure des informations de gestion ainsi que la base d'informations de gestion, ces documents fournissent une architecture et un système simples et pratiques pour la gestion des internets fondés sur TCP/IP et en particulier l'Internet.

Le Bureau des activités de l'Internet recommande que toutes les mises en œuvre IP et TCP soient gérables par le réseau. Cela implique la mise en œuvre du MIB Internet (RFC-1156) et au moins un des deux protocoles de gestion recommandés SNMP (RFC-1157) ou CMOT (RFC-1095). Il vaut de noter qu'à l'heure actuelle SNMP est une norme Internet adoptée et que CMOT est un projet de norme. Voir aussi les RFC sur les exigences d'hébergement et de passerelle pour des informations plus précises sur les conditions d'application de la présente norme.

Prière de se reporter à l'édition la plus récente de la RFC "IAB Official Protocol Standards" pour des informations actualisées sur l'état et le statut des protocoles Internet standard.

La distribution du présent mémo n'est soumise à aucune restriction.

2. Introduction

Comme indiqué dans la RFC 1052, Recommandations de l'IAB pour le développement des normes de gestion du réseau Internet [1], une stratégie à deux niveaux a été entreprise pour la gestion de réseau des internets fondés sur TCP/IP. A court terme, le protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*) était utilisé pour gérer les nœuds dans la communauté de l'Internet. Pour le long terme, l'utilisation du cadre de gestion de réseau OSI était à prendre en compte. Deux documents ont été produits pour définir les informations de gestion : la RFC 1065, qui définit la Structure des informations de gestion (SMI) [2], et la RFC 1066, qui définit la Base d'informations de gestion (MIB, *Management Information Base*) [3]. Ces deux documents étaient conçus pour être compatibles à la fois avec SNMP et le cadre de gestion de réseau OSI.
Cette stratégie a été un relatif succès à court terme : la technologie de gestion de réseau fondée sur l'Internet a été mise en application, à la fois par la communauté de la recherche et par le commercial, en quelques mois. Il en est résulté que des portions de la communauté de l'Internet ont eu très rapidement des réseaux gérables.

Comme indiqué dans la RFC 1109, Rapport du second groupe ad hoc de révision de la gestion de réseau [4], les exigences des cadres SNMP et de gestion de réseau OSI ont été plus différentes que prévu. Les exigences de compatibilité entre le SMI/MIB et les deux cadres ont donc été suspendues. Cette action a permis au cadre de fonctionnement de la gestion de réseau, le protocole SNMP, de répondre aux nouveaux besoins opérationnels de la communauté de l'Internet en produisant des documents définissant de nouveaux éléments de MIB.

L'IAB avait conçu SNMP, SMI, et le MIB Internet initial comme des "Protocoles normalisés" complets, avec un statut de "Recommandé". Par cette action, L'IAB recommande que toutes les mises en œuvre IP et TCP soient en réseau gérable et que les mises en œuvre qui sont en réseau gérable soient supposées adopter et mettre en œuvre SMI, MIB, et SNMP.

A ce titre, le cadre de gestion de réseau actuel pour les internets fondés sur TCP/IP consistent en : des informations de gestion de structure et d'identification pour les internets fondés sur TCP/IP, qui décrivent comment les objets gérés contenus dans le MIB sont définis comme exposé dans la RFC 1155 [5] ; une base d'informations de gestion pour la gestion de réseau d'internets fondés sur TCP/IP, qui décrit les objets gérés contenus dans le MIB comme exposé dans la RFC 1156 [6] ; et, le protocole simple de gestion de réseau, qui définit le protocole utilisé pour gérer ces objets, comme exposé dans le présent document.

Comme indiqué dans la RFC 1052, les Recommandations de l'IAB pour le développement de normes de gestion de réseau pour l'Internet [1], le Bureau des activités Internet (IAB, *Internet Activities Board*) a dirigé le groupe de travail d'ingénierie de l'Internet (IETF, *Internet Engineering Task Force*) vers la création de nouveaux groupes de travail dans le domaine de la gestion de réseau. Un groupe a été chargé des nouvelles spécifications et définitions des éléments à inclure dans la base d'informations de gestion (MIB). L'autre a été chargé de définir les modifications à apporter au protocole simple de gestion de réseau (SNMP) pour s'accommoder des besoins à court terme des communautés de fabricants de réseau et des utilisateurs, et de le mettre en ligne avec les résultats du groupe de travail MIB.

Le groupe de travail MIB a produit deux documents, dont l'un définit une Structure pour les informations de gestion (SMI) [2] à utiliser pour les objets gérés contenus dans le MIB. Un second document [3] définit la liste des objets gérés.

Le résultat du groupe de travail Extensions SNMP est le présent document, qui incorpore des changements à la définition SNMP initiale [7] nécessaires pour atteindre l'alignement avec le résultat du groupe de travail MIB. Les changements devraient être minimes afin d'être cohérent avec la directive de l'IAB que les groupes de travail soient "extrêmement sensibles à la nécessité de garder SNMP simple." Bien qu'un soin considérable ait été apporté au débat sur les changements à SNMP qui sont rapportés dans le présent document, le protocole résultant n'est pas rétro-compatible avec son prédécesseur, le protocole simple de surveillance de passerelle (SGMP, *Simple Gateway Monitoring Protocol*) [8]. Bien que la syntaxe du protocole ait été altérée, la philosophie d'origine, les décisions de conception, et l'architecture restent intactes. Afin d'éviter la confusion, de nouveaux ports UDP ont été alloués pour l'utilisation du protocole décrit dans la présent document.

3. L'architecture SNMP

Une collection de stations de gestion de réseau et d'éléments de réseau est implicite dans le modèle architectural de SNMP. Les stations de gestion de réseau exécutent des applications de gestion qui surveillent et contrôlent des éléments de réseau. Les éléments de réseau sont des appareils comme les hôtes, les passerelles, les serveurs de terminaux, et ainsi de suite, qui ont des agents de gestion responsables de l'accomplissement des fonctions de gestion de réseau demandées par les stations de gestion du réseau. Le protocole simple de gestion de réseau (SNMP) sert à

communiquer les informations de gestion entre les stations de gestion du réseau et les agents dans les éléments de réseau.

3.1. Buts de l'architecture

Le protocole SNMP minimise explicitement le nombre et la complexité des fonctions de gestion réalisées par l'agent de gestion lui-même. Cet objectif est intéressant au moins sur quatre plans :

(1) Les coûts de développement des logiciels d'agent de gestion nécessaires à la prise en charge du protocole en sont réduits d'autant.

(2) Le degré de fonction de gestion prise en charge à distance en est augmenté d'autant, ce qui laisse la plus large place à la gestion des ressources de l'Internet dans les tâches de gestion.

(3) Le degré de fonction de gestion qui est pris en charge à distance est augmenté d'autant, ce qui impose le moins de restrictions possibles sur la forme et la sophistication des outils de gestion.

(4) Des ensembles simplifiés de fonctions de gestion sont facilement compris et utilisés par les développeurs d'outils de gestion de réseau.

Un second objectif du protocole est que le paradigme fonctionnel pour la surveillance et le contrôle soit suffisamment extensible pour s'accommoder d'aspects supplémentaires, éventuellement non prévus du fonctionnement et de la gestion du réseau.

Un troisième objectif est que l'architecture soit, autant que possible, indépendante de l'architecture et des mécanismes des hôtes ou serveurs particuliers.

3.2 Éléments de l'architecture

L'architecture SNMP articule la solution du problème de la gestion du réseau en termes :

(1) de portée des informations de gestion communiquées par le protocole,
(2) de représentation des informations de gestion communiquées par le protocole,
(3) des opérations sur les informations de gestion communiquées par le protocole,
(4) de forme et de signification des échanges entre les entités de gestion,
(5) de définition des relations administratives entre les entités de gestion, et
(6) de forme et de signification des références aux informations de gestion.

3.2.1 Portée des informations de gestion

La portée des informations de gestion communiquées par le fonctionnement du protocole SNMP est exactement celle représentée par les instances de tous les types d'objets non agrégés définis dans le MIB standard Internet ou définis n'importe où conformément aux conventions établies dans le SMI standard de l'Internet SMI [5].

La prise en charge des types d'objet agrégés dans le MIB n'est pas exigée pour la conformité avec le SMI ni réalisée par le protocole SNMP.

3.2.2 Représentation des informations de gestion

Les informations de gestion communiquées par le fonctionnement de SNMP sont représentées conformément au sous ensemble du langage ASN.1 [9] qui est spécifié pour la définition des types non agrégés dans le SMI.

Le protocole SGMP avait adopté la convention d'utiliser un sous ensemble bien défini du langage ASN.1 [9]. Le protocole SNMP continue et étend cette tradition en utilisant un sous ensemble légèrement plus complexe du langage ASN.1 pour décrire les objets gérés et pour décrire les unités de données de protocole utilisées pour gérer ces objets. De plus, le désir de faciliter une éventuelle transition des protocoles de gestion de réseau fondés sur OSI a conduit à la définition en langage ASN.1 d'une Structure d'informations de gestion (SMI) [5] et d'une Base d'informations de gestion (MIB) [6] standard pour l'Internet. L'utilisation du langage ASN.1 était en partie encouragée par le succès de l'utilisation d'ASN.1 dans des tâches précédentes, en particulier, pour SGMP. Les restrictions sur l'utilisation de l'ASN.1 qui font partie du SMI contribuent à la simplicité souhaitée et validée par expérience avec SGMP.

Au nom de la simplicité, le protocole SNMP utilise seulement un sous ensemble des règles de codage de base de l'ASN.1 [10]. A savoir que tous les codages utilisent la forme de longueur définie. De plus, chaque fois que c'est permis, des codages non incorporés sont utilisés plutôt que des codages incorporés. Cette restriction s'applique à tous les aspects du codage en ASN.1, à la fois pour les unités de données de protocole de niveau supérieur et pour les objets de données qu'elles contiennent.

3.2.3 Opérations prises en charge sur les informations de gestion

Le protocole SNMP modèle toutes les fonctions d'agent de gestion comme des altérations ou inspections de variables. Et donc, une entité de protocole sur une hôte logiquement distant (qui peut être l'élément de réseau lui-même) interagit avec l'agent de gestion résident sur l'élément de réseau afin de restituer (get) ou d'altérer (set) les variables. Cette stratégie a au moins deux conséquences positives :

(1) Il a pour effet de limiter à deux le nombre de fonctions de gestions essentielles réalisées par l'agent de gestion : une opération pour assigner une valeur à une configuration spécifiée ou un autre paramètre à une autre pour restituer une telle valeur.

(2) Le second effet de cette décision est d'éviter d'introduire dans la définition du protocole la prise en charge de commandes de gestion impératives : le nombre de commandes de cette sorte est en pratique toujours croissant et la sémantique de telles commandes est en général arbitrairement complexe.

La stratégie implicite dans le protocole SNMP est que la surveillance de l'état du réseau à tout niveau significatif de détail est principalement accomplie en interrogeant les informations appropriées de la part du ou des centres de surveillance. Un nombre limité de messages non sollicités (traps) guide l'ordonnancement et l'objet des interrogations. Limiter le nombre de messages non sollicités est cohérent avec l'objectif de simplicité et de réduction au minimum de la quantité de trafic généré par la fonction de gestion du réseau.

L'exclusion des commandes impératives en dehors de l'ensemble des fonctions de gestion explicitement prises en charge n'est pas de nature à empêcher le fonctionnement d'aucun agent de gestion désirable. Actuellement, la plupart des commandes sont des demandes soit pour régler ma valeur de certains paramètres, soit de restituer une telle valeur, et la fonction des quelques commandes impératives actuellement prises en charge est facilement traitée en mode asynchrone par ce modèle de gestion. Dans ce schéma, une commande impérative peut être réalisée comme le réglage de la valeur d'un paramètre qui déclanche ensuite l'action désirée. Par exemple, plutôt que de mettre en œuvre une "commande de réamorçage", cette action peut être invoquée simplement en réglant un paramètre indiquant le nombre de secondes jusqu'au réamorçage du système.

3.2.4. Forme et signification des échanges de protocole

La communication des informations de gestion entre les entités de gestion est réalisée dans le protocole SNMP au moyen de l'échange de messages de protocole. La forme et la signification de ces messages sont définies ci-dessous à la Section 4.

Cohérent avec l'objectif de minimiser la complexité de l'agent de gestion, l'échange de messages SNMP ne requiert qu'un service de datagrammes non fiable, et chaque message est entièrement représente de façon indépendante par un seul datagramme de transport. Alors que ce document spécifie l'échange de messages via le protocole UDP [11], les mécanismes du protocole SNMP sont généralement convenables pour une large variété d'utilisation de services de transport.

3.2.5 Définition des relations administratives

L'architecture SNMP admet une variété de relations administratives entre les entités qui participent au protocole. Les entités qui résident dans les stations de gestion et les éléments de réseau qui communiquent les unes avec les autres en utilisant le protocole SNMP sont appelées des entités d'application SNMP. Les processus d'homologue qui mettent en œuvre le protocole SNMP, et donc prennent en charge les entités d'application SNMP, sont appelées des entités de protocole.

Un appariement d'un agent SNMP avec un ensemble arbitraire d'entités d'application SNMP est appelé une communauté SNMP. Chaque communauté SNMP est nommée par une chaîne d'octets, qui est appelé le nom de communauté pour la dite communauté.

Un message SNMP créé par une entité d'application SNMP qui appartient en fait à la communauté SNMP nommée par le composant de communauté du dit message est appelé un message SNMP authentique. L'ensemble des règles par lesquelles un message SNMP est identifié comme un message SNMP authentique pour une communauté SNMP particulière est appelé un schéma d'authentification. Une mise en œuvre d'une fonction qui identifie des messages SNMP authentiques conformément à un ou plusieurs schémas d'authentification est appelée un service d'authentification.

En clair, la gestion effective des relations administratives entre les entités d'application SNMP requiert des services d'authentification qui (en utilisant le chiffrement ou d'autres techniques) sont capables d'identifier les messages SNMP authentiques avec un fort degré de certitude. Certaines mises en œuvre de SNMP peuvent souhaiter ne prendre en charge qu'un service d'authentification trivial qui identifie tous les messages SNMP comme des messages SNMP authentiques.

Pour tout élément de réseau, un sous ensemble d'objets dans le MIB qui appartiennent à cet élément est appelé une vue de MIB SNMP. Noter que les noms des types d'objet représentés dans une vue de MIB SNMP n'a pas besoin d'appartenir à un seul sous arbre de l'espace de nom du type d'objet.

Un élément de l'ensemble { READ-ONLY, READ-WRITE } *(lecture seule, lecture écriture)* est appelé un mode d'accès SNMP.

Un appariement du mode d'accès SNMP avec une vue de MIB SNMP est appelé un profil de communauté SNMP. Un profil de communauté SNMP représente des privilèges d'accès spécifiés à des variables dans une vue de MIB spécifiée. Pour chaque variable dans la vue de MIB d'un profil de communauté SNMP donnée, l'accès à cette variable est représenté par le profil, conformément aux conventions suivantes :

(1) si la dite variable est définie dans le MIB avec un "Access:" de "none", elle est indisponible comme opérande pour tout opérateur ;

(2) si la dite variable est définie dans le MIB avec un "Access:" de "read-write" *(lecture-écriture)* ou "write-only" *(écriture seule)* et que le mode d'accès pour ce profil donné est READ-WRITE, cette variable est disponible comme opérande pour les opérations get, set, et trap ;

(3) autrement, la variable est disponible comme opérande pour les opérations get et trap ;

(4) dans les cas où une variable "write-only" est un opérande utilisé pour les opérations get ou trap, la valeur donnée pour la variable est spécifique de la mise en œuvre.

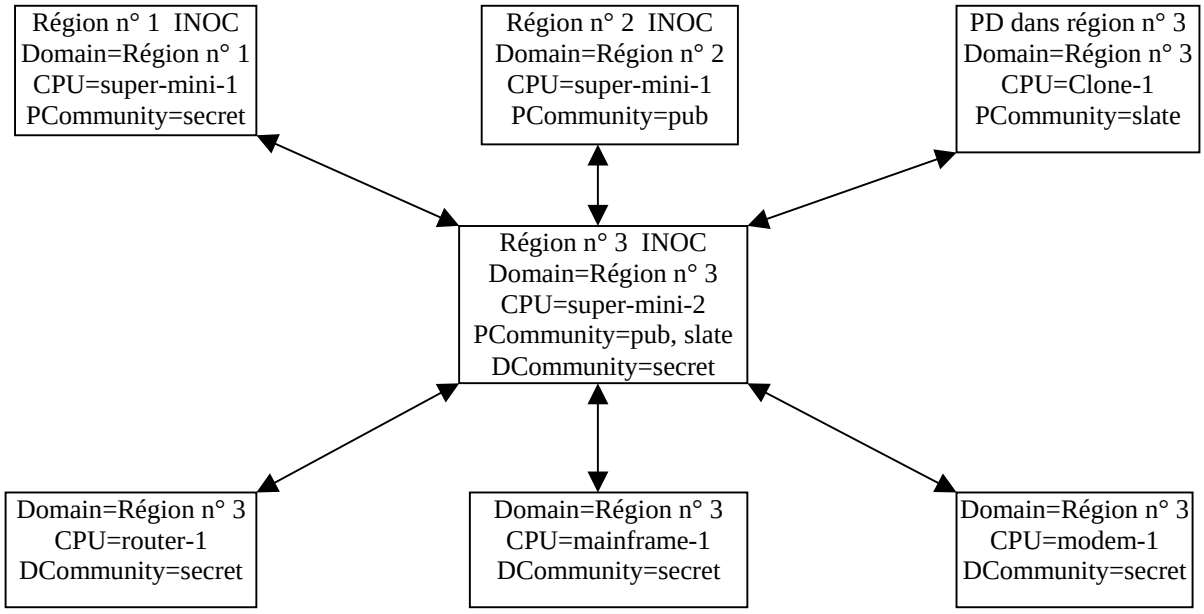
Un appariement d'une communauté SNMP avec un profil de communauté SNMP s'appelle une politique d'accès SNMP. Une politique d'accès représente un profil de communauté spécifié offert par l'agent SNMP d'une communauté SNMP spécifiée aux autres membres de cette communauté. Toutes les relations administratives entre les entités d'application SNMP sont définies architecturalement en termes de politiques d'accès SNMP.

Pour toute politique d'accès SNMP, si l'élément de réseau sur lequel réside l'agent SNMP pour la communauté SNMP spécifiée n'est pas celui auquel appartient la vue de MIB pour le profil spécifié, cette politique est alors appelée une politique d'accès de mandataire SNMP. L'agent SNMP associé à une politique d'accès par mandataire est appelé un agent SNMP mandataire. Alors que des définitions négligentes de politiques d'accès par mandataire peuvent résulter en des boucles de gestion, une définition prudente des politiques de mandataire est utile au moins de deux façons :

(1) elle permet la surveillance et le contrôle des éléments de réseau qui ne sont pas autrement adressables en utilisant le protocole de gestion et le protocole de transport. C'est à dire qu'un agent mandataire peut fournir une fonction de conversion de protocole permettant à une station de gestion d'appliquer un cadre de gestion cohérent à tous les éléments de réseau, y compris des appareils comme des modems, des multiplexeurs, et autres appareils qui acceptent des cadres de gestion différents.

(2) elle protège potentiellement les éléments de réseau de l'élaboration des politiques de contrôle d'accès. Par exemple, un agent mandataire peut mettre en œuvre un contrôle d'accès sophistiqué par lequel divers sous ensembles de variables au sein du MIB sont rendus accessibles à différentes stations de gestion sans augmenter la complexité de l'élément de réseau.

A titre d'exemple, la Figure 1 illustre les relations entre des stations de gestion, des agents mandataires, et des agents de gestion. Dans cet exemple, l'agent mandataire est vu comme un centre d'opérations de réseau Internet (INOC, *Internet Network Operations Center*) normal d'un domaine administratif qui a une relation de gestion standard avec un ensemble d'agents de gestion.



Domain : domaine administratif de l'élément
PCommunity : nom d'une communauté utilisant un agent mandataire
DCommunity : nom d'une communauté directe

Figure 1 : Exemple de configuration de gestion de réseau

3.2.6. Forme et signification des références aux objets gérés

Le SMI exige que la définition d'une adresse de protocole de gestion vise :

(1) la résolution des références de MIB ambiguës,

(2) la résolution des références de MIB en présence de plusieurs versions de MIB, et

(3) l'identification d'instances particulières de types d'objet définis dans le MIB.

3.2.6.1 Résolution des références de MIB ambiguës

Comme la portée de toute opération SNMP est par conception confinée aux objets relevant d'un seul élément de réseau, et comme toutes les références SNMP à des objets de MIB sont faites (implicitement ou explicitement) par des noms de variable uniques, il n'y a pas de possibilité qu'une référence SNMP à un type d'objet défini dans le MIB puisse se résoudre en plusieurs instances de ce type.

3.2.6.2 Résolution des références à travers les versions de MIB

L'instance d'objet référée dans toute opération SNMP est exactement celle spécifiée au titre de la demande d'opération ou (dans le cas d'une opération get-next) son successeur immédiat dans le MIB comme un tout. En particulier, une référence à un objet au titre d'une version du MIB standard de l'Internet ne se résout en aucun objet qui ne fasse pas partie de la dite version du MIB standard de l'Internet, excepté dans le cas où l'opération demandée est get-next et où le nom de l'objet spécifié est lexicographiquement le dernier parmi les noms de tous les objets présentés au titre de la dite version de MIB standard de l'Internet.

3.2.6.3 Identification des instances d'objet

Les noms pour tous les types d'objet dans le MIB sont définis explicitement dans la MIB standard de l'Internet ou dans d'autres documents qui se conforment aux conventions de dénomination du SMI. Le SMI exige que les protocoles de gestion conformes définissent des mécanismes pour identifier les instances individuelles de ces types d'objet pour un élément de réseau particulier.

Chaque instance de tout type d'objet défini dans le MIB est identifiée dans les opérations SNMP par un nom unique appelé son "nom variable." En général, le nom d'une variable SNMP est un OBJECT IDENTIFIER *(identifiant d'objet)* de forme x.y, où x est le nom d'un type d'objet non agrégé défini dans le MIB et y est un fragment de OBJECT IDENTIFIER qui, d'une façon spécifique du type d'objet nommé, identifie l'instance désirée.

Cette stratégie de dénomination admet la plus complète exploitation de la sémantique de GetNextRequest-PDU (voir la Section 4), parce qu'elle alloue des noms pour les variables en question de sorte qu'ils soient contigus dans l'ordre lexicographique de tous les noms de variable connus dans le MIB.

La dénomination spécifique du type des instances d'objet est définie ci-dessous pour un certain nombre de classes de types d'objet. Les instances d'un type d'objet auxquelles aucune des conventions de dénomination suivantes ne sont applicables sont nommées par des OBJECT IDENTIFIER de la forme x.0, où x est le nom du dit type d'objet dans la définition du MIB.

Par exemple, supposons qu'on veuille identifier une instance de la variable sysDescr. La classe d'objet pour sysDescr est :

```
iso org dod internet mgmt mib system sysDescr
 1   3   6      1       2       1       1        1
```

Et donc, le type d'objet, x, serait 1.3.6.1.2.1.1.1 auquel est ajouté un sous identifiant d'instance de 0. C'est à dire que 1.3.6.1.2.1.1.1.0 identifie cette seule et unique instance de sysDescr.

3.2.6.3.1 Noms de type d'objet ifTable

Le nom d'une interface de sous réseau, s, est la valeur de OBJECT IDENTIFIER de la forme i, où i a la valeur de cette instance du type d'objet ifIndex associé à s.

Pour chaque type d'objet, t, pour lequel le nom définit, n, a un préfixe de ifEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.s, où s est le nom de l'interface de sous réseau pour lequel i représente l'information.

Par exemple, supposons qu'on veuille identifier l'instance de la variable ifType associée à l'interface 2. En conséquence, ifType.2 identifierait l'instance désirée.

3.2.6.3.2 Noms de type d'objet atTable

Le nom d'une adresse réseau en mémoire cache par AT, x, est un OBJECT IDENTIFIER de forme 1.a.b.c.d, où a.b.c.d est la valeur (en notation "dot" familière *[c'est-à-dire en octets séparés par des points]*) du type d'objet atNetAddress associé à x.

Le nom d'une équivalence de traduction d'adresse e est une valeur d'OBJECT IDENTIFIER de forme s.w, telle que s soit la valeur de cette instance du type d'objet atIndex associé à e et telle que w soit le nom de l'adresse de réseau en mémoire cache par AT associée à e.

Pour chaque type d'objet, t, pour lequel le nom défini, n, a un préfixe de atEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.y, où y est le nom de l'équivalence de traduction d'adresse sur laquelle i représente l'information.

Par exemple, supposons qu'on veuille trouver l'adresse physique d'une entrée dans le tableau de traduction d'adresse (en mémoire cache ARP) associée à une adresse IP de 89.1.1.42 et d'interface 3. En conséquence, atPhysAddress.3.1.89.1.1.42 identifierait l'instance désirée.

3.2.6.3.3 Noms de type d'objet ipAddrTable

Le nom d'un élément de réseau IP-addressable, x, est l'OBJECT IDENTIFIER de forme a.b.c.d tel que a.b.c.d soit la valeur (dans la notation familière d'octets séparés par des points) de cette instance du type d'objet ipAdEntAddr associé à x.

Pour chaque type d'objet, t, pour lequel le nom défini, n, a un préfixe de ipAddrEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.y, où y est le nom de l'élément de réseau IP adressable au sujet duquel i représente l'information.

Par exemple, supposons qu'on veuille trouver le gabarit de réseau d'une entrée dans le tableau d'interface IP associé à une adresse IP de 89.1.1.42. En conséquence, ipAdEntNetMask.89.1.1.42 identifierait l'instance désirée.

3.2.6.3.4 Noms de type d'objet ipRoutingTable

Le nom d'une route IP, x, est l'OBJECT IDENTIFIER de forme a.b.c.d telle que a.b.c.d soit la valeur (dans la notation familière d'octets séparés par des points) de cette instance du type d'objet ipRouteDest associé à x.

Pour chaque type d'objet, t, pour lequel le nom défini, n, a un préfixe de ipRoutingEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.y, où y est le nom de la route IP au sujet de laquelle i représente l'information.

Par exemple, supposons qu'on veuille trouver le prochain bond d'une entrée dans le tableau d'acheminements IP associé à la destination 89.1.1.42. En conséquence, ipRouteNextHop.89.1.1.42 identifierait l'instance désirée.

3.2.6.3.5 Noms de type d'objet tcpConnTable

Le nom d'une connexion TCP, x, est l'OBJECT IDENTIFIER de forme a.b.c.d.e.f.g.h.i.j telle que a.b.c.d soit la valeur (dans la notation familière d'octets séparés par des points) de cette instance du type d'objet tcpConnLocalAddress associé à x et telle que f.g.h.i soit la valeur (dans la notation familière d'octets séparés par des points) de cette instance du type d'objet tcpConnRemoteAddress associé à x et telle que e soit la valeur de cette instance du type d'objet tcpConnLocalPort associé à x et telle que j soit la valeur de cette instance du type d'objet tcpConnRemotePort associé à x.

Pour chaque type d'objet, t, pour lequel le nom défini, n, a un préfixe de tcpConnEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.y, où y est le nom de la connexion TCP au sujet de laquelle i représente l'information.

Par exemple, supposons qu'on veuille trouver l'état d'une connexion TCP entre l'adresse locale de 89.1.1.42 sur le port TCP 21 et l'adresse distante de 10.0.0.51 sur le port TCP 2059. En conséquence, tcpConnState.89.1.1.42.21.10.0.0.51.2059 identifierait l'instance désirée.

3.2.6.3.6 Noms de type d'objet egpNeighTable

Le nom d'un voisin EGP, x, est l'OBJECT IDENTIFIER de forme a.b.c.d telle que a.b.c.d soit la valeur (dans la notation familière d'octets séparés par des points) de cette instance du type d'objet egpNeighAddr associé à x.

Pour chaque type d'objet, t, pour lequel le nom défini, n, a un préfixe de egpNeighEntry, une instance, i, de t est nommée par un OBJECT IDENTIFIER de forme n.y, où y est le nom du voisin EGP au sujet duquel i représente l'information.

Par exemple, supposons qu'on veuille trouver l'état voisin pour l'adresse IP 89.1.1.42. En conséquence, egpNeighState.89.1.1.42 identifierait l'instance désirée.

4 Spécification du protocole

Le protocole de gestion du réseau est un protocole d'application par lequel les variables du MIB d'un agent peuvent être inspectées ou altérées.

Les communications entre les entités de protocole sont accomplies par l'échange de messages, dont chacun est entièrement représenté de façon indépendante au sein d'un seul datagramme UDP en utilisant les règles de codage de base de l'ASN.1 (comme exposé au paragraphe 3.2.2). Un message consiste en un identifiant de version, un nom de communauté SNMP, et une unité de données de protocole (PDU).

Une entité de protocole reçoit des messages au port UDP 161 sur l'hôte auquel elle est associée pour tous les messages excepté pour ceux qui rapportent des traps (par exemple, tous les messages excepté ceux qui contiennent le Trap-PDU). Les messages qui rapportent des traps devraient être reçus sur le port UDP 162 pour traitement ultérieur. Une mise en œuvre de ce protocole n'est pas obligée d'accepter des messages d'une longueur excédant 484 octets. Cependant, il est recommandé que les mises en œuvre prennent en charge de pus grands datagrammes chaque fois que c'est faisable.

Il est obligatoire que toutes les mises en œuvre du protocole SNMP prennent en charge les cinq PDU : GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU, et Trap-PDU.

RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS

```
        ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
            FROM RFC1155-SMI;

-- message de niveau supérieur
        Message ::=
            SEQUENCE {
                version                 -- version-1 pour la présente RFC
                    INTEGER {
                        version-1(0)
                    },

                community               -- nom de communauté
                    OCTET STRING,
                data                    -- l'exemple des, PDU est le plus commun
                    ANY                 -- l'authentification est utilisée
            }

-- unités de données de protocole
            PDU ::=
                CHOICE {
                    get-request
                        GetRequest-PDU,
                    get-next-request
                        GetNextRequest-PDU,
                    get-response
                        GetResponse-PDU,
                    set-request
                        SetRequest-PDU,
                    trap
                        Trap-PDU
                    }
                -- les PDU individuelles et les types de données communément utilisés seront définis ultérieurement
            END
```

4.1 Éléments de procédure

Ce paragraphe décrit les actions d'une entité de protocole mettant en œuvre le protocole SNMP. Noter, cependant, qu'il n'est pas prévu de contraindre l'architecture interne à aucune conformité de mise en œuvre.

Dans le texte qui suit, on utilise le terme d'adresse de transport. Dans le cas du protocole UDP, une adresse de transport consiste en une adresse IP avec un port UDP. D'autres services de transport peuvent être utilisés pour prendre en charge le protocole SNMP. Dans ces cas, la définition d'une adresse de transport devrait être faite en conséquence.

Les actions de niveau supérieur d'une entité de protocole qui génère un message sont les suivantes :

(1) Elle construit d'abord la PDU appropriée, par exemple, la GetRequest-PDU, comme un objet ASN.1.

(2) Elle passe ensuite cet objet ASN.1 avec son nom de communauté, son adresse source de transport et l'adresse de transport de destination, au service qui met en œuvre le schéma d'authentification désiré. Ce service d'authentification retourne un autre objet ASN.1.

(3) L'entité de protocole construit ensuite un objet Message ASN.1, en utilisant le nom de communauté et l'objet ASN.1 résultant.

(4) Ce nouvel objet ASN.1 est alors mis en série, en utilisant les règles de codage de base de l'ASN.1, et ensuite envoyé en utilisant un service de transport à l'entité de protocole homologue.

De façon similaire, les actions de niveau supérieur d'une entité de protocole qui reçoit un message sont les suivantes :

(1) Elle effectue une analyse rudimentaire du datagramme entrant pour construire un objet ASN.1 correspondant à un objet Message ASN.1. Si l'analyse échoue, elle détruit le datagramme et n'effectue aucune autre action.

(2) Elle vérifie ensuite le numéro de version du message SNMP. Si il y a discordance, elle détruit le datagramme et n'effectue aucune autre action.

(3) L'entité de protocole passe ensuite le nom de communauté et les données d'utilisateur trouvées dans l'objet Message ASN.1, avec les adresses de transport de source et de destination du datagramme au service qui mettra en œuvre le schéma d'authentification désiré. Cette entité retourne un autre objet ASN.1, ou signale un échec d'authentification. Dans ce dernier cas, l'entité de protocole note cet échec, génère (éventuellement un trap, et détruit le datagramme et n'effectue aucune autre action.

(4) L'entité de protocole effectue ensuite une analyse rudimentaire sur l'objet ASN.1 retourné du service d'authentification pour construire un objet ASN.1 correspondant à un objet de PDU ASN.1. Si l'analyse échoue, elle détruit le datagramme et n'effectue aucune autre action. Autrement, en utilisant la communauté SNMP nommée, elle choisit le profil approprié, et la PDU est traitée en conséquence. Si, à la suite de ce traitement, un message est retourné, l'adresse de transport de source d'où le message de réponse est envoyé doit être identique à l'adresse de transport de destination à laquelle était envoyé le message de demande d'origine.

4.1.1 Constructions communes

Avant d'introduire les six types de PDU du protocole, il est approprié de considérer les constructions ASN.1 utilisées le plus fréquemment :

```
-- informations de demande/réponse
        RequestID ::=
                INTEGER
        ErrorStatus ::=
                INTEGER {
                    noError(0),
                    tooBig(1),
                    noSuchName(2),
                    badValue(3),
                    readOnly(4)
                    genErr(5)
          }
        ErrorIndex ::=
                INTEGER

-- liaisons variables
        VarBind ::=
                SEQUENCE {
                    name
                        ObjectName,
                    value
                        ObjectSyntax
            }
        VarBindList ::=
                SEQUENCE OF
                    VarBind
```

Les RequestID *(identifiant de demande)* sont utilisés pour distinguer les demandes en cours. Par l'utilisation de RequestID, une entité d'application SNMP peut corréler les réponses entrantes avec les demandes en cours. Dans les cas où le service de datagramme utilisé n'est pas fiable, le RequestID fournit aussi un moyen simple d'identifier des messages dupliqués par le réseau.

Une instance de ErrorStatus différente de zéro est utilisée pour indiquer qu'une exception est survenue lors du traitement d'une demande. Dans ces cas, ErrorIndex peut fournir des informations supplémentaires en indiquant quelle variable dans une liste en est la cause.

Le terme de variable se réfère à une instance d'un objet géré. Une liaison de variable, ou VarBind, se réfère à l'appariement du nom d'une variable et de la valeur de la variable. Une VarBindList est une simple liste des noms des variables et des valeurs correspondantes. Certaines PDU ne sont concernées que par le nom d'un variable et non par sa valeur (par exemple, GetRequest-PDU). Dans ce cas, la portion valeur de la liaison est ignorée par l'entité de protocole. Cependant, la portion valeur doit toujours avoir une syntaxe et un codage ASN.1 valides. Il est recommandé que la valeur ASN.1 NULL soit utilisée pour la portion valeur de telles liaisons.

4.1.2 La GetRequest-PDU

La forme de la GetRequest-PDU est :

```
GetRequest-PDU ::=
        [0]
          IMPLICIT SEQUENCE {
            request-id
              RequestID,
            error-status        -- toujours 0
              ErrorStatus,
            error-index         -- toujours 0
              ErrorIndex,
            variable-bindings
              VarBindList
          }
```

La GetRequest-PDU n'est générée par une entité de protocole qu'à la demande de son entité d'application SNMP.

A réception de la GetRequest-PDU, l'entité de protocole receveuse répond conformément à toute règle applicable de la liste ci-dessous :

(1) Si, pour tout objet nommé dans le champ variable-bindings, le nom de l'objet ne correspond pas exactement au nom d'un objet disponible pour les opérations get dans la vue de MIB pertinente, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status *(état d'erreur)* est noSuchName *(ce nom n'existe pas)*, et la valeur du champ error-index *(indice d'erreur)* est l'indice du dit composant de nom d'objet dans le message reçu.

(2) Si, pour tout objet nommé dans le champ variable-bindings, l'objet est un type agrégé (comme défini dans le SMI), l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est noSuchName, et la valeur du champ error-index est l'indice du dit composant de nom d'objet dans le message reçu.

(3) Si la taille de la GetResponse-PDU générée comme décrit ci-dessous devait dépasser une limite locale, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est tooBig *(trop gros)*, et la valeur du champ error-index est zéro.

(4) Si, pour tout objet nommé dans le champ variable-bindings, la valeur de l'objet ne peut pas être restitués pour des raisons non couvertes par une des règles en cours, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est genErr *(erreur générale)* et la valeur du champ error-index est l'indice du dit composant de nom d'objet dans le message reçu.

Si aucune des règles précédentes ne s'applique, l'entité de protocole receveuse envoie alors à l'origine du message reçu la GetResponse-PDU telle que, pour chaque objet nommé dans le champ variable-bindings du message reçu, le composant correspondant de la GetResponse-PDU représente le nom et la valeur de cette variable. La valeur du champ error- status de la GetResponse-PDU est noError et la valeur du champ error-index est zéro. La valeur du champ request-id de la GetResponse-PDU est celle du message reçu.

4.1.3 La GetNextRequest-PDU

La forme de la GetNextRequest-PDU est identique à celle de la GetRequest-PDU excepté pour l'indication du type de la PDU. En langage ASN.1 :

```
GetNextRequest-PDU ::=
  [1]
    IMPLICIT SEQUENCE {
      request-id
        RequestID,
      error-status        -- toujours 0
        ErrorStatus,
      error-index         -- toujours 0
        ErrorIndex,
      variable-bindings
```

```
        VarBindList
    }
```

La GetNextRequest-PDU n'est générée par une entité de protocole qu'à la demande de son entité d'application SNMP.

A réception de la GetNextRequest-PDU, l'entité de protocole receveuse répond conformément à toute règle applicable dans la liste ci-dessous :

(1) Si, pour tout nom d'objet dans le champ variable-bindings, ce nom ne précède pas lexicographiquement le nom d'un objet disponible pour les opérations get dans la vue de MIB pertinente, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est noSuchName, et la valeur du champ error-index est l'indice du dit composant de nom d'objet dans le message reçu.

(2) Si la taille de la GetResponse-PDU générée comme décrit ci-dessous devait excéder une limite locale, l'entité receveuse enverrait alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status serait tooBig, et la valeur du champ error-index serait zéro.

(3) Si, pour tout objet nommé dans le champ variable-bindings, la valeur du successeur lexicographique de l'objet nommé ne peut pas être restituée pour des raisons non couvertes par une des règles précédentes, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est genErr et la valeur du champ error-index est l'indice du dit composant de nom d'objet dans le message reçu.

Si aucune des règles précédentes ne s'applique, l'entité de protocole receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de telle sorte que, pour chaque nom dans la champ variable-bindings du message reçu, le composant correspondant de la GetResponse-PDU représente le nom et la valeur de cet objet dont le nom est, dans l'ordre lexicographique des noms de tous les objets disponibles pour les opérations get dans la vue de MIB pertinente, ainsi que la valeur du champ de nom du composant donné, le successeur immédiat de cette valeur. La valeur du champ error-status de la GetResponse-PDU est noError et la valeur du champ errorindex est zéro. La valeur du champ request-id de la GetResponse-PDU est celle du message reçu.

4.1.3.1 Exemple de traversée de tableau

Une importante utilisation de la GetNextRequest-PDU est la traversée de tableaux conceptuels d'informations au sein du MIB. La sémantique de ce type de message SNMP, avec les mécanismes spécifiques du protocole pour identifier les instances individuelles de types d'objet dans le MIB, permet l'accès aux objets en rapport dans le MIB comme si ils bénéficiaient d'une organisation en tableau.

Par l'échange SNMP décrit ci-dessous, une entité d'application SNMP peut extraire l'adresse de destination et la prochaine passerelle de bond pour chaque entrée dans le tableau d'acheminement d'un élément de réseau particulier. Supposons que ce tableau d'acheminement ait trois entrées :

Destination	NextHop	Métrique
10.0.0.99	89.1.1.42	5
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

La station de gestion envoie à l'agent SNMP une GetNextRequest-PDU contenant les valeurs de OBJECT IDENTIFIER indiquées comme les noms de variables demandés :

GetNextRequest (ipRouteDest, ipRouteNextHop, ipRouteMetric1)

L'agent SNMP répond par un GetResponse-PDU:

```
GetResponse (( ipRouteDest.9.1.2.3 =  "9.1.2.3" ),
        ( ipRouteNextHop.9.1.2.3 = "99.0.0.3" ),
        ( ipRouteMetric1.9.1.2.3 = 3 ))
```

La station de gestion continue par :

GetNextRequest (ipRouteDest.9.1.2.3,

```
        ipRouteNextHop.9.1.2.3,
        ipRouteMetric1.9.1.2.3 )
```

L'agent SNMP répond :

```
GetResponse (( ipRouteDest.10.0.0.51 = "10.0.0.51" ),
        ( ipRouteNextHop.10.0.0.51 = "89.1.1.42" ),
        ( ipRouteMetric1.10.0.0.51 = 5 ))
```

La station de gestion continue par :

```
GetNextRequest ( ipRouteDest.10.0.0.51,
        ipRouteNextHop.10.0.0.51,
        ipRouteMetric1.10.0.0.51 )
```

L'agent SNMP répond :

```
GetResponse (( ipRouteDest.10.0.0.99 = "10.0.0.99" ),
        ( ipRouteNextHop.10.0.0.99 = "89.1.1.42" ),
        ( ipRouteMetric1.10.0.0.99 = 5 ))
```

La station de gestion continue par :

```
GetNextRequest ( ipRouteDest.10.0.0.99,
        ipRouteNextHop.10.0.0.99,
        ipRouteMetric1.10.0.0.99 )
```

Comme il n'y a plus d'autres entrées dans le tableau, l'agent SNMP retourne les objets qui sont les suivants dans l'ordre lexicographique des noms d'objet connus. Cette réponse signale la fin du tableau d'acheminement à la station de gestion.

4.1.4 La GetResponse-PDU

La forme de la GetResponse-PDU est identique à celle de la GetRequest-PDU excepté pour l'indication du type de PDU. Dans le langage ASN.1 :

```
GetResponse-PDU ::=
   [2]
      IMPLICIT SEQUENCE {
         request-id
            RequestID,
         error-status
            ErrorStatus,

         error-index
            ErrorIndex,

         variable-bindings
            VarBindList
      }
```

La GetResponse-PDU n'est générée par une entité de protocole qu'à réception de la GetRequest-PDU, GetNextRequest-PDU, ou SetRequest-PDU, comme décrit ailleurs dans le présent document.

A réception de la GetResponse-PDU, l'entité de protocole receveuse présente son contenu à son entité d'application SNMP.

4.1.5 La SetRequest-PDU

La forme de la SetRequest-PDU est identique à celle de la GetRequest-PDU excepté pour l'indication du type de PDU. Dans le langage ASN.1 :

```
SetRequest-PDU ::=
   [3]

      IMPLICIT SEQUENCE {
         request-id
            RequestID,

         error-status       -- toujours 0
            ErrorStatus,

         error-index        -- toujours 0
            ErrorIndex,

         variable-bindings
            VarBindList
      }
```

La SetRequest-PDU n'est générée par une entité de protocole qu'à la demande de son entité d'application SNMP.

A réception de la SetRequest-PDU, l'entité receveuse répond conformément à toute règle applicable dans la liste ci-dessous :

(1) Si, pour tout objet nommé dans le champ variable-bindings, l'objet n'est pas disponible pour les opérations set dans la vue de MIB pertinente, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est noSuchName, et la valeur du champ error-index est l'indice du composant de nom du dit objet dans le message reçu.

(2) Si, pour tout objet nommé dans le champ variable-bindings, le contenu du champ valeur ne manisfeste pas, conformément au langage ASN.1, un type, longueur, et valeur qui soient cohérents avec ce qui est exigé pour la variable, l'entité receveuse envoi alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est badValue, et la valeur du champ error-index est l'indice du dit nom d'objet dans le message reçu.

(3) Si la taille du message de type GetResponse généré comme décrit ci-dessous devait excéder une limite locale, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est tooBig, et la valeur du champ error-index est zéro.

(4) Si, pour tout objet nommé dans le champ variable-bindings, la valeur de l'objet nommé ne paut pas être altérée pour des raisons non couvertes par une des règles précédentes, l'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique, excepté que la valeur du champ error-status est genErr et la valeur du champ error-index est l'indice du composant de nom du dit objet dans le message reçu.

Si aucune des règles précédentes ne s'applique, alors pour chaque objet nommé dans le champ variable-bindings du message reçu, la valeur correspondante est allouée à la variable. Chaque allocation de variable spécifiée par la SetRequest-PDU devrait être effectuée comme si elle était un établissement simultané par rapport à toutes les autres allocations spécifiées dans le même message.

L'entité receveuse envoie alors à l'origine du message reçu la GetResponse-PDU de forme identique excepté que la valeur du champ error-status du message généré est noError et la valeur du champ error-index est zéro.

4.1.6 Le Trap-PDU

La forme de la Trap-PDU est :

```
Trap-PDU ::=
   [4]

      IMPLICIT SEQUENCE {
        enterprise       -- type de l'objet qui génère le trap, voir sysObjectID en [5]
```

```
        OBJECT IDENTIFIER,

    agent-addr      -- adresse de l'objet générant le trap
        NetworkAddress,

    generic-trap    -- type trap générique
        INTEGER {
            coldStart(0),
            warmStart(1),
            linkDown(2),
            linkUp(3),
            authenticationFailure(4),
            egpNeighborLoss(5),
            enterpriseSpecific(6)
        },

    specific-trap   -- code spécifique, présent même si generic-trap n'est pas enterpriseSpecific
        INTEGER,

    time-stamp      -- temps écoulé entre la dernière (ré)initialisation de l'entité réseau et la génération du trap
        TimeTicks,

    variable-bindings   -- informations "intéressantes"
        VarBindList
}
```

La Trap-PDU n'est générée par une entité de protocole qu'à la demande de l'entité d'application SNMP. les moyens par lesquels une entité d'application SNMP choisit les adresses de destination des entités d'application SNMP sont spécifiques de la mise en œuvre.

A réception de la Trap-PDU, l'entité de protocole receveuse présente son contenu à son entité d'application SNMP.

La signification du composant variable-bindings de la Trap-PDU est spécifique de la mise en œuvre.

Les interprétations de la valeur du champ generic-trap sont :

4.1.6.1 Trap coldStartp

Un trap coldStart(0) signifie que l'entité de protocole expéditrice se réinitialise de sorte que la configuration de l'agent ou la mise en œuvre de l'entité de protocole peut en être altérée.

4.1.6.2 Trap warmStart

Un trap warmStart(1) signifie que l'entité de protocole expéditrice se réinitialise de sorte que ni la configuration de l'agent ni la mise en œuvre de l'entité de protocole n'en sont altérées.

4.1.6.3 Trap linkDown

Un trap linkDown(2) signifie que l'entité de protocole expéditrice reconnaît une défaillance dans une des liaisons de communication représentées dans la configuration de l'agent.

Le trap Trap-PDU de type linkDown contient comme premier élément de sa variable-bindings, le nom et la valeur de l'instance de ifIndex pour l'interface affectée.

4.1.6.4 Trap linkUp

Un trap linkUp(3) signifie que l'entité de protocole expéditrice reconnaît qu'une des liaisons de communication représentées dans la configuration de l'agent est rétablie.

Le trap Trap-PDU de type linkUp contient comme premier élément de sa variable-bindings, le nom et la valeur de l'instance de ifIndex pour l'interface affectée.

4.1.6.5 Trap authenticationFailure

Un trap authenticationFailure(4) signifie que l'entité de protocole expéditrice est le destinataire d'un massage de protocole qui n'est pas correctement authentifié. Alors que les mises en œuvre du protocole SNMP doivent être capables de générer ce trap, elles doivent aussi être capables de supprimer l'émission de tels traps via un mécanisme spécifique de la mise en œuvre.

4.1.6.6 Trap egpNeighborLoss

Un trap egpNeighborLoss(5) signifie qu'un voisin EGP pour lequel l'entité de protocole expéditirice était un homologue EGP a été déclassé et que la relation d'homologue n'est plus obtenue.

Le trap Trap-PDU de type egpNeighborLoss contient comme premier élément de sa variable-bindings, le nom et la valeur de l'instance de egpNeighAddr pour le voisin affecté.

4.1.6.7 Trap enterpriseSpecific

Un trap enterpriseSpecific(6) signifie que l'entité de protocole expéditrice reconnaît qu'un événement spécifique de l'entreprise est survenu. Le champ specific-trap identifie le trap particulier survenu.

5 Définitions

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

  IMPORTS
     ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
        FROM RFC1155-SMI;
     -- message de niveau supérieur
     Message ::=
           SEQUENCE {
              version         -- version-1 pour cette RFC.
                 INTEGER {
                    version-1(0)
                 },
              community       -- nom de communauté.
                 OCTET STRING,
              data            -- par exemple, des PDU si une authentification triviale est utilisée.
                 ANY
           }
     -- unités de données de protocole
     PDUs ::=
           CHOICE {
                    get-request
                       GetRequest-PDU,
                    get-next-request
                       GetNextRequest-PDU,
                    get-response
                       GetResponse-PDU,
                    set-request
                       SetRequest-PDU,
                    trap
                       Trap-PDU
                 }
   -- PDUs
     GetRequest-PDU ::=
        [0]
           IMPLICIT PDU
     GetNextRequest-PDU ::=
        [1]
           IMPLICIT PDU
     GetResponse-PDU ::=
```

```
   [2]
      IMPLICIT PDU
SetRequest-PDU ::=
   [3]
      IMPLICIT PDU
PDU ::=
      SEQUENCE {
        request-id
           INTEGER,
        error-status      -- parfois ignoré.
           INTEGER {
              noError(0),
              tooBig(1),
              noSuchName(2),
              badValue(3),
              readOnly(4),
              genErr(5)
           },
        error-index       -- parfois ignoré.
          INTEGER,
        variable-bindings – ces valeurs sont parfois ignorées.
           VarBindList
      }
Trap-PDU ::=
   [4]
     IMPLICIT SEQUENCE {
        enterprise        -- type d'objet générant des trap, voir sysObjectID dans [5].
           OBJECT IDENTIFIER,
        agent-addr        -- adresse de l'objet générant des traps.
           NetworkAddress,
        generic-trap      -- type de trap générique.
           INTEGER {
              coldStart(0),
              warmStart(1),
              linkDown(2),
              linkUp(3),
              authenticationFailure(4),
              egpNeighborLoss(5),
              enterpriseSpecific(6)
           },
        specific-trap  -- code spécifique, présent même si le trap générique n'est pas enterpriseSpecific.
           INTEGER,
        time-stamp     -- temps écoulé entre la dernière (ré)initialization de l'entité réseau et la génération du trap.
           TimeTicks,
         variable-bindings – informations "intéressantes".
           VarBindList
      }

-- variable bindings
VarBind ::=
      SEQUENCE {
        name
           ObjectName,
        value
           ObjectSyntax
      }
VarBindList ::=
      SEQUENCE OF
        VarBind
END
```

6 Remerciements

Le présent mémo a été influencé par le groupe de travail Extensions SNMP de l'IETF :

Karl Auerbach, Epilogue Technology	Phill Gross, The MITRE Corporation
K. Ramesh Babu, Excelan	Satish Joshi, ACC
Amatzia Ben-Artzi, 3Com/Bridge	Dan Lynch, Advanced Computing Environments
Lawrence Besaw, Hewlett-Packard	Keith McCloghrie, The Wollongong Group
Jeffrey D. Case, University of Tennessee at Knoxville	Marshall T. Rose, The Wollongong Group (chair)
Anthony Chung, Sytek	Greg Satz, cisco
James Davidson, The Wollongong Group	Martin Lee Schoffstall, Rensselaer Polytechnic Institute
James R. Davin, MIT Laboratory for Computer Science	Wengyik Yeong, NYSERNet
Mark S. Fedor, NYSERNet	

7 Références

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

[2] Rose, M., et K. McCloghrie, "Structure et identification des informations de gestion pour les internets fondés sur TCP/IP", RFC 1065, TWG, août 1988.

[3] McCloghrie, K., et M. Rose, "Base d'informations de gestion pour la gestion de réseau d'internets fondés sur TCP/IP", RFC 1066, TWG, août 1988.

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

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

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

[7] Case, J., M. Fedor, M. Schoffstall, et J. Davin, "A Simple Network Management Protocol", Note de travail de l'Internet Engineering Task Force, Network Information Center, SRI International, Menlo Park, California, mars 1988.

[8] Davin, J., J. Case, M. Fedor, et M. Schoffstall, "Protocole simple de surveillance de passerelle", RFC 1028, Proteon, University of Tennessee at Knoxville, Cornell University, et Rensselaer Polytechnic Institute, novembre 1987.

[9] Organisation Internationale de Normalisation (ISO), "Systèmes de traitement de l'information – Interconnexion des systèmes ouverts – Spécification de la notation de syntaxe abstraite n° 1 (ASN.1)", Norme Internationale 8824, décembre 1987.

[10] Organisation Internationale de Normalisation (ISO), "Systèmes de traitement de l'information – Interconnexion des systèmes ouverts – Spécification des règles de codage de base pour la Notation de syntaxe abstraite n° 1 (ASN.1)", Norme Internationale 8825, décembre 1987.

[11] Postel, J., "Protocole de datagramme d'utilisateur", RFC 768, USC/Information Sciences Institute, novembre 1980.

8 Considérations sur la sécurité

Les questions de sécurité ne sont pas discutées dans le présent mémo.

9 Adresse des auteurs

Jeffrey D. Case
SNMP Research
P.O. Box 8593
Knoxville, TN 37996-4800
USA
tél : (615) 573-1434
mél : case@CS.UTK.EDU

Mark Fedor
Performance Systems International
Rensselaer Technology Park
125 Jordan Road
Troy, NY 12180
tél : (518) 283-8860
mél : fedor@patton.NYSER.NET

Martin Lee Schoffstall
Performance Systems International
Rensselaer Technology Park
165 Jordan Road
Troy, NY 12180
tél : (518) 283-8860
mél : schoff@NISC.NYSER.NET

James R. Davin
MIT Laboratory for Computer Science, NE43-507
545 Technology Square
Cambridge, MA 02139
tél : (617) 253-6020
mél : jrd@ptt.lcs.mit.edu