Groupe de travail Réseau
Request for Comments : 3411
STD : 62
RFC rendue obsolète : 2571
Catégorie : Norme

D. Harrington, Enterasys Networks
R. Presuhn, BMC Software, Inc.
B. Wijnen, Lucent Technologies
décembre 2002
Traduction Claude Brière de L'Isle

Architecture de description des cadres de gestion du protocole simple de gestion de réseau (SNMP)

Statut de ce mémoire
Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document décrit une architecture de description des cadres de travail de gestion du protocole simple de gestion de réseau (SNMP). L'architecture est conçue comme modulaire pour permettre l'évolution dans le temps des normes du protocole SNMP. Les portions majeures de l'architecture sont un moteur SNMP qui contient un sous système de traitement de message, un sous système de sécurité et un sous système de contrôle d'accès, et éventuellement plusieurs applications SNMP qui fournissent un traitement fonctionnel spécifique des données de gestion. Le présent document rend obsolète la RFC 2571.

Table des Matières

1. Introduction

1.1 Généralités

Le présent document définit un vocabulaire pour décrire le cadre de travail de la gestion SNMP, et une architecture pour décrire les principales portions du cadre de gestion SNMP.

Le présent document ne constitue pas une introduction générale à SNMP. D'autres documents et ouvrages peuvent fournir une bien meilleure introduction à SNMP. Ce document ne fait pas non plus l'historique de SNMP. On peut aussi trouver cela dans d'autres documents et ouvrages.

La section 1 décrit les objectifs, les raisons et les décisions de la conception de cette architecture.

La section 2 décrit les divers types de documents qui définissent le (les éléments du) cadre de SNMP, comment ils s'assemblent dans cette architecture. Il fournit aussi un canevas minimal des documents qui ont précédemment défini le cadre de SNMP.

La section 3 détaille le vocabulaire de cette architecture et de ses éléments. Cette section est importante pour la compréhension des sections suivantes, et pour comprendre les documents qui ont été écrits pour remplir le cadre architectural.

La section 4 décrit les primitives utilisées pour les interfaces de service abstrait entre les divers sous systèmes, les modèles et les applications au sein de cette architecture.

La section 5 définit une collection d'objets gérés utilisés pour mettre en œuvre les entités SNMP au sein de cette architecture.

Les sections 6, 7, 8, 9, 10 et 11 sont de nature administrative.

L'appendice A contient des lignes directrices pour les concepteurs des modèles qui devraient entrer dans le cadre de cette architecture.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" dans ce document sont à interpréter comme décrit dans la [RFC2119].

1.2 SNMP

Un système de gestion SNMP contient :
- plusieurs nœuds (éventuellement beaucoup) avec chacun une entité SNMP contenant des applications de réponse de commandes et génératrices de notifications, qui ont accès aux instruments de gestion (traditionnellement appelés des agents) ;
- au moins une entité SNMP contenant un générateur de commandes et/ou des applications de réception de notification (traditionnellement appelé un gestionnaire) et,
- un protocole de gestion, utilisé pour convoyer les informations de gestion entre les entités SNMP.

Les entités SNMP qui exécutent les applications de générateur de commandes et de receveur de notifications surveillent et contrôlent les éléments gérés. Les éléments gérés sont des appareils tels que les hôtes, les routeurs, serveurs de terminaux, etc., qui sont surveillés et contrôlés via l'accès à leurs informations de gestion.

L'objet du présent document est de définir une architecture qui puisse évoluer pour réaliser une gestion effective dans diverses configurations et environnements. L'architecture a été conçue pour satisfaire aux besoins de mise en œuvre :
- d'entités SNMP minimales avec des applications de réponse de commandes et/ou générateur de notification (traditionnellement appelées agents SNMP),
- des entités SNMP avec des applications de transmission mandataires (traditionnellement appelées agents mandataires SNMP),
- des entités SNMP pilotées par ligne de commande avec des applications de générateur de commande et/ou de receveur de notification (traditionnellement appelées gestionnaires de ligne de commande SNMP),
- des entités SNMP avec générateur de commande et/ou receveur de notification, plus des applications de réponse de commande et/ou générateur de notification (traditionnellement appelées gestionnaire SNMP de mi-niveau ou entités à double rôle),
- des entités SNMP avec générateur de commande et/ou receveur de notification et éventuellement d'autres types d'applications pour un potentiel très grand nombre de nœuds gérés (traditionnellement appelés stations de gestion (réseau)).

1.3 Objectifs de cette architecture

Cette architecture recherche les objectifs suivants :
- Utiliser autant que possible les matériaux existants. Elle s'appuie fortement sur les travaux antérieurs, connus de façon informelle sous les noms de SNMPv2u et SNMPv2*, fondés à leur tour sur SNMPv2p.
- Satisfaire le besoin d'une prise en charge sécurisée de SET, ce qui est considéré comme le manque le plus important de SNMPv1 et SNMPv2c.
- Rendre possible le placement de portions de l'architecture dans la voie de la normalisation, même si le consensus n'a pas été obtenu sur toutes les parties.
- Définir une architecture qui permette la longévité du cadre SNMP qui a été et sera défini.
- Garder SNMP aussi simple que possible.
- Rendre relativement peu coûteux le déploiement d'une mise en œuvre de conformité minimale.
- Rendre possible la mise à niveau de portions de SNMP lorsque de nouvelles approches deviennent disponibles, sans briser la totalité du cadre SNMP.
- Rendre possible la prise en charge des dispositifs nécessaires dans les grands réseaux, mais mettre les dépenses de prise en charge d'un dispositif en relation directe avec la prise en charge du dispositif.

1.4 Exigences de sécurité de cette architecture

Plusieurs des menaces classiques contre les protocoles réseau sont applicables au problème de la gestion et seront donc applicables à tout modèle de sécurité utilisé dans un cadre de gestion SNMP. Les autres menaces ne sont pas applicables au problème de la gestion. Ce paragraphe expose les menaces principales, les menaces secondaires, et les menaces qui sont de moindre importance.

Les principales menaces contre lesquelles tout modèle de sécurité utilisé au sein de cette architecture DEVRAIT fournir protection sont :

Modification des informations
La menace de modification est le danger que quelque entité non autorisée puisse altérer les messages SNMP en transit générés au nom d'un principal autorisé de façon à effectuer des opérations de gestion non autorisées, y compris la falsification de la valeur d'un objet.

Usurpation d'identité *(Masquerade)*
La menace d'usurpation d'identité est le danger que des opérations de gestion non autorisées pour un principal puissent être

tentées en empruntant l'identité d'un autre principal qui a les autorisations appropriées.

Les menaces secondaires contre lesquelles tout modèle de sécurité au sein de cette architecture DEVRAIT fournir protection sont :

Modification de flux de message
Le protocole SNMP est normalement fondé sur un service de transport sans connexion qui peut fonctionner sur tout service de sous-réseau. La réorganisation, le retard ou la répétition des messages peuvent survenir et surviennent à travers le fonctionnement naturel de nombreux services de sous réseau de cette sorte. La menace de modification de flux de message est le danger que des messages soit réorganisés, retardés ou répétés de façon malveillante dans une mesure supérieure à celle qui peut survenir par le fonctionnement naturel d'un service de sous réseau, afin d'effectuer des opérations de gestion non autorisées.

Divulgation
La menace de divulgation est le danger d'espionnage des échanges entre les moteurs SNMP. La protection contre cette menace peut être exigée selon une politique locale.

Il y a au moins deux menaces contre lesquelles un modèle de sécurité au sein de cette architecture n'a pas besoin d'être protégé, dans la mesure où elles sont réputées être de moindre importance dans ce contexte :

Déni de service
Un modèle de sécurité n'a pas besoin de tenter de contrer une large gamme d'attaques par lesquelles le service est dénié aux utilisateurs autorisés. Bien sûr, de telles attaques de déni de service sont dans de nombreux cas indistinguables du type de défaillance de réseau à laquelle tout protocole de gestion viable doit normalement savoir faire face.

Analyse de trafic
Un modèle de sécurité n'a pas besoin de tenter de contrer les attaques d'analyse de trafic. De nombreux schémas de trafic sont prévisibles – les entités peuvent être gérées d'une façon régulière par un nombre relativement faible de stations de gestion – et donc, il n'y a pas d'avantage significatif à se protéger contre l'analyse de trafic.

1.5 Décisions de conception

Diverses décisions de conception ont été prise pour la réalisation des objectifs de l'architecture et des exigences de sécurité :

- Architecture
Une architecture devrait être définie comme identifiant les limites conceptuelles entre les documents. Les sous systèmes définis devraient décrire les services abstraits fournis par les portions spécifiques d'un cadre SNMP. Les interfaces de service abstraites, comme décrites par les primitives de service, définissent les limites abstraites entre les documents, et les services abstraits sont fournis par les sous systèmes conceptuels d'un cadre SNMP.

- Documents autosuffisants
Les éléments de procédure plus les objets de MIB qui sont nécessaires pour traiter une portion spécifique d'un cadre SNMP devraient être définis dans le même document, et autant que possible, ne devraient pas être référencés dans d'autres documents. Cela permet que des morceaux soient conçus et documentés comme des parties indépendantes et autosuffisantes, qui soient cohérentes avec l'approche générale de module de MIB SNMP. Comme les portions de SNMP changent avec le temps, les documents qui décrivent les autres portions de SNMP ne sont pas directement impactés. Cette modularité permet, par exemple, que les modèles de sécurité, les mécanismes d'authentification et de confidentialité, et les formats de message soient mis à niveau et complétés lorsque le besoin s'en fait sentir. Les documents autosuffisants peuvent avancer sur la voie de la normalisation à leur rythme propre.

Cette modularité des spécifications n'est pas destinée à être interprétée comme imposant d'exigence spécifique à la mise en œuvre.

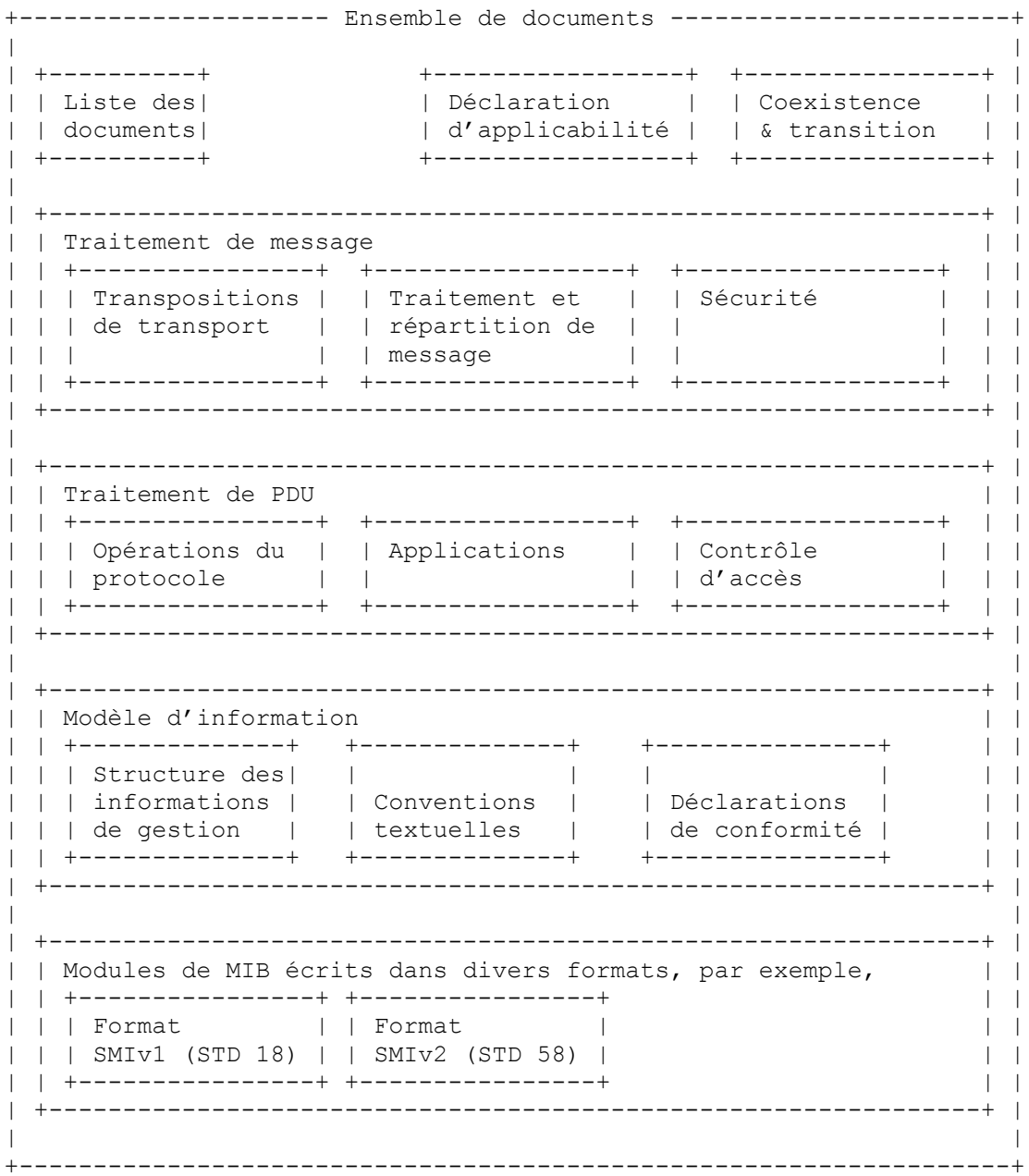
- Menaces
Le modèle de sécurité dans le sous système de sécurité DEVRAIT protéger contre les menaces principales et secondaires : la modification d'informations, l'usurpation d'identité, la modification de flux de message et la divulgation. Il n'est pas nécessaire qu'il protège contre les dénis de service et l'analyse de trafic.

- Configuration à distance
Les sous systèmes de sécurité et de contrôle d'accès ajoutent tout un nouvel ensemble de paramètres de configuration SNMP. Le sous système de sécurité exige aussi de fréquents changements de secrets sur les diverses entités SNMP. Pour permettre de les déployer dans un grand environnement de fonctionnement, ces paramètres SNMP doivent être configurables à distance.

- Complexité contrôlée
Il est reconnu que les producteurs d'appareils à gestion simple veulent garder les ressources utilisées par SNMP à un minimum. En même temps, il y a un besoin de configurations plus complexes qui peuvent dépenser plus de ressources pour SNMP et donc apporter des fonctionnalités supplémentaires. La conception du système essaye de conserver l'équilibre entre les exigences opposées de ces deux environnements et permet que les environnements plus complexes fassent des extensions logiques à l'environnement simple.

2. Généralités sur la documentation

La figure suivante montre l'ensemble des documents qui constituent ensemble l'architecture SNMP.



Chacun de ces documents peut être remplacé ou complété. Ce document d'architecture décrit spécifiquement comment de

nouveaux documents trouvent leur place dans l'ensemble, dans le domaine du traitement du message et de la PDU.

2.1 Cadre pour les documents

Un ou plusieurs documents peuvent être écrits pour décrire comment les ensembles de documents pris ensemble forment des cadres spécifiques. La configuration d'ensembles de documents peut changer avec le temps, de sorte que le "cadre" devrait être conservé dans un document séparé des documents de normalisation eux-mêmes.

Un exemple d'un tel cadre est "Introduction et déclarations d'applicabilité pour le cadre de gestion des normes de l'Internet" [RFC3410].

2.2 Déclaration d'applicabilité

SNMP est utilisé dans des réseaux dont la taille et la complexité sont très diverses, par des organisations dont les exigences de gestion sont très différentes. Certains modèles seront conçus pour régler des problèmes de gestion spécifiques, tels que la sécurité des messages.

Un ou plusieurs documents peuvent être rédigés pour décrire les environnements auxquels certaines versions de SNMP ou certains modèles au sein de SNMP s'appliqueraient de façon appropriée, et ceux auxquels l'application d'un modèle donné pourrait être inappropriée.

2.3 Coexistence et transition

L'objet d'une architecture évolutive est de permettre aux nouveaux modèles de remplacer ou compléter les modèles existants. Les interactions entre modèles pourraient déboucher sur des incompatibilités, des "trous" dans la sécurité, et autres effets indésirables.

L'objet des documents de coexistence est de détailler les anomalies reconnues et de décrire les comportements exigés et recommandés pour résoudre les interactions entre modèles au sein de l'architecture.

Les documents de coexistence peuvent être préparés séparément des documents de définition de modèles, pour décrire et résoudre les anomalies d'interaction entre la définition d'un modèle et une ou plusieurs autres définitions de modèles.

De plus, les recommandations pour les transitions entre les modèles peuvent aussi être décrites, soit dans un document de coexistence, soit dans un document distinct.

Un exemple de document de coexistence est la [RFC2576], "Coexistence entre les version 1, version 2 et version 3 du cadre de gestion de réseau de l'Internet".

2.4 Transpositions de transport

Les messages SNMP sont envoyés sur divers transports. L'objet des documents de transposition de transport est de définir comment est faite la transposition entre SNMP et le transport.

2.5 Traitement de message

Un document de modèle de traitement de message définit un format de message, qui est normalement identifié par un champ Version dans un en-tête de message SNMP. Le document peut aussi définir un module de MIB à utiliser dans le traitement de message et pour l'instrumentation des interactions spécifiques de la version.

Un moteur SNMP comporte un ou plusieurs modèles de traitement de message, et peut donc prendre en charge l'envoi et la réception de plusieurs versions de messages SNMP.

2.6 Sécurité

Certains environnements exigent des interactions de protocole sécurisées. La sécurité est normalement appliquée à deux niveaux différents :

- dans la transmission/réception des messages, et
- dans le traitement du contenu des messages.

Pour les besoins du présent document, "sécurité" se réfère à la sécurité au niveau message ; "contrôle d'accès" se réfère à la

sécurité appliquée aux opérations du protocole.

L'authentification, le chiffrement, et la vérification de l'exactitude temporelle sont des fonctions communes de la sécurité de niveau message.
Un document de sécurité décrit un modèle de sécurité, les menaces contre lesquelles protège le modèle, les buts du modèle de sécurité, les protocoles qu'il utilise pour satisfaire à ces objectifs, et il peut définir un module de MIB pour décrire les données utilisées durant le traitement, et permettre la configuration à distance des paramètres de sécurité de niveau message, tels que les clés.
Un moteur SNMP peut prendre en charge concurremment plusieurs modèles de sécurité.

2.7 Contrôle d'accès

Durant le traitement, il peut être exigé de contrôler l'accès aux objets gérés pour les opérations.
Un modèle de contrôle d'accès définit des mécanismes pour déterminer si l'accès à un objet géré devrait être permis. Un modèle de contrôle d'accès peut définir un module de MIB utilisé durant le traitement et pour permettre la configuration à distance des politiques de contrôle d'accès.

2.8 Opérations du protocole

Les messages SNMP encapsulent une unité de données de protocole (PDU) SNMP. Les PDU SNMP définissent les opérations effectuées par le moteur SNMP receveur. L'objet d'un document d'opérations de protocole est de définir les opérations du protocole par rapport au traitement des PDU. Chaque PDU appartient à une ou plusieurs classes de PDU définies ci-dessous :

1) Classe Lecture *(Read)* : La classe Lecture contient les opérations de protocole qui restituent les informations de gestion. Par exemple, la [RFC3416] définit les opérations de protocole suivantes pour la classe Lecture : GetRequest-PDU, GetNextRequest-PDU et GetBulkRequest-PDU.

2) Classe Écriture *(Write)* : La classe Écriture contient les opérations de protocole qui essayent de modifier les informations de gestion. Par exemple, la [RFC3416] définit l'opération de protocole suivante pour la classe Écriture : SetRequest-PDU.

3) Classe Réponse (*response*) : La classe Réponse contient les opérations de protocole qui sont envoyées en réponse à une demande précédente. Par exemple, la [RFC3416] définit les opérations suivantes pour la classe Réponse : Response-PDU, Report-PDU.

4) Classe Notification : La classe Notification contient les opérations de protocole qui envoient une notification à une application de receveur de notification. Par exemple, la [RFC3416] définit les opérations suivantes pour la classe Notification : Trapv2-PDU, InformRequest-PDU.

5) Classe Interne *(internal)* : La classe Interne contient les opérations de protocole qui sont échangées en interne entre les moteurs SNMP. Par exemple, la [RFC3416] définit les opérations suivantes pour la classe Interne : Report-PDU.

Les cinq classes précédentes se fondent sur les propriétés fonctionnelles d'une PDU. Il est aussi utile de classer les PDU sur la base de la réponse attendue :

6) Classe Confirmée *(confirmed)* : La classe Confirmée contient toutes les opérations de protocole qui font que le moteur SNMP receveur renvoie une réponse. Par exemple, la [RFC3416] définit les opérations suivantes pour la classe Confirmée : GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, et InformRequest-PDU.

7) Classe Non confirmée *(non confirmed)* : La classe Non confirmée contient toutes les opérations de protocole qui ne reçoivent pas d'accusé de réception. Par exemple, la [RFC3416] définit les opérations suivantes pour la classe Non confirmée : Report-PDU, Trapv2-PDU et GetResponse-PDU.

Un document d'application définit les opérations de protocole qui sont prises en charge par l'application.

2.9 Applications

Une entité SNMP comporte normalement un certain nombre d'applications. Les applications utilisent les services d'un moteur SNMP pour accomplir des tâches spécifiques. Elles coordonnent le traitement des opérations d'informations de gestion, et peuvent utiliser les messages SNMP pour communiquer avec d'autres entités SNMP.

Un document d'application décrit l'objet d'une application, les services requis par le moteur SNMP associé, et les opérations de protocole ainsi que le modèle informationnel qu'utilise l'application pour effectuer les opérations de gestion.
Un document d'application définit quel ensemble de documents est utilisé pour définir spécifiquement la structure des information de gestion, les conventions textuelles, les exigences de conformité, et les opérations prises en charge par l'application.

2.10 Structure des informations de gestion

Les informations de gestion sont vues comme une collection d'objets gérés, résidant dans une mémoire d'informations virtuelle, appelée la base de données d'informations de gestion (MIB, *Management Information Base*). Les collections d'objets qui s'y rapportent sont définies dans les modules de MIB.
C'est l'objet d'une structure de document d'informations de gestion d'établir la notation pour définir les objets, modules et autres éléments des informations gérées.

2.11 Conventions textuelles

Lors de la conception d'un module de MIB, il est souvent utile de définir de nouveaux types similaires à ceux définis dans la SMI, mais avec une sémantique plus précise, ou qui ont une sémantique associée particulière. Ces nouveaux types définis sont appelés des conventions textuelles (TC, *textual conventions*) et peuvent être définis dans des documents séparés, ou au sein d'un module de MIB.

2.12 Déclarations de conformité

Il peut être utile de définir les limites inférieures de mise en œuvre acceptable, avec le niveau réel de mise en œuvre réalisé. C'est l'objet du document de déclaration de conformité de définir la notation utilisée à cette fin.

2.13 Modules de base de données d'information de gestion

Les documents de MIB décrivent les collections d'objets gérés qui instrumentent certains aspects d'un nœud géré.

2.13.1 Les MIB d'instrumentation SNMP

Un document de MIB SNMP peut définir une collection d'objets gérés qui instrumentent le protocole SNMP lui-même. De plus, les modules de MIB peuvent être définis au sein de documents qui décrivent des portions de l'architecture SNMP, tels que les documents pour les modèles de traitement de message, les modèles de sécurité, etc., pour les besoins de l'instrumentation de ces modèles, et pour permettre leur configuration à distance.

2.14 Documents cadre de SNMP

Cette architecture est conçue pour permettre une évolution ordonnée de portions des cadres SNMP.
Dans tout le reste du présent document, le terme "sous système" se réfère à une spécification abstraite et incomplète d'une portion d'un cadre, qui est précisée par une spécification de modèle.
Un "modèle" décrit un concept spécifique d'un sous système, définit des contraintes et règles supplémentaires pour la conformité au modèle. Un modèle est suffisamment détaillé pour rendre possible la mise en œuvre de la spécification.
Une "mise en œuvre" est une instanciation d'un sous système, se conformant à un ou plusieurs modèles spécifiques.
SNMP version 1 (SNMPv1) est le cadre original de gestion de réseau selon les normes de l'Internet, tel que décrit dans les [RFC1155], [RFC1157] et [RFC1212].
SNMP version 2 (SNMPv2) est le cadre de SNMPv2 tel que dérivé du cadre de SNMPv1. Il est décrit dans le STD 58, constitué des [RFC2578], [RFC2579] et [RFC2580], et dans le STD 62, constitué des [RFC3416], [RFC3417] et [RFC3418]. SNMPv2 n'a pas de définition de message.
SNMP version 2 fondé sur la communauté (SNMPv2c), est un cadre SNMP expérimental qui complète le cadre SNMPv2, tel que décrit dans la [RFC1901]. Il ajoute le format de message SNMPv2c, qui est similaire au format de message SNMPv1.

SNMP version 3 (SNMPv3), est un cadre SNMP extensible qui complète le cadre SNMPv2 en prenant en charge ce qui suit :
- un nouveau format de message SNMP,
- la sécurité pour les messages,
- le contrôle d'accès, et
- la configuration à distance des paramètres SNMP.

D'autres cadres SNMP, c'est à dire, d'autres configurations de sous systèmes mis en œuvre, devraient à l'avenir être en

cohérence avec cette architecture.

3. Éléments de l'architecture

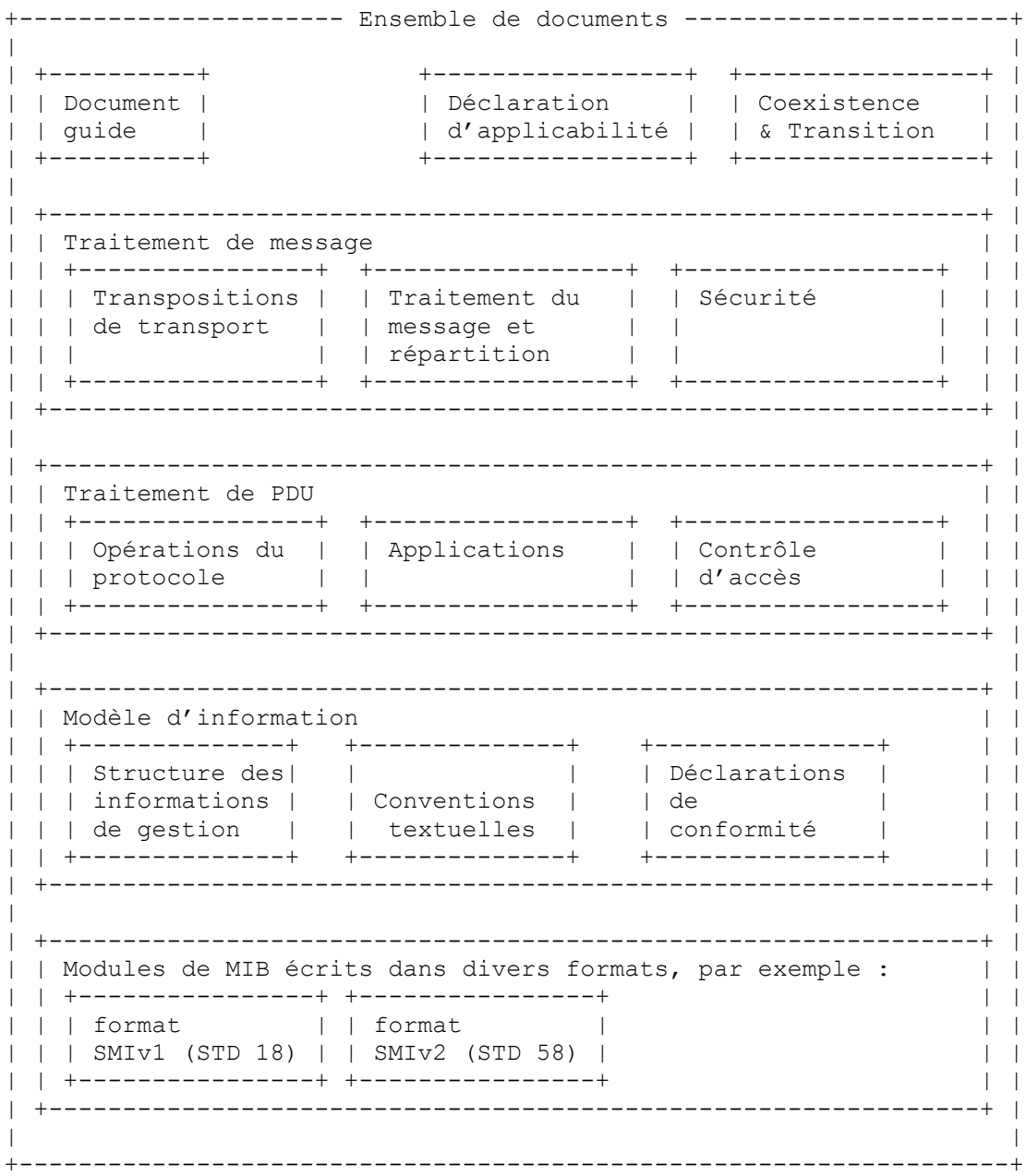
La présente section décrit les divers éléments de l'architecture et leur dénomination. Il y a trois sortes de désignations :
1) la désignation des entités,
2) la désignation des identités, et
3) la désignation des informations de gestion.

Cette architecture définit aussi certains noms pour d'autres constructions qui sont utilisées dans la documentation.

3.1 Dénomination des entités

Une entité SNMP est une mise en œuvre de cette architecture. Chacune de ces entités SNMP consiste en un moteur SNMP et une ou plusieurs applications associées.

La figure suivante montre les détails d'une entité SNMP et de ses composants internes.



3.1.1 Moteur SNMP

Un moteur SNMP fournit des services pour l'envoi, la réception, l'authentification et le chiffrement des messages, et pour contrôler l'accès aux objets gérés. Il y a une association biunivoque entre un moteur SNMP et l'entité SNMP qui le contient.

Le moteur contient :
1) un répartiteur *(Dispatcher)*,
2) un sous système de traitement de message,
3) un sous système de sécurité, et
4) un sous système de contrôle d'accès.

3.1.1.1 snmpEngineID

Au sein d'un domaine administratif, un snmpEngineID *(identifiant de moteur SNMP)* est l'identifiant univoque et sans ambiguïté d'un moteur SNMP. Comme il y a une association biunivoque entre les moteurs SNMP et les entités SNMP, il identifie aussi de façon univoque et non ambiguë l'entité SNMP au sein de ce domaine administratif. Noter qu'il est possible que des entités SNMP dans des domaines administratifs différents aient la même valeur pour snmpEngineID. La fédération de domaines administratifs peut nécessiter l'allocation de nouvelles valeurs.

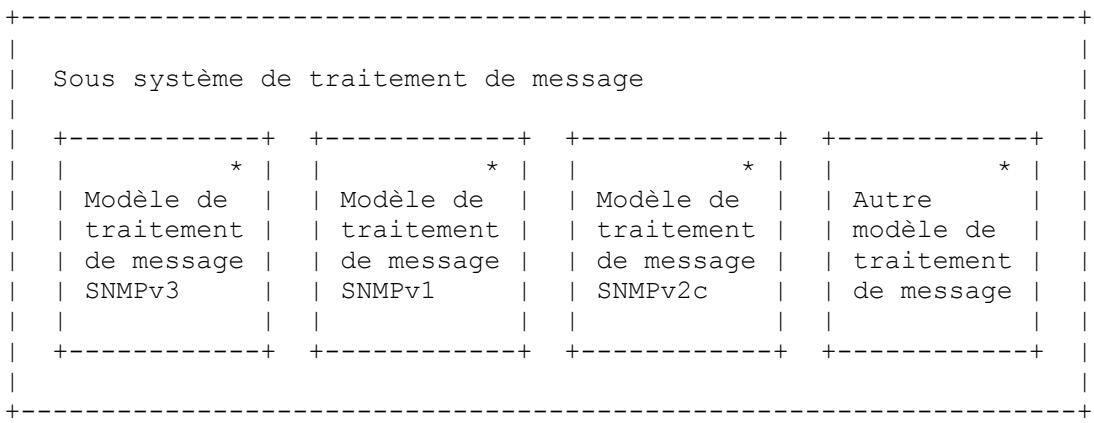
3.1.1.2 Répartiteur

Il y a un seul répartiteur dans un moteur SNMP. Il permet la prise en charge en concurrence de plusieurs versions de messages SNMP dans le moteur SNMP. Il le fait en :
- envoyant et recevant des messages SNMP du/vers le réseau,
- déterminant la version d'un message SNMP et en interagissant avec le modèle de traitement de message correspondant,
- fournissant une interface abstraite aux applications SNMP pour la livraison d'une PDU à une application.
- fournissant une interface abstraite aux applications SNMP qui leur permet d'envoyer une PDU à une entité SNMP distante.

3.1.1.3 Sous système de traitement de message

Le sous système de traitement de message est chargé de préparer les messages à l'envoi et d'extraire les données des messages reçus.

Le sous système de traitement de message contient potentiellement plusieurs modèles de traitement de message comme indiqué à la figure suivante.



* Un ou plusieurs modèles de traitement de message peuvent être présents.

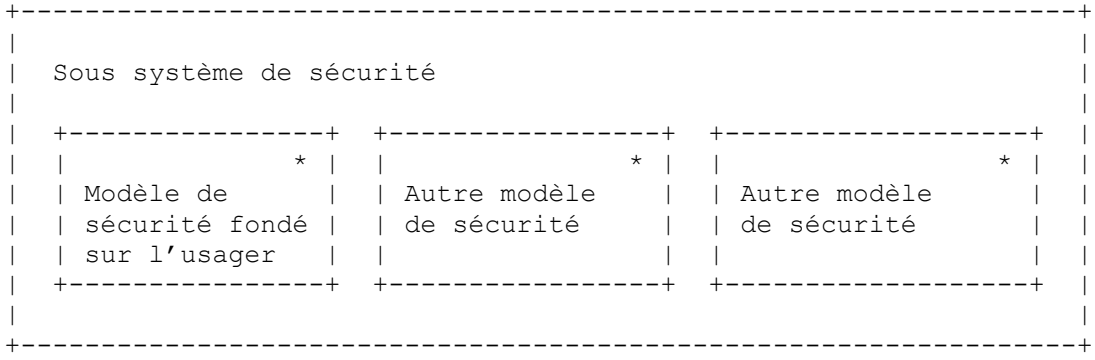
3.1.1.3.1 Modèle de traitement de message

Chaque modèle de traitement de message définit le format d'une version particulière d'un message SNMP et coordonne la préparation et l'extraction de chacun de ces formats de message spécifiques de la version.

3.1.1.4 Sous système de sécurité

Le sous système de sécurité fournit des services de sécurité tels que l'authentification et la confidentialité des messages et peut contenir plusieurs modèles de sécurité, comme indiqué dans la figure suivante.

* Un ou plusieurs modèles de sécurité peuvent être présents.



3.1.1.4.1 Modèle de sécurité

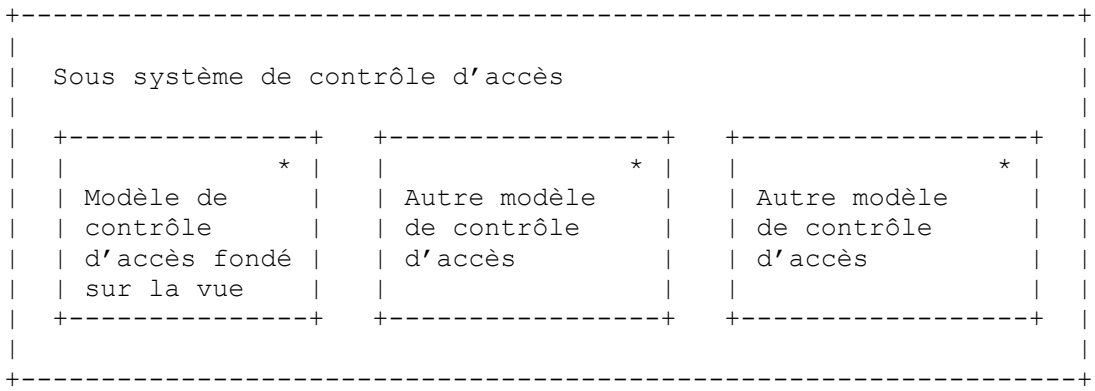
Un modèle de sécurité spécifie les menaces contre lesquelles il protège, les buts de ses services, et les protocoles de sécurité utilisés pour fournir des services de sécurité tels que l'authentification et la confidentialité.

3.1.1.4.2 Protocole de sécurité

Un protocole de sécurité spécifie les mécanismes, les procédures, et les objets de MIB utilisés pour fournir un service de sécurité tel que l'authentification ou la confidentialité.

3.1.2 Sous système de contrôle d'accès

Le sous système de contrôle d'accès fournit des services d'autorisation au moyen d'un ou plusieurs (*) modèles de contrôle d'accès.



3.1.2.1 Modèle de contrôle d'accès

Un modèle de contrôle d'accès définit une fonction particulière de décision d'accès afin de prendre en charge les décisions concernant les droits d'accès.

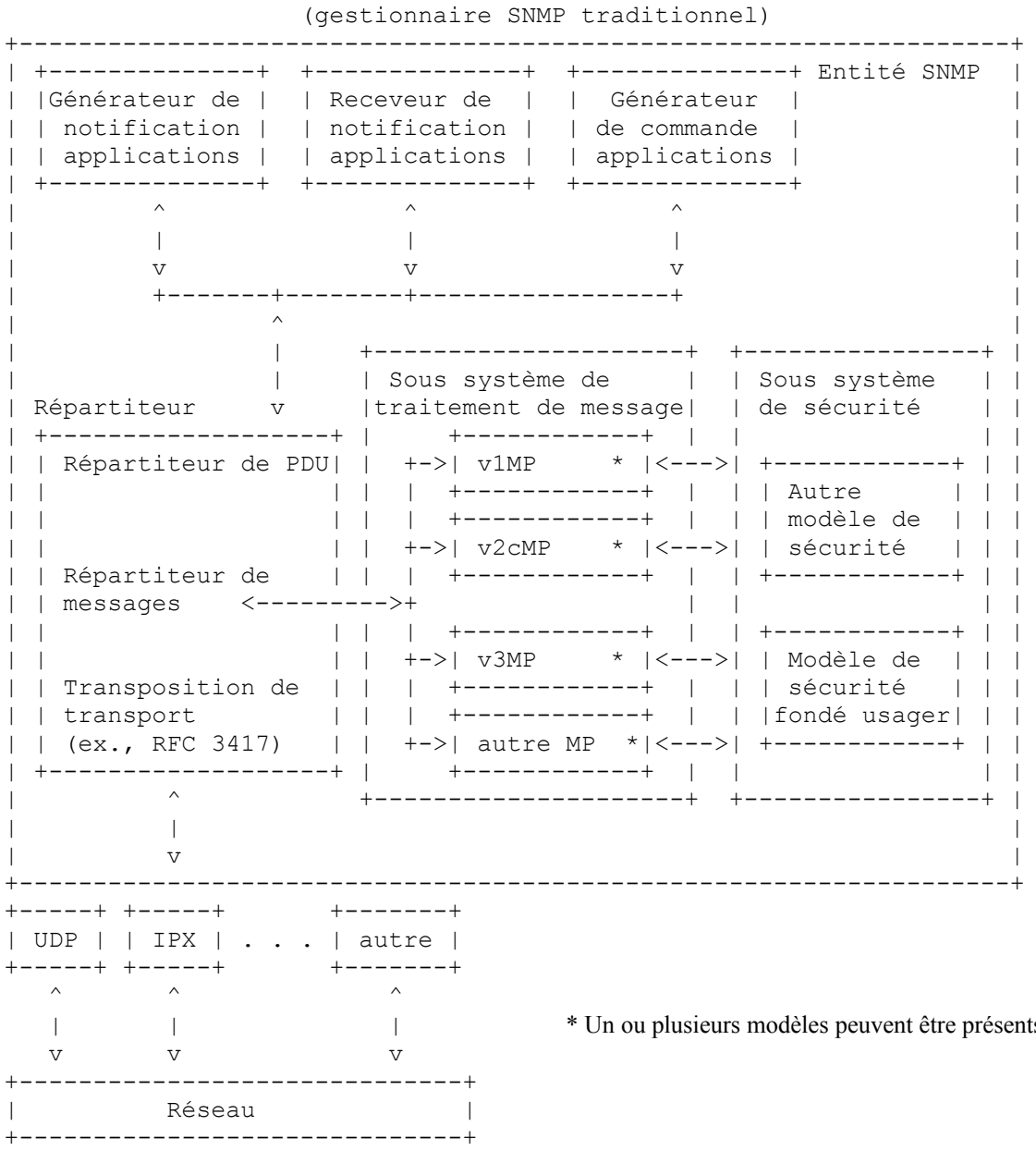
3.1.3 Applications

Il y a plusieurs types d'applications, parmi lesquels :
- des générateurs de commandes, qui surveillent et manipulent les données de gestion,
- les répondeurs de commande, qui donnent accès aux données de gestion,
- les générateurs de notification, qui initient les messages asynchrones,
- les receveurs de notification, qui traitent les messages asynchrones, et
- les mandataires transmetteurs, qui transmettent les messages entre les entités.

Ces applications font usage des services fournis par le moteur SNMP.

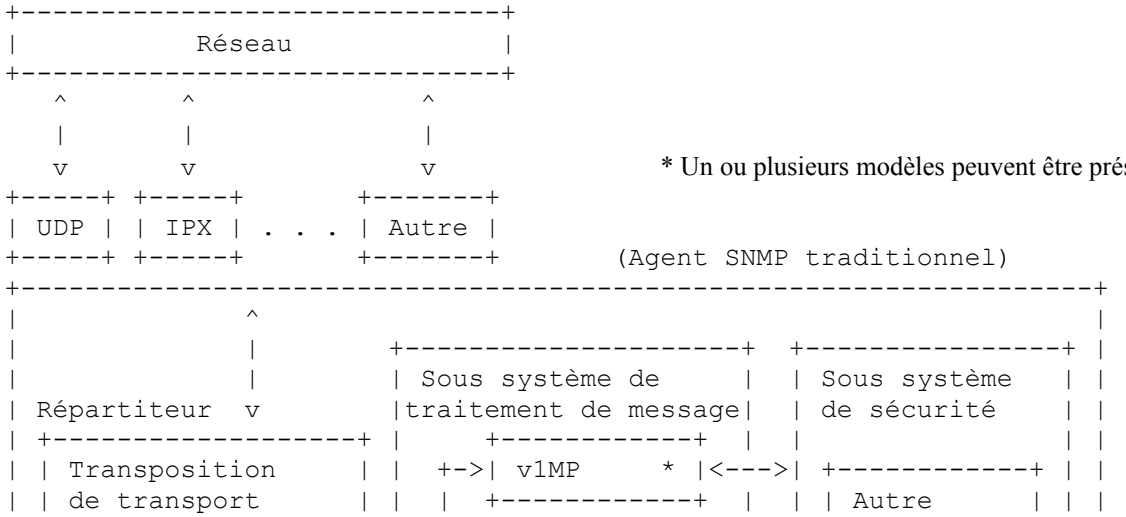
3.1.3.1 Gestionnaire SNMP

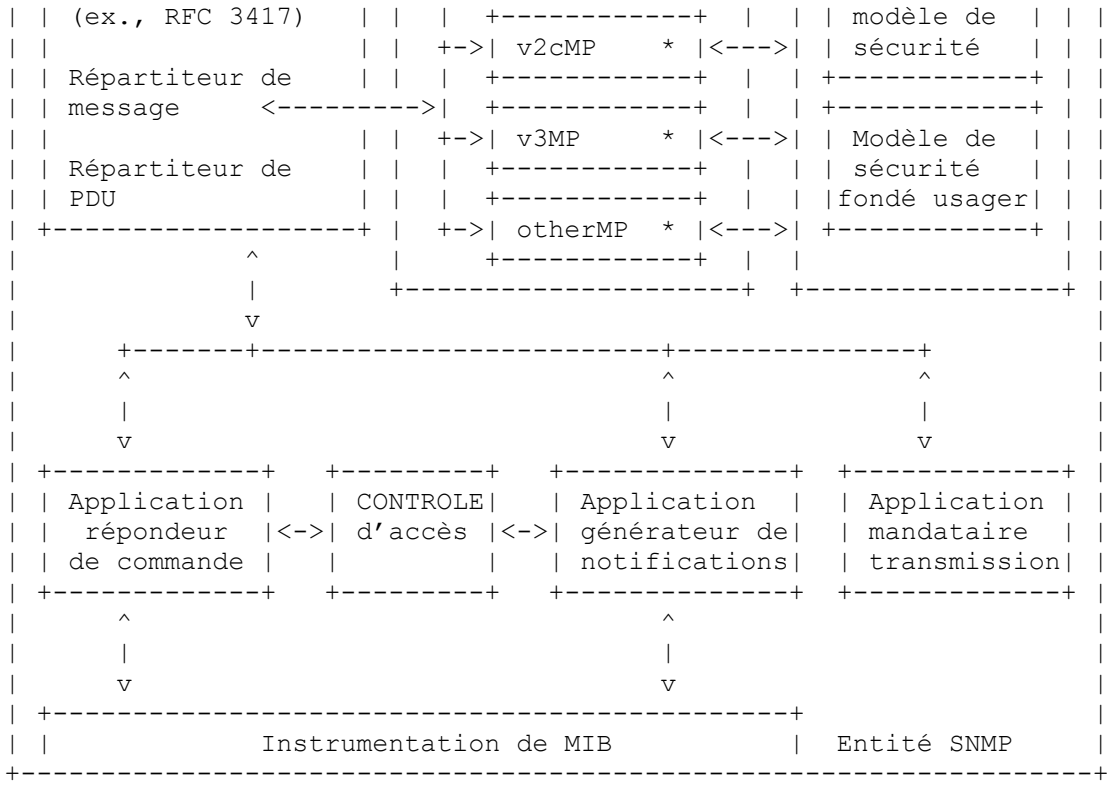
C'est une entité SNMP qui contient une ou plusieurs applications de générateurs de commandes et/ou de receveur de notifications (avec leur moteur SNMP associé) et a traditionnellement été appelée un gestionnaire SNMP.



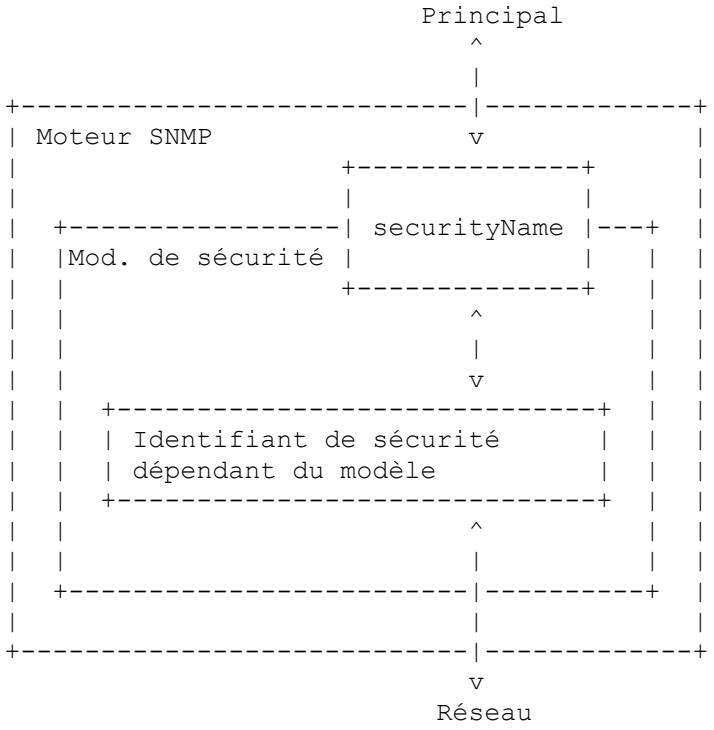
3.1.3.2 Agent SNMP

Une entité SNMP contenant une ou plusieurs applications de répondeur de commande et/ou de générateur de notification (avec leur moteur SNMP associé) a traditionnellement été appelée un agent SNMP.





3.2 Désignation des identités



3.2.1 Principal

Un principal est le "qui" au nom duquel les services sont fournis, ou le traitement a lieu.

Un principal peut être, entre autres choses, un individu qui joue un rôle particulier, un ensemble d'individus dont chacun joue un rôle particulier, une application ou un ensemble d'applications, et leurs combinaisons.

3.2.2 securityName

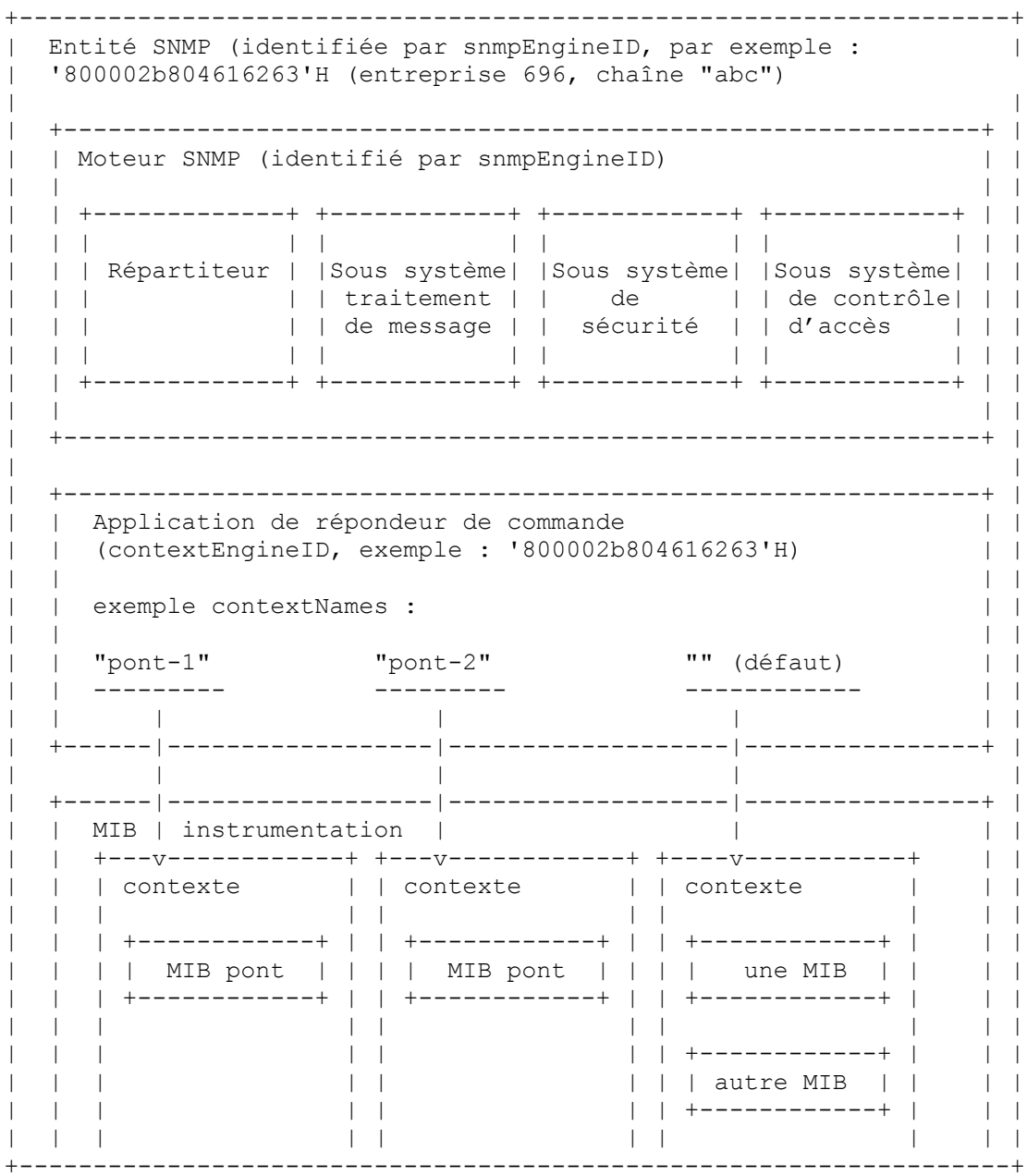
Un securityName *(nom de sécurité)* est une chaîne lisible par l'homme représentant un principal. Il a un format indépendant du modèle, et peut être utilisé en dehors d'un modèle de sécurité particulier.

3.2.3 Identifiant de sécurité dépendant du modèle

Un identifiant de sécurité dépendant du modèle est la représentation spécifique du modèle d'un securityName au sein d'un modèle de sécurité particulier.
Les identifiants de sécurité dépendants du modèle peuvent être ou non lisibles par l'homme, et ont une syntaxe dépendante du modèle. Les exemples incluent des noms de communauté, et des noms d'utilisateur.
La transformation des identifiants de sécurité dépendants du modèle en securityNames et vice versa est de la responsabilité du modèle de sécurité pertinent.

3.3 Désignation des informations de gestion

Les informations de gestion résident dans une entité SNMP où une application de répondeur de commande a un accès local à des contextes éventuellement multiples. Cette application utilise un contextEngineID égal au snmpEngineID de son moteur SNMP associé.



3.3.1 Contexte SNMP

Un contexte SNMP, ou juste "contexte" en bref, est une collection d'informations de gestion accessibles par une entité SNMP. Un élément d'informations de gestion peut exister dans plus d'un contexte. Une entité SNMP peut accéder à de nombreux contextes.

Normalement, il y a de nombreuses instances de chaque type d'objet géré au sein d'un domaine de gestion. Pour simplifier, la méthode pour identifier les instances spécifiées par le module de MIB ne permet pas de distinguer chaque instance parmi l'ensemble de toutes les instances d'un domaine de gestion ; elle permet plutôt que chaque instance soit identifiée seulement dans une certaine portée ou "contexte", où il y a nombre de ces contextes au sein du domaine de gestion. Souvent, un contexte est un appareil physique, ou peut-être, un appareil logique, bien qu'un contexte puisse aussi englober plusieurs appareils, ou un sous ensemble d'un seul appareil, ou même un sous ensemble de plusieurs appareils, mais un contexte est toujours défini comme sous ensemble d'une seule entité SNMP. Donc, pour identifier un élément individuel d'informations de gestion dans le domaine de gestion, son contextName et son contextEngineID doivent être identifiés en plus de son type d'objet et son instance.

Par exemple, le type d'objet géré ifDescr [RFC2863], est défini comme la description d'une interface réseau. Pour identifier la description de la première interface réseau de l'appareil X, quatre éléments d'information sont nécessaires : le snmpEngineID de l'entité SNMP qui fournit l'accès aux informations de gestion à l'appareil X, le contextName (appareil X), le type d'objet géré (ifDescr), et l'instance ("1").

Chaque contexte a (au moins) une identification unique au sein du domaine de gestion. Le même élément d'informations de gestion peut exister dans plusieurs contextes. Un élément d'informations de gestion peut avoir plusieurs identifications uniques. Cela se produit lorsque un élément d'informations de gestion existe dans plusieurs contextes, et cela se produit aussi lorsque un contexte a plusieurs identifications uniques.

La combinaison d'un contextEngineID et d'un contextName identifie sans ambiguïté un contexte au sein d'un domaine administratif ; noter qu'il peut y avoir plusieurs combinaisons uniques de contextEngineID et contextName qui identifient sans ambiguïté le même contexte.

3.3.2 contextEngineID

Au sein d'un domaine administratif, un contextEngineID identifie de façon univoque une entité SNMP qui peut réaliser une instance d'un contexte avec un contextName particulier.

3.3.3 contextName

Un contextName est utilisé pour désigner un contexte. Chaque contextName DOIT être unique au sein d'une entité SNMP.

3.3.4 scopedPDU

Une scopedPDU est un bloc de données qui contient un contextEngineID, un contextName, et une PDU.

La PDU est une unité de données de protocole SNMP qui contient des informations nommées dans le contexte qui est identifié sans ambiguïté au sein d'un domaine administratif par la combinaison du contextEngineID et du contextName. Voir, par exemple, dans la [RFC3416] plus d'informations sur la PDU SNMP.

3.4 Autres constructions

3.4.1 maxSizeResponseScopedPDU

La maxSizeResponseScopedPDU est la taille maximum d'une scopedPDU que l'envoyeur d'une PDU accepterait. Noter que la taille d'une scopedPDU n'inclut pas la taille de l'en-tête de message SNMP.

3.4.2 Mémorisation des données de configuration locale

Les sous systèmes, modèles, et applications au sein d'une entité SNMP peuvent avoir besoin de conserver leur propre ensemble d'informations de configuration.

Des portions des informations de configuration peuvent être accessibles comme objets gérés.

La collection de ces ensembles d'informations est appelée "magasin local de données de configuration" (LCD, *Local Configuration Datastore*) d'une entité.

3.4.3 securityLevel

Cette architecture reconnaît trois niveaux de sécurité :
- sans authentification et sans confidentialité (noAuthNoPriv)
- avec authentification mais sans confidentialité (authNoPriv)
- avec authentification et avec confidentialité (authPriv)

Ces trois valeurs sont ordonnées de telle sorte que noAuthNoPriv est inférieur à authNoPriv et authNoPriv est inférieur à authPriv.

Chaque message a un niveau de sécurité associé. Il est EXIGÉ de tous les sous systèmes (traitement de message, sécurité, contrôle d'accès) et applications de fournir une valeur de securityLevel ou de souscrire à la valeur fournie de securityLevel lors du traitement du message et de son contenu.

4. Interfaces de service abstraites

Des interfaces de service abstraites ont été définies pour décrire les interfaces conceptuelles entre les divers sous systèmes au sein d'une entité SNMP. Les interfaces de service abstraites sont destinées à aider à préciser le comportement observable de l'extérieur des entités SNMP, et ne sont en aucune façon destinées à contraindre la structure ou l'organisation des mises en œuvre. Très précisément, elles ne devraient pas être interprétées comme des API ou comme des déclarations d'exigences pour des API.

Ces interfaces de service abstraites sont définies par un ensemble de primitives qui définissent les services fournis et les éléments de données abstraites qui sont à passer lorsque les services sont invoqués. La présente section fait la liste des primitives qui ont été définies pour les divers sous systèmes.

4.1 Primitives de répartiteur

Le répartiteur fournit normalement des services aux applications SNMP via son répartiteur de PDU. Ce paragraphe décrit les primitives fournies par le répartiteur de PDU.

4.1.1 Générer des demandes ou notifications sortantes

Le répartiteur de PDU fournit la primitive suivante pour qu'une application envoie une demande ou notification SNMP à une autre entité SNMP :

```
statusInformation =                     -- sendPduHandle en cas de succès
                                        -- errorIndication en cas d'échec
sendPdu(
    IN      transportDomain             -- domaine de transport à utiliser
    IN      transportAddress            -- adresse de transport à utiliser
    IN      messageProcessingModel      -- normalement, la version SNMP
    IN      securityModel               -- modèle de sécurité à utiliser
    IN      securityName                -- au nom de ce principal
    IN      securityLevel               -- niveau de sécurité demandé
    IN      contextEngineID             -- données de ou vers cette entité
    IN      contextName                 -- données de ou vers ce contexte
    IN      pduVersion                  -- la version de la PDU
    IN      PDU                         -- unités de données de protocole
SNMP
    IN      expectResponse              -- VRAI ou FAUX
        )
```

4.1.2 Traiter la PDU de demande ou notification entrante

Le répartiteur de PDU fournit la primitive suivante pour passer une PDU SNMP entrante à une application :

```
processPdu(                          -- traiter la PDU de demande/notification
   IN  messageProcessingModel        -- normalement, version SNMP
   IN  securityModel                 -- modèle de sécurité utilisé
   IN  securityName                  -- au nom de ce principal
   IN  securityLevel                 -- niveau de sécurité
   IN  contextEngineID               -- données de/vers cette entité SNMP
   IN  contextName                   -- données de/dans ce contexte
   IN  pduVersion                    -- version de la PDU
   IN  PDU                           -- unités de données de protocole SNMP
   IN  maxSizeResponseScopedPDU      -- taille maximum de la PDU de réponse
   IN  stateReference                -- référence aux informations d'état nécessaires pour une
                                     réponse
   )
```

4.1.3 Générer une réponse sortante

Le répartiteur de PDU fournit la primitive suivante pour qu'une application retourne une PDU de réponse SNMP au répartiteur de PDU :

```
result =                             -- SUCCES ou ÉCHEC
returnResponsePdu(
   IN  messageProcessingModel        -- normalement, version SNMP
   IN  securityModel                 -- modèle de sécurité utilisé
   IN  securityName                  -- au nom de ce principal
   IN  securityLevel                 -- le même que sur une demande entrante
   IN  contextEngineID               -- données de/vers cette entité SNMP
   IN  contextName                   -- données de/dans ce contexte
   IN  pduVersion                    -- la version de la PDU
   IN  PDU                           -- unité de données de protocole SNMP
   IN  maxSizeResponseScopedPDU      -- taille maximum que l'envoyeur peut accepter
   IN  stateReference                -- référence aux informations d'état présentées avec la
                                     demande
   IN  statusInformation             -- indication de réussite ou d'erreur
   )                                 -- OID/valeur de compteur d'erreur en cas d'erreur
```

4.1.4 Traiter la PDU de réponse entrante

Le répartiteur de PDU fournit la primitive suivante pour passer une PDU de réponse SNMP entrante à une application :

```
processResponsePdu(                  -- traiter les PDU de réponse
   IN  messageProcessingModel        -- normalement, version SNMP
   IN  securityModel                 -- modèle de sécurité utilisé
   IN  securityName                  -- au nom de ce principal
   IN  securityLevel                 -- niveau de sécurité
   IN  contextEngineID               -- données de/vers cette entité SNMP
   IN  contextName                   -- données de/dans ce contexte
   IN  pduVersion                    -- la version de la PDU
   IN  PDU                           -- unité de données de protocole SNMP
   IN  statusInformation             -- indication de réussite ou d'erreur
   IN  sendPduHandle                 -- bride provenant de la sendPdu
   )
```

4.1.5 Enregistrement de la responsabilité du traitement de PDU SNMP

Les applications peuvent enregistrer/désenregistrer la responsabilité d'un contextEngineID spécifique, pour des pduTypes spécifiques, auprès du répartiteur de PDU selon les primitives suivantes. La liste des pduTypes particuliers qu'une application peut enregistrer est déterminée par le ou les modèles de traitement de message pris en charge par l'entité SNMP qui contient le répartiteur de PDU.

```
statusInformation =                  -- Indication de succès ou d'erreur
registerContextEngineID(
```

```
    IN      contextEngineID         -- prend la responsabilité pour celui-ci
    IN      pduType                 -- le ou les pduTypes à enregistrer
    )

unregisterContextEngineID(
   IN   contextEngineID             -- abandonne la responsabilité pour celui-ci
   IN   pduType                     -- le ou les pduType à désenregistrer
   )
```

Noter que les réalisations d'interfaces de service abstraites de registerContextEngineID et unregisterContextEngineID peuvent fournir aux applications des façons, spécifiques de la mise en œuvre, de la responsabilité d'enregistrer/désenregistrer pour toutes les valeurs possibles des paramètres contextEngineID ou pduType.

4.2 Primitives du sous système de traitement de message

Le répartiteur interagit avec un modèle de traitement de message pour traiter une version spécifique de message SNMP. Ce paragraphe décrit les primitives fournies par le sous système de traitement de message.

4.2.1 Préparer une demande SNMP sortante ou un message de notification

Le sous système de traitement de message fournit cette primitive de service pour préparer une demande SNMP sortante ou un message de notification :

```
statusInformation =                 -- indication de réussite ou d'échec
prepareOutgoingMessage(
   IN    transportDomain            -- domaine de transport à utiliser
   IN    transportAddress           -- adresse de transport à utiliser
   IN    messageProcessingModel     -- normalement, la version SNMP
   IN    securityModel              -- modèle de sécurité à utiliser
   IN    securityName               -- au nom de ce principal
   IN    securityLevel              -- niveau de sécurité demandé
   IN    contextEngineID            -- donnée de/vers cette entité
   IN    contextName                -- données de/vers ce contexte
   IN    pduVersion                 -- version de la PDU
   IN    PDU                        -- unité de données de protocole SNMP
   IN    expectResponse             -- VRAI ou FAUX
   IN    sendPduHandle              -- bride pour faire correspondre les réponses entrantes
   OUT   destTransportDomain        -- domaine de transport de destination
   OUT   destTransportAddress       -- adresse  de transport de destination
   OUT   outgoingMessage            -- le message à envoyer
   OUT   outgoingMessageLength      -- sa longueur
   )
```

4.2.2 Préparer un message de réponse SNMP sortant

Le sous système de traitement de message fournit cette primitive de service pour préparer un message de réponse SNMP sortant :

```
result =                                -- SUCCÈS ou ÉCHEC
prepareResponseMessage(
    IN      messageProcessingModel      -- normalement, la version SNMP
    IN      securityModel               -- le même que sur une demande entrante
    IN      securityName                -- le même que sur une demande entrante
    IN      securityLevel               -- le même que sur une demande entrante
    IN      contextEngineID             -- données de/vers cette entité SNMP
    IN      contextName                 -- donnés de/dans ce contexte
    IN      pduVersion                  -- version de la PDU
    IN      PDU                         -- unité de données de protocole SNMP
    IN      maxSizeResponseScopedPDU    -- taille maximum acceptable
    IN      stateReference              -- référence aux informations d'état présentées dans la
demande
    IN      statusInformation           -- Indication de succès ou erreur
```

```
                                        -- OID/valeur de compteur d'erreur en cas d'erreur
    OUT   destTransportDomain           -- domaine de transport de destination
    OUT   destTransportAddress          -- adresse de transport de destination
    OUT   outgoingMessage               -- message à envoyer
    OUT   outgoingMessageLength         -- sa longueur
    )
```

4.2.3 Préparer des éléments de données provenant d'un message SNMP entrant

Le sous système de traitement de message fournit cette primitive de service pour préparer les éléments de données abstraits provenant d'un message SNMP entrant :

```
result =                                -- SUCCÈS ou indication d'erreur
   prepareDataElements(
   IN    transportDomain                -- domaine de transport d'origine
   IN    transportAddress               -- adresse de transport d'origine
   IN    wholeMsg                       -- comme reçu du réseau
   IN    wholeMsgLength                 -- comme reçu du réseau
   OUT   messageProcessingModel         -- normalement, la version SNMP
   OUT   securityModel                  -- modèle de sécurité à utiliser
   OUT   securityName                   -- au nom de ce principal
   OUT   securityLevel                  -- niveau de sécurité demandé
   OUT   contextEngineID                -- données de/vers cette entité
   OUT   contextName                    -- données de/dans ce contexte
   OUT   pduVersion                     -- version de la PDU
   OUT   PDU                            -- unité de données de protocole SNMP
   OUT   pduType                        -- type de PDU SNMP
   OUT   sendPduHandle                  -- bride pour faire correspondre les demandes
   OUT   maxSizeResponseScopedPDU       -- taille maximum que l'envoyeur peut accepter
   OUT   statusInformation              -- indication de succès ou d'erreur
                                        -- OID/valeur de compteur d'erreur en cas d'erreur
   OUT   stateReference                 -- référence aux informations d'état à utiliser pour une réponse possible
         )
```

4.3 Primitives du sous système de contrôle d'accès

Les applications sont les clients normaux du ou des services du sous système de contrôle d'accès.

La primitive suivante est fournie par le sous système de contrôle d'accès pour vérifier si l'accès est permis :

```
statusInformation =                     -- indication de succès ou d'erreur
isAccessAllowed(
   IN    securityModel                  -- modèle de sécurité utilisé
   IN    securityName                   -- principal qui veut l'accès
   IN    securityLevel                  -- niveau de sécurité
   IN    viewType                       -- vue en lecture, écriture, ou notification
   IN    contextName                    -- contexte contenant un variableName
   IN    variableName                   -- OID pour l'objet géré
         )
```

4.4 Primitives du sous système de sécurité

Le sous système de traitement de message est le client normal des services du sous système de sécurité.

4.4.1 Générer un message de demande ou de notification

Le sous système de sécurité fournit la primitive suivante pour générer un message de demande ou de notification :

```
statusInformation =
   generateRequestMsg(
   IN    messageProcessingModel        -- normalement, la version SNMP
```

```
    IN      globalData              -- en-tête de message, données d'administration
    IN      maxMessageSize          -- de l'entité SNMP d'envoi
    IN      securityModel           -- pour le message sortant
    IN      securityEngineID        -- entité SNMP d'autorité
    IN      securityName            -- au nom de ce principal
    IN      securityLevel           -- niveau de sécurité demandé
    IN      scopedPDU               -- charge utile du message (texte source)
    OUT     securityParameters      -- remplie par le module de sécurité
    OUT     wholeMsg                -- message généré complet
    OUT     wholeMsgLength          -- longueur du message généré
            )
```

4.4.2 Traitement de message entrant

Le sous système de sécurité fournit la primitive suivante pour traiter un message entrant :

```
statusInformation =                 -- indication d'erreur ou de succès
                                    -- OID/valeur de compteur d'erreur si erreur
    processIncomingMsg(
    IN      messageProcessingModel  -- normalement, la version SNMP
    IN      maxMessageSize          -- de l'entité SNMP envoyeuse
    IN      securityParameters      -- pour le message reçu
    IN      securityModel           -- pour le message reçu
    IN      securityLevel           -- niveau de sécurité
    IN      wholeMsg                -- comme reçu sur le réseau
    IN      wholeMsgLength          -- longueur telle que reçue sur le réseau
    OUT     securityEngineID        -- entité SNMP d'autorité
    OUT     securityName            -- identification du principal
    OUT     scopedPDU               -- charge utile du message (texte source)
    OUT     maxSizeResponseScopedPDU -- taille maximum que l'envoyeur peut traiter
    OUT     securityStateReference  -- référence aux informations d'état de sécurité, nécessaires pour la réponse
            )
```

4.4.3 Générer un message de réponse

Le sous système de sécurité fournit la primitive suivante pour générer un message de réponse :

```
statusInformation =
    generateResponseMsg(
    IN      messageProcessingModel  -- normalement, la version SNMP
    IN      globalData              -- en-tête de message, données administratives
    IN      maxMessageSize          -- de l'entité SNMP d'envoi
    IN      securityModel           -- pour le message sortant
    IN      securityEngineID        -- entité SNMP d'autorité
    IN      securityName            -- au nom de ce principal
    IN      securityLevel           -- pour le message sortant
    IN      scopedPDU               -- charge utile de message (texte source)
    IN      securityStateReference  -- référence aux informations d'état de sécurité dans la demande
                                    d'origine
    OUT     securityParameters      -- rempli par le module de sécurité
    OUT     wholeMsg                -- message généré complet
    OUT     wholeMsgLength          -- longueur du message généré
            )
```

4.5 Primitives communes

Ces primitives sont fournies par plusieurs sous systèmes.

4.5.1 Libérer les informations de référence d'état

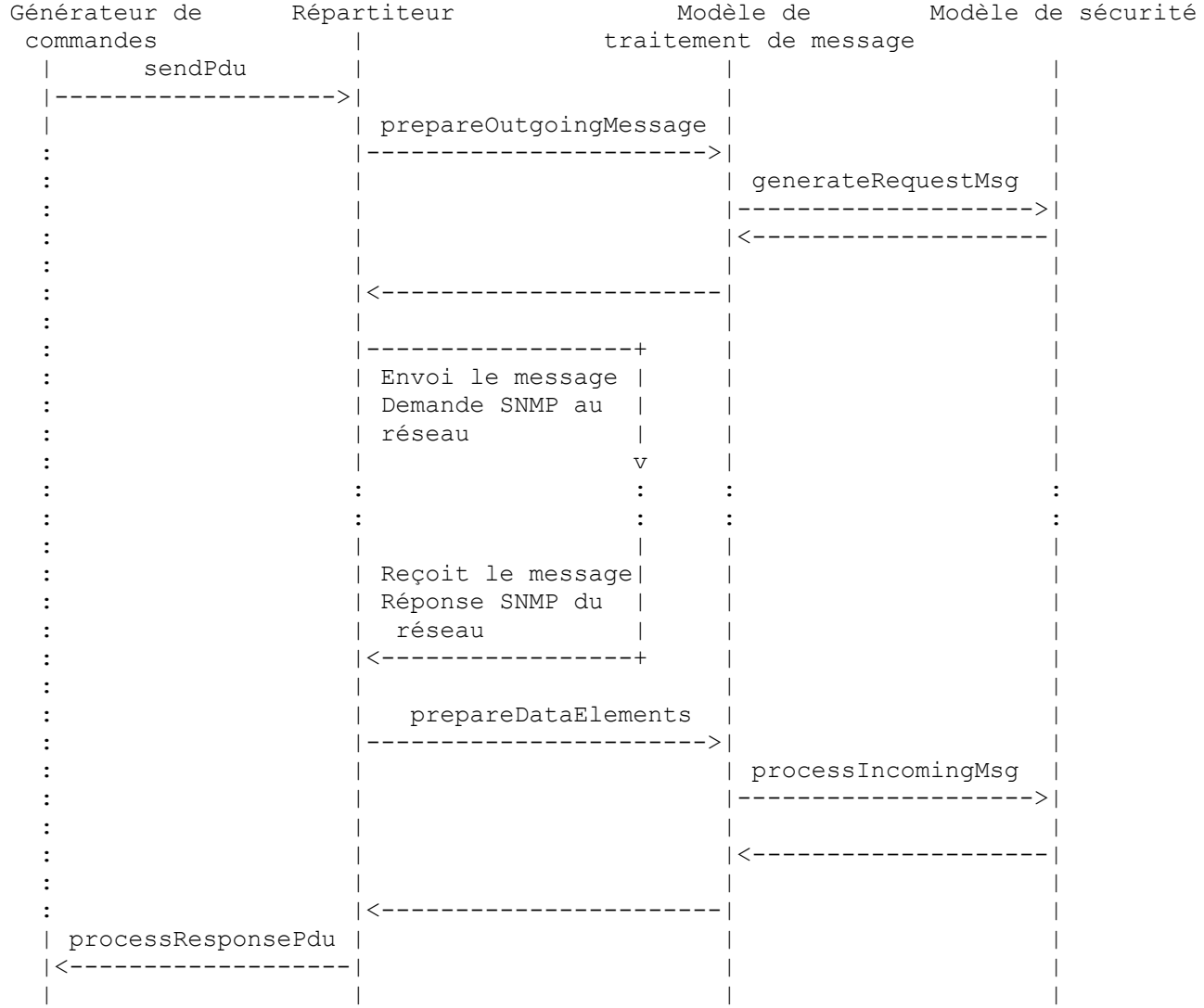
Tous les sous systèmes qui passent des informations de référence d'état *(stateReference)* fournissent aussi une primitive pour libérer la mémoire qui détient les informations d'état référencées :

```
stateRelease(
   IN   stateReference          -- bride vers les référence à libérer
        )
```

4.6 Diagrammes de scénarios

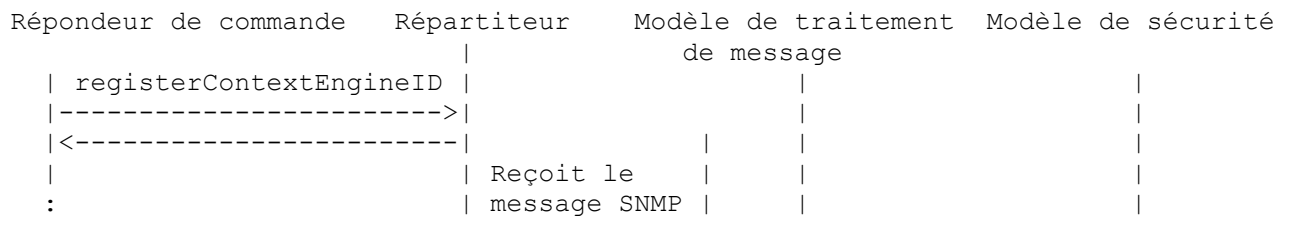
4.6.1 Générateur de commande ou générateur de notification

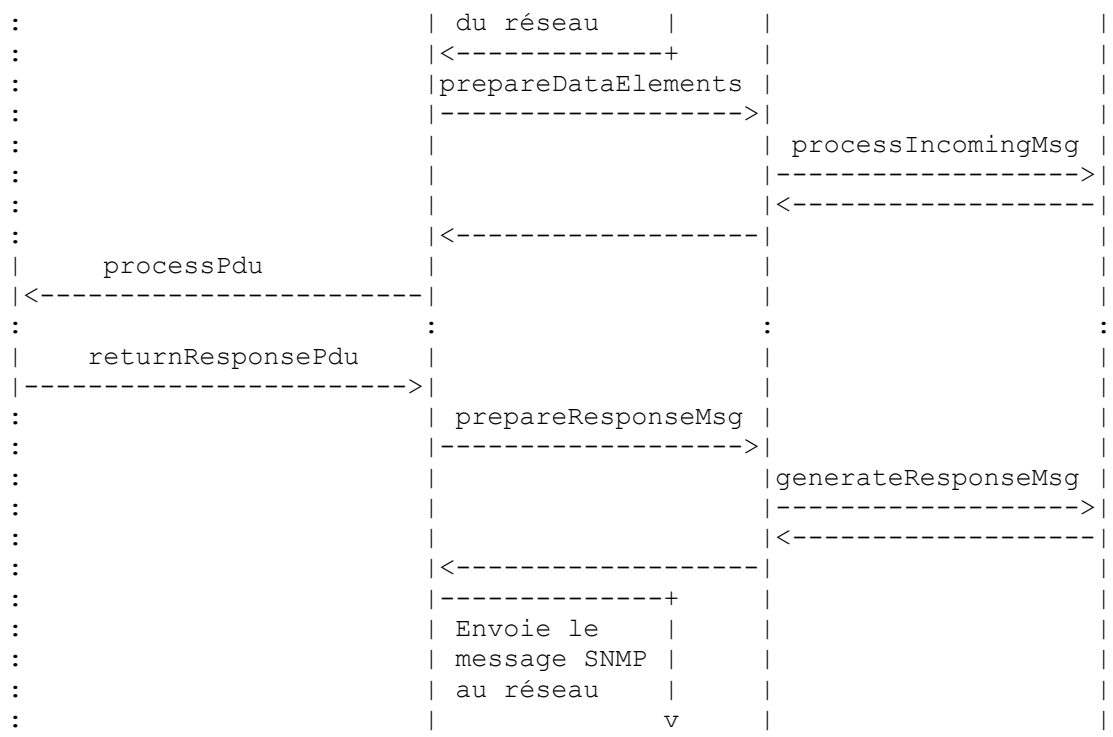
Ce diagramme montre comment une application de générateur de commandes ou générateur de notifications demande qu'une PDU soit envoyée, et comment la réponse est retournée (en asynchrone) à cette application.



4.6.2 Diagramme du scénario pour une application de répondeur de commande

Ce diagramme montre comment une application de répondeur de commandes ou de receveur de notifications s'enregistre pour traiter un type de PDU *(pduType)*, comment une PDU est expédiée à l'application après la réception d'un message SNMP, et comment la réponse est renvoyée (en asynchrone) au réseau.





5. Définitions des objets gérés pour les cadres de travail SNMP

DEFINITIONS SNMP-FRAMEWORK-MIB ::= DÉBUT

IMPORTE
IDENTITÉ-DE-MODULE, TYPE-D'OBJET, IDENTITÉ-D'OBJET,
snmpModules DE SNMPv2-SMI
CONVENTION-TEXTUELLE DE SNMPv2-TCCONFORMITÉ-DE-MODULE, GROUPE-D'OBJET DE SNMPv2-CONF;

IDENTITÉ-DE-MODULE snmpFrameworkMIB
DERNIERE-MISE-A-JOUR "200210140000Z"
ORGANISATION "Groupe de travail SNMPv3"
CONTACT-INFO "WG-EMail: snmpv3@lists.tislabs.com
Abonnement : snmpv3-request@lists.tislabs.com
Co-présidence : Russ Mundy
Network Associates Laboratories
adresse postale : 15204 Omega Drive, Suite 300
Rockville, MD 20850-4601
USA
mél : mundy@tislabs.com
téléphone : +1 301-947-7107

Co-président/co-éditeur : David Harrington
Enterasys Networks
adresse postale : 35 Industrial Way
P. O. Box 5005
Rochester, New Hampshire 03866-5005
USA
mél : dbh@enterasys.com
téléphone : +1 603-337-2614

Co-éditeur : Randy Presuhn
BMC Software, Inc.
adresse postale : 2141 North First Street
San Jose, California 95131

USA
mél : randy_presuhn@bmc.com
téléphone : +1 408-546-1006

Co-éditeur : Bert Wijnen
Lucent Technologies
adresse postale : Schagen 33
3461 GL Linschoten
Netherlands
mél : bwijnen@lucent.com
téléphone : +31 348-680-485 "

DESCRIPTION : "MIB d'architecture de gestion de SNMP. Cette version de ce module de MIB fait partie de la RFC 3411 ; voir dans la RFC elle –même les notices légales complètes".

REVISION "200210140000Z" -- 14 octobre 2002
DESCRIPTION "Changements dans cette révision :
- Mise à jour de diverses informations administratives.
- Correction d'erreurs typographiques.
- Correction d'erreurs typographiques de la description de SnmpEngineID donnant un recouvrement de gamme pour 127.
- Changé '255a' en '255t' dans la définition de SnmpAdminString pour l'aligner avec la SMI réelle.
- Reformulation de 'réservé' pour la valeur zéro dans la DESCRIPTION de SnmpSecurityModel.
- L'algorithme d'allocation des modèles de sécurité devrait donner 256 par bloc d'entreprise, plutôt que 255.
- L'exemple d'identifiant de moteur de 'abcd' n'est pas légal. Remplacé par '800002b804616263'H fondé sur l'exemple d'entreprise 696, chaîne 'abc'.
- Ajout de la précision que l'identifiant de moteur devrait persister à travers une réinitialisation.
Cette révision est publiée comme RFC 3411."
REVISION : "199901190000Z" -- 19 janvier 1999
DESCRIPTION :"mise à jour de l'adresse des éditeurs, correction de fautes de frappe. Publié comme RFC 2571. "
REVISION : "199711200000Z" -- 20 novembre 1997
DESCRIPTION "Version initiale, publiée comme RFC 2271."
::= { snmpModules 10 }

-- Conventions textuelles utilisées dans l'architecture de gestion SNMP ***

SnmpEngineID ::= CONVENTION TEXTUELLE
STATUT : actuel
DESCRIPTION "Identifiant administrativement unique d'un moteur SNMP. Les objets de ce type sont pour l'identification, pas pour l'adressage, même s'il est possible qu'une adresse puisse être utilisée dans la génération d'une valeur spécifique.
La valeur de cet objet peut n'être pas toute à zéro ou toute à 'ff'H ou la chaîne vide (longueur zéro).
La valeur initiale pour cet objet peut être configurée via une entrée de console d'opérateur ou via une fonction algorithmique. Dans ce dernier cas, l'exemple d'algorithme suivant est recommandé.
Dans les cas où il y a plusieurs moteurs sur le même système, l'utilisation de cet algorithme N'EST PAS appropriée, car elle résulterait en ce que tous les moteurs finissent avec la même valeur d'ID.

1) Le premier bit est utilisé pour indiquer comment le reste des données est composé.
0 – comme défini par l'entreprise qui utilise d'anciennes méthodes existant avant SNMPv3 ; voir le 2 ci-dessous.
1 - comme défini par la présente architecture ; voir le 3 ci-dessous.
Noter que ceci permet que coexistent les utilisations actuelles de engineID (aussi appelé AgentID [RFC1910]) avec tous nouveaux usages.
2) snmpEngineID a une longueur de 12 octets.
Les quatre premiers octets sont réglés à l'équivalent binaire du numéro d'entreprise privée de gestion SNMP de l'agent comme alloué par l'Autorité d'allocation des numéros de l'Internet (IANA). Par exemple, si Acme Networks a reçu { enterprises 696 }, les quatre premiers octets alloués seront '000002b8'H.
Les huit bits restants sont déterminés via une ou plusieurs méthodes spécifiques de l'entreprise. Des telles méthodes doivent être conçues de façon à maximiser la possibilité que la valeur de cet objet soit unique dans le domaine administratif de l'agent. Par exemple, ce peut être l'adresse IP de l'entité SNMP, ou d'adresse MAC d'une ces interfaces, chaque adresse étant bourrée adéquatement d'octets aléatoires. Si plusieurs méthodes sont définies, il est alors recommandé que le premier octet indique la méthode utilisée et que les octets restants soient fonction de la méthode.
3) La longueur de la chaîne d'octets varie.
Les quatre premiers octets sont réglés à l'équivalent binaire du numéro d'entreprise privé de gestion SNMP de l'agent comme alloué par l'IANA. Par exemple, si Acme Networks a reçu { enterprises 696 }, les quatre premiers octets alloués seront '000002b8'H.
Le premier bit est réglé à 1. Par exemple, la valeur ci-dessus pour Acme Networks change maintenant en '800002b8'H.

Le cinquième octet indique comment le reste (octets 6 et suivants) sont formatés. Les valeurs pour le cinquième octet sont :

0 - réservé, non utilisé
1 - adresse IPv4 (4 octets) plus basse adresse IP non spéciale
2 - adresse IPv6 (16 octets) plus basse adresse IP non spéciale
3 - adresse MAC (6 octets) plus basse adresse MAC IEEE, ordre canonique
4 - Texte, longueur restante maximum allouée administrativement de 27
5 - Octets, longueur restante maximum allouée administrativement de 27
6-127 - réservé, non utilisé
128-255 - comme défini par la longueur restante maximum de l'entreprise de 27 "

SYNTAXE : CHAINE D'OCTETS (TAILLE(5 à 32))

SnmpSecurityModel ::= CONVENTION TEXTUELLE
STATUT : actuel
DESCRIPTION : "Identifiant qui identifie de façon univoque un modèle de sécurité du sous système de sécurité au sein de l'architecture de gestion SNMP.

Les valeurs pour securityModel sont allouées comme suit :

- La valeur zéro n'identifie aucun modèle de sécurité particulier.
- Les valeurs entre 1 et 255, inclus, sont réservées pour les modèles de sécurité en cours de normalisation et sont gérées par l'IANA.
- Les valeurs supérieures à 255 sont allouées aux modèles de sécurité spécifiques des entreprises. Une valeur de securityModel spécifique d'une entreprise est définie comme étant :
  enterpriseID * 256 + modèle de sécurité au sein de l'entreprise

Par exemple, le quatrième modèle de sécurité défini par l'entreprise dont l'enterpriseID est 1 serait 259.

Ce schéma d'allocation des valeurs de securityModel permet un maximum de 255 modèles de sécurité fondés sur des normes, et un maximum de 256 modèles de sécurité par entreprise.

On estime que l'allocation de nouvelles valeurs de securityModel sera rare en pratique parce que plus le nombre de modèles de sécurité est grand, plus les chances que souffre l'interopérabilité croissent. Par conséquent, on estime qu'une telle gamme sera suffisante. Dans le cas improbable où le comité de normalisation trouverait que ce nombre est insuffisant à l'avenir, un numéro d'entreprise peut être alloué pour obtenir 256 valeurs possibles supplémentaires.

Noter que le bit de poids fort doit être zéro ; donc, il y a 23 bits alloués pour les diverses organisations pour concevoir et définir des securityModel non standard. Cela limite la capacité de définir de nouvelles mises en œuvre propriétaires de modèles de sécurité aux 8 388 608 premières entreprises.

On peut noter que, sous sa forme codée, la valeur de securityModel va normalement requérir seulement un octet, car, en pratique, les bits les plus à gauche seront à zéro pour la plupart des messages et que l'extension de signe est supprimée par les règles de codage.

Au moment de cette rédaction, plusieurs valeurs de securityModel sont définies pour l'usage de SNMP ou réservées pour être utilisées avec les objets de MIB support. Ce sont :

0 réservé pour 'any'
1 réservé pour SNMPv1
2 réservé pour SNMPv2c
3 modèle de sécurité fondé sur l'utilisateur (USM) "

SYNTAXE : ENTIER(0 à 2 147 483 647)

SnmpMessageProcessingModel ::= CONVENTION TEXTUELLE
STATUT : actuel
DESCRIPTION : "Identifiant qui identifie de façon univoque un modèle de traitement de message du sous système de traitement de message au sein de cette architecture de gestion SNMP.

Les valeurs pour messageProcessingModel sont allouées comme suit :

- Les valeurs entre 0 et 255, inclus, sont réservées pour les modèles de traitement de message en cours de normalisation et sont gérées par l'IANA.
- Les valeurs supérieures à 255 sont allouées aux modèles de traitement de message spécifique des entreprises. Une valeur de messageProcessingModel d'entreprise est définie comme étant : enterpriseID * 256 + messageProcessingModel au sein de l'entreprise

Par exemple, le quatrième modèle de traitement de message défini par l'entreprise dont l'enterpriseID est 1 sera 259.

Ce schéma pour allouer les valeurs de messageProcessingModel permet un maximum de 255 modèles de traitement de message fondés sur les normes, et un maximum de 256 modèles de traitement de message par entreprise.

On estime que l'allocation de nouvelles valeurs de messageProcessingModel sera rare en pratique parce que plus le nombre de modèles de traitement de message utilisés simultanément est grand, plus grandes sont les chances que cela soit au détriment de l'interopérabilité. On estime qu'une telle gamme sera suffisante. Dans le cas improbable où le comité de normalisation trouverait que ce nombre est insuffisant à l'avenir, un numéro d'entreprise peut être alloué pour obtenir 256 valeurs possibles supplémentaires.

Noter que le bit de poids fort doit être zéro ; donc, il y a 23 bits alloués pour diverses organisations pour concevoir et définir des messageProcessingModel non standard. Cela limite la capacité à définir la mise en œuvre de nouveaux modèles de traitement de message supplémentaires aux 8 388 608 premières entreprises.

On pourra noter que, sous sa forme codée, la valeur de messageProcessingModel va normalement exiger seulement un octet, en pratique, les bits de gauche seront à zéro pour la plupart des messages et l'extension de signe est supprimée par les règles de codage.

Au moment de cette rédaction, plusieurs valeurs de messageProcessingModel sont définies pour l'usage de SNMP.

Ce sont :

0 réservé pour SNMPv1
1 réservé pour SNMPv2c
2 réservé pour SNMPv2u et SNMPv2*
3 réservé pour SNMPv3 "

SYNTAXE : ENTIER(0 à 2 147 483 647)

SnmpSecurityLevel ::= CONVENTION TEXTUELLE
STATUT : actuel
DESCRIPTION : "Niveau de sécurité auquel les messages SNMP peuvent être envoyés ou auquel les opérations sont traitées ; en particulier, un des suivants :

noAuthNoPriv - sans authentification et sans confidentialité,
authNoPriv - avec authentification mais sans confidentialité,
authPriv - avec authentification et avec confidentialité.

Ces trois valeurs sont ordonnées de façon que noAuthNoPriv soit moins que authNoPriv et authNoPriv soit moins que authPriv".
SYNTAXE : ENTIER { noAuthNoPriv(1), authNoPriv(2), authPriv(3) }

SnmpAdminString ::= CONVENTION TEXTUELLE
CONSEIL D'AFFICHAGE "255t"
STATUT : actuel
DESCRIPTION : "Chaîne d'octets contenant des informations administratives, de préférence sous une forme lisible par l'homme.

Pour faciliter l'internationalisation, ces informations sont représentées en utilisant la norme internationale de jeu de caractères ISO/CEI 10646-1, codées comme chaîne d'octets en utilisant le format de transformation UTF-8 décrit dans la [RFC2279].

Comme des codets supplémentaires sont ajoutés de temps en temps par les amendements à la norme 10646, les mises en œuvre doivent être prêtes à rencontrer tout codet de 0x00000000 à 0x7fffffff. Les séquences d'octets qui ne correspondent pas au codage valide UTF-8 d'un codet ou sont en-dehors de cette gamme sont interdites.

L'utilisation de codes de contrôle devrait être évitée.

Lorsque il est nécessaire de représenter une nouvelle ligne, la séquence de code de contrôle CR LF devrait être utilisée.

L'utilisation d'espaces en tête ou en queue devrait être évitée.

Pour les codets non directement pris en charge par le matériel ou logiciel d'interface d'utilisateur, un moyen de remplacement d'entrée et d'affichage, tel que l'hexadécimal, peut être fourni.

Pour les informations codées en US-ASCII à 7 bits, le codage UTF-8 est identique à celui de l'US-ASCII.

L'UTF-8 peut exiger plusieurs octets pour représenter un seul caractère/codet ; donc la longueur de cet objet en octets peut être différente du nombre de caractères codés. De même, les contraintes de taille se réfèrent au nombre d'octets codés, non au nombre de caractères représentés par un codage.

Noter que lorsque cette convention textuelle est utilisée pour un objet qui sert ou est envisage de servir comme indice, une restriction de TAILLE DOIT être spécifiée afin que le nombre de sous identifiants pour toute instance d'objet n'excède pas la limite de 128, comme défini par la [RFC3416].

Noter que la taille d'un objet SnmpAdminString est mesurée en octets, et non en caractères".
SYNTAXE : CHAINE D'OCTETS (TAILLE (0 à 255))

-- Allocations administratives **********

IDENTIFIANT D'OBJET snmpFrameworkAdmin ::= { snmpFrameworkMIB 1 }
IDENTIFIANT D'OBJET snmpFrameworkMIBObjects ::= { snmpFrameworkMIB 2 }
IDENTIFIANT D'OBJET snmpFrameworkMIBConformance ::= { snmpFrameworkMIB 3 }

-- Groupe snmpEngine **********

IDENTIFIANT D'OBJET snmpEngine ::= { snmpFrameworkMIBObjects 1 }

TYPE-D'OBJET snmpEngineID

SYNTAXE : SnmpEngineID
MAX-ACCES : lecture seule
STATUT : actuel
DESCRIPTION : "Identifiant administrativement unique d'un moteur SNMP. Cette information DEVRAIT être mémorisée dans une mémoire non volatile afin qu'elle reste constante à travers les réinitialisations du moteur SNMP".
::= { snmpEngine 1 }

TYPE-D'OBJET snmpEngineBoots
SYNTAXE : ENTIER (1 à 2 147 483 647)
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre de fois que le moteur SNMP s'est réinitialisé depuis la dernière configuration de snmpEngineID".
::= { snmpEngine 2 }

TYPE-D'OBJET snmpEngineTime
SYNTAXE : ENTIER (0 à 2 147 483 647)
UNITÉS : "secondes"
MAX-ACCES : lecture-seule
STATUT : actuel
DESCRIPTION : "Nombre de secondes depuis le dernier changement de la valeur de l'objet snmpEngineBoots. Lorsque l'incrémentation de la valeur de cet objet lui ferait dépasser son maximum, snmpEngineBoots est incrémenté comme si une réinitialisation s'était produite, et la valeur de cet objet revient par conséquent à zéro".
::= { snmpEngine 3 }

TYPE-D'OBJET snmpEngineMaxMessageSize
SYNTAXE : ENTIER (484 à 2 147 483 647)
MAX-ACCES : lecture-seule
STATUT : actuel
DESCRIPTION : "Longueur maximum en octets d'un message SNMP que ce moteur SNMP peut envoyer ou recevoir et traiter, déterminée comme le minimum de la valeur de taille maximum de message acceptée parmi tous les transports disponibles et pris en charge par le moteur".
::= { snmpEngine 4 }

-- Points d'enregistrement pour les protocoles d'authentification et de confidentialité **

IDENTITÉ-D'OBJET snmpAuthProtocols
STATUT : actuel
DESCRIPTION : Point d'enregistrement pour les protocoles d'authentification en cours de normalisation utilisés dans les cadres de gestion SNMP".
::= { snmpFrameworkAdmin 1 }

IDENTITÉ-D'OBJET snmpPrivProtocols
STATUT : actuel
DESCRIPTION : "Point d'enregistrement pour les protocoles de confidentialité en cours de normalisation utilisés dans les cadres de gestion SNMP".
::= { snmpFrameworkAdmin 2 }

-- Informations de conformité **********

IDENTIFIANT D'OBJET snmpFrameworkMIBCompliances ::= {snmpFrameworkMIBConformance 1}
IDENTIFIANT D'OBJET snmpFrameworkMIBGroups ::= {snmpFrameworkMIBConformance 2}

-- Déclarations de conformité

CONFORMITÉ-DE-MODULE snmpFrameworkMIBCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les moteurs SNMP qui mettent en œuvre la MIB de cadre de gestion SNMP".
MODULE -- ce module

```
GROUPES-OBLIGATOIRES        { snmpEngineGroup }
        ::= { snmpFrameworkMIBCompliances 1 }

-- unités de conformité

GROUPE-D'OBJETS snmpEngineGroup
OBJETS { snmpEngineID, snmpEngineBoots, snmpEngineTime, snmpEngineMaxMessageSize }
STATUT : actuel
DESCRIPTION : "Collection d'objets pour identifier et déterminer les valeurs de configuration et d'opportunité réelles d'un
              moteur SNMP".
        ::= { snmpFrameworkMIBGroups 1 }

FIN
```

6. Considérations relatives à l'IANA

Le présent document définit trois espaces de numéros administrés par l'IANA, un pour les modèles de sécurité, un autre pour les modèles de traitement de message, et un troisième pour les formats SnmpEngineID.

6.1 Modèles de sécurité

Les valeurs de CONVENTION TEXTUELLE SnmpSecurityModel gérées par l'IANA sont dans la gamme de 0 à 255 inclus, et sont réservées pour les modèles de sécurité en cours de normalisation. Si cette gamme devrait à l'avenir se révéler insuffisante, un numéro d'entreprise pourrait être alloué pour obtenir 256 valeurs possibles supplémentaires.

Au moment de cette rédaction, plusieurs valeurs de securityModel sont définies pour l'usage de SNMP ou réservées pour être utilisées avec les objets de MIB support. Ce sont :

0 réservé pour 'tous'
1 réservé pour SNMPv1
2 réservé pour SNMPv2c
3 modèle de sécurité fondé sur l'utilisateur (USM)

6.2 Modèles de traitement des messages

Les valeurs de CONVENTION TEXTUELLE SnmpMessageProcessingModel gérées par l'IANA sont dans la gamme 0 à 255, inclus. Chaque valeur identifie de façon univoque un modèle de traitement de message en cours de normalisation du sous système de traitement de message au sein de l'architecture de gestion SNMP.

Si cette gamme devait se révéler insuffisante à l'avenir, un numéro d'entreprise pourrait être obtenu pour que le comité de normalisation obtienne 256 valeurs supplémentaires possibles.

Au moment de cette rédaction, plusieurs valeurs de messageProcessingModel sont définies pour l'usage de SNMP. Ce sont :

0 réservé pour SNMPv1
1 réservé pour SNMPv2c
2 réservé pour SNMPv2u et SNMPv2*
3 réservé pour SNMPv3

6.3 Formats de SnmpEngineID

Le cinquième octet de la CONVENTION TEXTUELLE SnmpEngineID contient un identifiant de format. Les valeurs gérées par l'IANA sont dans la gamme 6 à 127, inclus. Chaque valeur identifie de façon univoque un format SnmpEngineID en cours de normalisation.

7. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document

ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous droits de reproduction, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

8. Remerciements

Le présent document est le résultat des efforts du groupe de travail SNMPv3. Des remerciements particuliers sont adressés par ordre alphabétique aux membres suivants du GT SNMPv3 :

Harald Tveit Alvestrand (Maxware)
Dave Battle (SNMP Research, Inc.)
Alan Beard (Disney Worldwide Services)
Paul Berrevoets (SWI Systemware/Halcyon Inc.)
Martin Bjorklund (Ericsson)
Uri Blumenthal (IBM T.J. Watson Research Center)
Jeff Case (SNMP Research, Inc.)
John Curran (BBN)
Mike Daniele (Compaq Computer Corporation)
T. Max Devlin (Eltrax Systems)
John Flick (Hewlett Packard)
Rob Frye (MCI)
Wes Hardaker (U.C.Davis, Information Technology - D.C.A.S.)
David Harrington (Cabletron Systems Inc.)
Lauren Heintz (BMC Software, Inc.)
N.C. Hien (IBM T.J. Watson Research Center)
Michael Kirkham (InterWorking Labs, Inc.)
Dave Levi (SNMP Research, Inc.)
Louis A Mamakos (UUNET Technologies Inc.)
Joe Marzot (Nortel Networks)
Paul Meyer (Secure Computing Corporation)
Keith McCloghrie (Cisco Systems)
Bob Moore (IBM)
Russ Mundy (TIS Labs at Network Associates)
Bob Natale (ACE*COMM Corporation)
Mike O'Dell (UUNET Technologies Inc.)
Dave Perkins (DeskTalk)
Peter Polkinghorne (Brunel University)
Randy Presuhn (BMC Software, Inc.)
David Reeder (TIS Labs at Network Associates)
David Reid (SNMP Research, Inc.)
Aleksey Romanov (Quality Quorum)
Shawn Routhier (Epilogue)
Juergen Schoenwaelder (TU Braunschweig)
Bob Stewart (Cisco Systems)
Mike Thatcher (Independent Consultant)
Bert Wijnen (IBM T.J. Watson Research Center)

Le document se fonde sur les recommandations de l'équipe conseil Évolution du cadre administratif et de sécurité pour SNMP de l'IETF. Les membres de cette équipe conseil étaient :
David Harrington (Cabletron Systems Inc.)
Jeff Johnson (Cisco Systems)
David Levi (SNMP Research Inc.)

John Linn (Openvision)
Russ Mundy (Trusted Information Systems) chair
Shawn Routhier (Epilogue)
Glenn Waters (Nortel)
Bert Wijnen (IBM T. J. Watson Research Center)

Comme recommandé par l'équipe conseil et la charte du groupe de travail SNMPv3, la conception a incorporé autant que faire se peut des précédentes RFC et projets. Il en résulte que des remerciements particuliers sont dus aux auteurs des projets précédents connus sous les noms de SNMPv2u et de SNMPv2*:

Jeff Case (SNMP Research, Inc.)
David Harrington (Cabletron Systems Inc.)
David Levi (SNMP Research, Inc.)
Keith McCloghrie (Cisco Systems)
Brian O'Keefe (Hewlett Packard)
Marshall T. Rose (Dover Beach Consulting)
Jon Saperia (BGS Systems Inc.)
Steve Waldbusser (International Network Services)
Glenn W. Waters (Bell-Northern Research Ltd.)

9. Considérations pour la sécurité

Le présent document décrit comment une mise en œuvre peut inclure un modèle de sécurité pour protéger les messages de gestion et un modèle de contrôle d'accès pour contrôler l'accès aux informations de gestion.

Le niveau de sécurité fourni est déterminé par la ou les mises en œuvre spécifiques du modèle de sécurité et la ou les mises en œuvre spécifiques de modèle de contrôle d'accès utilisées.

Les applications ont accès à des données qui ne sont pas sécurisées. Les applications DEVRAIENT prendre des mesures raisonnables pour protéger les données contre leur divulgation.

Il est de la responsabilité de l'acheteur d'une mise en œuvre de s'assurer que :
1) une mise en œuvre est conforme aux règles définies par cette architecture,
2) les modèles de sécurité et de contrôle d'accès utilisés satisfont aux besoins de sécurité et de contrôle d'accès de l'organisation,
3) les mises en œuvre des modèles et des applications se conforment aux spécifications du modèle et des applications,
4) que la mise en œuvre protège les secrets de configuration d'une divulgation accidentelle.

Le présent document contient aussi un module de définition de MIB. Aucun des objets définis ici n'est écrivable, et les informations qui y sont représentées ne sont pas réputées être particulièrement sensibles. Cependant, si elles sont réputées sensibles dans un environnement particulier, leur accès devrait être restreint grâce à l'utilisation de modèles de sécurité et de contrôle d'accès configurés de façon appropriée.

10. Références

10.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2279] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", janvier 1998. (*Obsolète, voir* RFC3629) *(D.S.)*

[RFC2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure des informations de gestion, version 2 (SMIv2)", avril 1999. (STD0058)

[RFC2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Conventions textuelles pour SMIv2", avril 1999. (STD0058)

[RFC2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Déclarations de conformité pour SMIv2", avril 1999. (STD0058)

[RFC3412] J. Case et autres, "Traitement et distribution de message pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3413] D. Levi, P. Meyer et B. Stewart, "Applications du protocole simple de gestion de réseau (SNMP)", STD 62, décembre 2002.

[RFC3414] U. Blumenthal, B. Wijnen, "Modèle de sécurité fondée sur l'utilisateur (USM) pour la version 3 du protocole simple de gestion de réseau (SNMPv3)", décembre 2002. (STD0062)

[RFC3415] B. Wijnen, R. Presuhn, K. McCloghrie, "Modèle de contrôle d'accès fondé sur la vue (VACM) pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3416] R. Presuhn, éd., "Version 2 des opérations de protocole pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3417] R. Presuhn, éd. "Transpositions de transport pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (*MàJ par* RFC4789) (STD0062)

[RFC3418] R. Presuhn, éd., "Base de données d'informations de gestion (MIB) pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

10.2 Références pour information

[RFC1155] M. Rose et K. McCloghrie, "Structure et identification des informations de gestion pour les internets fondés sur TCP/IP", STD 16, mai 1990.

[RFC1157] J. Case, M. Fedor, M. Schoffstall et J. Davin, "Protocole simple de gestion de réseau", STD 15, mai 1990. *(Historique)*

[RFC1212] M. Rose et K. McCloghrie, "Définitions concises de MIB", STD 16, février 1991.

[RFC1901] J. Case, K. McCloghrie, M. Rose, S. Waldbusser "Introduction à SNMPv2 fondé sur la communauté", janvier 1996. *(Historique)*

[RFC1909] K. McCloghrie, éd. "Infrastructure administrative pour SNMPv2", février 1996. *(Historique)*

[RFC1910] G. Waters, éd., "Modèle de sécurité fondé sur l'utilisateur pour SNMPv2", février 1996. *(Historique)*

[RFC2028] R. Hovey, S. Bradner, "Les organisations impliquées dans le processus de normalisation de l'IETF", octobre 1996. (*MàJ par* RFC3668, RFC3979) (BCP0011)

[RFC2576] R. Frye, D. Levi, S. Routhier, B. Wijnen, "Coexistence entre les version 1, version 2 et version 3 du cadre de gestion de réseau de l'Internet" mars 2000. (*Obsolète, voir* RFC3584) *(P.S.)*

[RFC2863] K. McCloghrie, F. Kastenholz, "MIB de groupe Interfaces", juin 2000. *(D.S.)*

[RFC3410] J. Case et autres, "Introduction et déclarations d'applicabilité pour le cadre de gestion standard de l'Internet", décembre 2002. *(Information)*

Appendice A Lignes directrices pour les concepteurs de modèles

Cet appendice décrit des lignes directrices pour les concepteurs de modèles qui sont destinés à entrer dans l'architecture définie par le présent document.

SNMPv1 et SNMPv2c sont deux cadres SNMP qui utilisent des concepts communs pour assurer une authentification et un contrôle d'accès triviaux. Les cadres SNMPv1 et SNMPv2c peuvent coexister avec des cadres conçus conformément à cette architecture, et des versions modifiées des cadres SNMPv1 et SNMPv2c pourraient être conçus pour satisfaire aux exigences de cette architecture, mais le présent document ne fournit pas de lignes directrices pour cette coexistence.

Au sein de tout modèle de sous système, il ne devrait pas y avoir de référence à un modèle spécifique d'un autre sous système, ou à des données définies par un modèle spécifique d'un autre sous système.

Le transfert de données entre les sous systèmes est délibérément décrit comme un ensemble fixe d'éléments de données abstraits et de fonctions de primitives qui peuvent être surchargées pour satisfaire aux besoins de plusieurs définitions de modèle.

Les documents qui définissent des modèles à utiliser au sein de cette architecture DEVRAIENT utiliser les primitives standard entre sous systèmes, éventuellement en définissant des mécanismes spécifiques pour convertir les éléments de données abstraits dans des formats utilisables par les modèles. Cette contrainte existe pour permettre que les documents de sous système et de modèle soient rédigés en tenant compte des frontières communes des sous systèmes et des modèles. Les fabricants ne sont pas obligés de reconnaître ces frontières dans leurs mises en œuvre.

L'architecture définit certains services standard que doivent fournir tous les sous systèmes, et l'architecture définit des interfaces de service abstraites pour demander ces services.

Chaque définition de modèle pour un sous système DEVRAIT prendre en charge les interfaces de service standard, mais la façon dont il effectue le service dépend de la définition du modèle.

A.1 Exigence de conception du modèle de sécurité

A.1.1 Menaces

Un document décrivant un modèle de sécurité DOIT décrire comment le modèle protège contre les menaces décrites au paragraphe 1.4 "Exigences de sécurité de cette architecture".

A.1.2 Traitement de sécurité

Les messages reçus DOIVENT être validés par un modèle de sous système de sécurité. La validation inclut l'authentification et si nécessaire le traitement de la confidentialité, mais il est explicitement permis d'envoyer des messages qui n'exigent ni authentification ni confidentialité.

Un message reçu contient un securityLevel *(niveau de sécurité)* spécifié à utiliser durant le traitement. Tout message qui exige la confidentialité DOIT aussi exiger l'authentification.

Un modèle de sécurité spécifie les règles par lesquelles l'authentification et la confidentialité sont réalisées. Un modèle peut définir des mécanismes pour fournir des caractéristiques de sécurité supplémentaires, mais la définition du modèle est contrainte d'utiliser les éléments de données abstraites (éventuellement un sous ensemble de ceux-ci) définis dans le présent document pour transférer les données entre les sous systèmes.

Chaque modèle de sécurité peut permettre d'utiliser plusieurs protocoles de sécurité à utiliser concurremment au sein d'une mise en œuvre du modèle. Chaque modèle de sécurité définit comment déterminer quel protocole utiliser, étant donné le niveau de sécurité et les paramètres de sécurité pertinents pour le message. Chaque modèle de sécurité, avec son ou ses protocoles associés définit comment les entités envoyeuses/receveuses sont identifiées, et comment sont configurés les secrets.

Les protocoles d'authentification et de confidentialité pris en charge par les modèles de sécurité sont identifiés de façon univoque en utilisant des identifiants d'objet. Les protocoles standard de l'IETF pour l'authentification ou la confidentialité devraient avoir un identifiant défini au sein des sous arborescences snmpAuthProtocols ou snmpPrivProtocols. Les identifiants de protocole spécifiques d'entreprise devraient être définis au sein de la sous arborescence de l'entreprise.

Pour la confidentialité, le modèle de sécurité définit quelle portion du message est chiffrée.

Les données persistantes utilisées pour la sécurité devraient être gérables par SNMP, mais le modèle de sécurité définit si une instance de la MIB est une exigence de conformité.

Les modèles de sécurité sont remplaçables au sein du sous système de sécurité. Plusieurs mises en œuvre de modèle de sécurité peuvent exister concurremment au sein d'un moteur SNMP. Le nombre de modèles de sécurité définis par la communauté SNMP devrait rester assez petit pour promouvoir l'interopérabilité.

A.1.3 Valider le marquage de sécurité dans un message reçu

Un modèle de traitement de message demande qu'un modèle de sécurité :
- vérifie que le message n'a pas été altéré,
- authentifie l'identification du principal pour qui le message a été généré.
- déchiffre le message si il a été chiffré.

Des exigences supplémentaires peuvent être définies par le modèle, et des services supplémentaires peuvent être fournis par le modèle, mais le modèle est contraint d'utiliser les primitives suivantes pour transférer les données entre les sous systèmes. Les mises en œuvre ne subissent pas les mêmes contraintes.

Un modèle de traitement de message utilise la primitive processIncomingMsg, comme décrit au paragraphe 4.4.2.

A.1.4 Les MIB de sécurité

Chaque modèle de sécurité définit le ou les modules de MIB requis pour le traitement de la sécurité, incluant tous modules de MIB requis pour le ou les protocoles de sécurité pris en charge. Le ou les modules de MIB DEVRAIENT être définis concurremment avec les procédures qu'utilisent le ou les modules de MIB. Le ou les modules de MIB sont soumis aux règles normales de contrôle d'accès.

La transposition entre l'identifiant de sécurité dépendant du modèle et le securityName DOIT pouvoir être déterminée à l'aide de SNMP, si la MIB qui dépend du modèle est instanciée et si la politique de contrôle d'accès permet l'accès.

A.1.5 Données de sécurité mises en antémémoire

Pour chaque message reçu, le modèle de sécurité met en antémémoire les informations d'état afin qu'un message de réponse puisse être généré en utilisant les mêmes informations de sécurité, même si le magasin de données de configuration locale est altéré entre le moment de la demande entrante et la sortie de la réponse.

Un modèle de traitement de message a la responsabilité de libérer explicitement les données en antémémoire si de telles données ne sont plus nécessaires. Pour permettre cela, un élément de données abstrait securityStateReference est passé du modèle de sécurité au modèle de traitement de message.

Les données de sécurité en antémémoire peuvent être implicitement libérées via la génération d'une réponse, ou explicitement libérées en utilisant la primitive stateRelease, comme décrit au paragraphe 4.5.1.

A.2 Exigences de conception du modèle de traitement de message

Un moteur SNMP contient un sous système de traitement de message qui peut contenir plusieurs modèles de traitement de message.

Le modèle de traitement de message DOIT toujours (conceptuellement) passer la PDU complète, c'est-à-dire, il ne transmet jamais moins que la liste complète de varBinds.

A.2.1 Réception d'un message SNMP du réseau

À réception d'un message provenant du réseau, le répartiteur dans le moteur SNMP détermine la version du message SNMP et interagit avec le modèle de traitement de message correspondant pour déterminer les éléments de données abstraits.

Un modèle de traitement de message spécifie le format de message SNMP qu'il prend en charge et décrit comment déterminer les valeurs des éléments de données abstraits (comme msgID, msgMaxSize, msgFlags, msgSecurityParameters, securityModel, securityLevel, etc.). Un modèle de traitement de message interagit avec un modèle de sécurité pour fournir le traitement de sécurité pour le message en utilisant la primitive processIncomingMsg, comme décrit au paragraphe 4.4.2.

A.2.2 Envoi d'un message SNMP au réseau

Le répartiteur dans le moteur SNMP interagit avec un modèle de traitement de message pour préparer un message sortant. Il utilise pour cela les primitives suivantes :
- pour les demandes et notifications : prepareOutgoingMessage, comme décrit au paragraphe 4.2.1.
- pour les messages de réponse : prepareResponseMessage, comme décrit au paragraphe 4.2.2.

Un modèle de traitement de message, lorsque il prépare un message SNMP sortant, interagit avec un modèle de sécurité

pour sécuriser le message. Il utilise pour cela les primitives suivantes :
- pour les demandes et notifications : generateRequestMsg, comme décrit au paragraphe 4.4.1.
- pour les messages de réponse : generateResponseMsg, comme décrit au paragraphe 4.4.3.

Une fois que le message SNMP est préparé par un modèle de traitement de message, le répartiteur envoie le message à l'adresse désirée en utilisant le transport approprié.

A.3 Exigences pour la conception des applications

Au sein d'une application, il peut y avoir un lien explicite à une version spécifique de message SNMP, c'est-à-dire, un modèle spécifique de traitement de message, et à un modèle spécifique de contrôle d'accès, mais il ne devrait y avoir aucune référence à des données définies par un modèle spécifique de traitement de message ou de contrôle d'accès.

Au sein d'une application, il ne devrait y avoir aucune référence à un modèle de sécurité spécifique, ou à des données définies par un modèle de sécurité spécifique.

Une application détermine si un contrôle d'accès explicite ou implicite devrait être appliqué à l'opération, et, si le contrôle d'accès est nécessaire, quel modèle de contrôle d'accès devrait être utilisé.

Une application a la responsabilité de définir tous modules de MIB utilisés pour fournir les services spécifiques de l'application.

Les applications interagissent avec le moteur SNMP pour initier des messages, recevoir les réponses, recevoir des messages asynchrones, et envoyer des réponses.

A.3.1 Applications qui initient des messages

Des applications peuvent demander que le moteur SNMP envoie des messages contenant des commandes ou des notifications SNMP qui utilisent la primitive sendPdu, comme décrit au paragraphe 4.1.1.

Si il est désiré qu'un message soit envoyé à plusieurs cibles, il est de la responsabilité de l'application de fournir l'itération.

Le moteur SNMP suppose que le contrôle d'accès nécessaire a été appliqué à la PDU, et ne fournit aucun service de contrôle d'accès.

Le moteur SNMP cherche le paramètre "expectResponse", et si une réponse est attendue, les informations appropriées sont alors mises en antémémoire de façon qu'une réponse ultérieure puisse être associée à ce message, et puisse ensuite être retournée à l'application. Une sendPduHandle est retournée à l'application afin qu'elle puisse ultérieurement faire aussi correspondre la réponse avec ce message.

A.3.2 Applications qui reçoivent des réponses

Le moteur SNMP confronte les messages de réponse entrants avec les messages en instance envoyés par le moteur SNMP, et transmet la réponse à l'application associée en utilisant la primitive processResponsePdu, comme décrit au paragraphe 4.1.4.

A.3.3 Applications qui reçoivent des messages asynchrones

Lorsque un moteur SNMP reçoit un message qui n'est pas la réponse à une demande de ce moteur SNMP, il doit déterminer à quelle application le message devrait être donné.

Une application qui souhaite recevoir des messages asynchrones s'enregistre auprès du moteur en utilisant la primitive registerContextEngineID, comme décrit au paragraphe 4.1.5.

Une application qui souhaite cesser de recevoir des messages asynchrones devrait se désenregistrer du moteur SNMP en utilisant la primitive unregisterContextEngineID, comme décrit au paragraphe 4.1.5.

Un seul enregistrement par combinaison de type de PDU et contextEngineID est permise en même temps. Les enregistrements dupliqués sont ignorés. Une errorIndication *(indication d'erreur)* sera retournée à l'application qui tente de dupliquer un enregistrement.

Tous les messages reçus de façon asynchrone qui contiennent une combinaison enregistrée de type de PDU et de contextEngineID sont envoyés à l'application qui s'est enregistrée pour prendre en charge cette combinaison.

Le moteur transmet la PDU à l'application enregistrée, en utilisant la primitive processPdu, comme décrit au paragraphe 4.1.2.

A.3.4 Applications qui envoient des réponses

Les opérations de demande exigent des réponses. Une application envoie une réponse via la primitive returnResponsePdu, comme décrit au paragraphe 4.1.3.

Les paramètres contextEngineID, contextName, securityModel, securityName, securityLevel, et stateReference viennent de la primitive initialeprocessPdu. La PDU et les statusInformation sont le résultat du traitement.

A.4 Exigences pour la conception de modèle de contrôle d'accès

Un modèle de contrôle d'accès détermine si le securityName spécifié est autorisé à effectuer l'opération demandée sur un objet géré spécifié. Le modèle de contrôle d'accès spécifie les règles par lesquelles le contrôle d'accès est déterminé.

Les données persistantes utilisées pour le contrôle d'accès devraient être gérables en utilisant SNMP, mais le modèle de contrôle d'accès définit si une instanciation de la MIB est une exigence de conformité.

Le modèle de contrôle d'accès doit fournir la primitive isAccessAllowed.

Adresse des éditeurs

Bert Wijnen
Lucent Technologies
Schagen 33
3461 GL Linschoten
Netherlands

téléphone : +31 348-680-485
mél : bwijnen@lucent.com

David Harrington
Enterasys Networks
Post Office Box 5005
35 Industrial Way
Rochester, New Hampshire 03866-5005
USA
téléphone : +1 603-337-2614
mél : dbh@enterasys.com

Randy Presuhn
BMC Software, Inc.
2141 North First Street
San Jose, California 95131
USA
téléphone : +1 408-546-1006
Fax : +1 408-965-0359
mél : randy_presuhn@bmc.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.