Groupe de travail Réseau
Request for Comments : 3413
STD : 62
RFC rendue obsolète : 2573
Catégorie : Norme

D. Levi, Nortel Networks
P. Meyer, Secure Computing Corporation
B. Stewart, retraité
décembre 2002
Traduction Claude Brière de L'Isle

Applications du protocole simple de gestion de réseau (SNMP)

Résumé
Le présent document décrit cinq types d'applications du protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*) qui utilisent un moteur SNMP tel que décrit dans le STD 62, [RFC3411]. Les types d'application décrits sont les générateurs de commandes, les répondeurs de commandes, les générateurs de notifications, les receveurs de notification et les transmetteurs mandataires.
Le présent document définit aussi des modules de base de données d'informations de gestion (MIB, *Management Information Base*) pour la spécification de cibles d'opérations de gestion, pour le filtrage des notifications, et pour la transmission par mandataire. Le présent document rend obsolète la [RFC2573].

Table des Matières

1. Généralités

Le présent document décrit cinq types d'applications SNMP :
- les applications qui initient des demandes SNMP de classe Lecture, et/ou Écriture, appelées "générateurs de commandes",
- les applications qui répondent aux demandes SNMP de classe Lecture, et/ou Écriture, appelées "répondeurs de commandes",
- les applications qui génèrent des PDU SNMP de classe Notification, appelées des "générateurs de notification",
- les applications qui reçoivent des PDU SNMP de classe Notification, appelées des "receveurs de notification",
- les applications qui transmettent des messages SNMP, appelées des "transmetteurs mandataires".

Noter qu'il n'y a pas de restriction sur les types d'applications qui peuvent être associés à un moteur SNMP particulier. Par exemple, un seul moteur SNMP peut, en fait, être associés à un générateur de commandes et à une application de répondeur de commandes.

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

1.1 Applications de générateur de commande

Une application de générateur de commandes initie des demandes SNMP de classe Lecture et/ou Écriture, et traite les réponses aux demandes qu'elle a générées.

1.2 Applications de répondeur de commande

Une application de répondeur de commande reçoit des demandes SNMP de classe Lecture et/ou Écriture destinées au système local comme indiqué par le fait que le contextEngineID dans la demande reçue est égal à celui du moteur local à travers lequel a été reçue la demande. L'application de répondeur de commande va effectuer l'opération de protocole appropriée, en utilisant le contrôle d'accès, et va générer un message de réponse à envoyer au générateur de la demande.

1.3 Applications de générateur de notification

Une application de générateur de notifications surveille conceptuellement sur un système des événements ou conditions particuliers, et génère des messages de classe Notification sur la base de ces événements ou conditions. Le générateur de la notification doit avoir un mécanisme pour déterminer où envoyer les messages, et quelle version SNMP et paramètres de sécurité utiliser lors de l'envoi des messages. Un mécanisme et un module de MIB sont fournis à cette fin dans le présent document. Noter que les PDU de classe Notification générés par un générateur de notification peuvent être des types de PDU de classe Confirmée ou Non confirmée.

1.4 Applications de receveur de notification

Une application de receveur de notifications écoute les messages de notification, et génère des messages de réponse lorsque un message contenant une PDU de classe Confirmée est reçue.

1.5 Applications de transmetteur mandataire

Une application de transmetteur mandataire transmet les messages SNMP. Noter que la mise en œuvre d'une application de transmetteur mandataire est facultative. Les paragraphes qui décrivent le mandataire (3.5, 4.3, et 7) peuvent être sautés pour les mises en œuvre qui n'incluent pas d'application de transmetteur mandataire.

Le terme "mandataire" a été historiquement utilisé de façon très lâche, avec des significations différentes. Ces significations différentes incluent (entre autres) :

(1) la transmission de demandes SNMP à d'autres entités SNMP sans égard au type d'objet géré accédé ; par exemple, afin de transmettre une demande SNMP d'un domaine de transport à un autre, ou pour traduire une demande SNMP d'une version en demande SNMP d'une autre version;

(2) la traduction de demandes SNMP en opérations d'un protocole de gestion non SNMP ;

(3) la pris en charge des objets gérés agrégés où la valeur d'une instance d'objet géré dépend des valeurs de plusieurs autres éléments (distants) d'informations de gestion.

Chacun de ces scénarios peut être avantageux ; par exemple, la prise en charge de l'agrégation d'informations de gestion peut réduire significativement les exigences de bande passante d'activités de gestion à grande échelle.

Cependant, utiliser un seul terme pour couvrir plusieurs scénarios différents cause la confusion.

Pour éviter une telle confusion, le présent document utilise le terme "mandataire" avec une signification définie de façon plus stricte. Le terme de "mandataire" est utilisé dans le présent document pour se référer à un mandataire d'application de transmetteur qui transmet les messages SNMP sans égard aux objets gérés contenus dans ces messages. Cette définition est en rapport plus étroit avec la première des définitions ci-dessus. Noter cependant que dans l'architecture SNMP [RFC3411], un transmetteur mandataire est en fait une application, et n'a pas besoin d'être associé à ce qui est traditionnellement vu comme un agent SNMP.

Précisément, la distinction entre un agent SNMP traditionnel et un mandataire d'application de transmetteur est simple :
- un mandataire d'application de transmetteur transmet les messages SNMP aux autres moteurs SNMP conformément au contexte, et sans égard aux types spécifiques d'objets gérés auxquels on accède, et transmet la réponse aux messages précédemment transmis en retour au moteur SNMP d'où le message d'origine a été reçu ;
- à l'opposé, l'application de répondeur de commande qui fait partie de ce qui est traditionnellement vu comme un agent SNMP, et qui traite les demandes SNMP conformément aux (noms des) types individuels d'objets gérés et instances auxquelles on accède, N'EST PAS un mandataire d'application de transmetteur du point de vue du présent document.

Donc, lorsque un mandataire d'application de transmetteur transmet une demande ou une notification pour une certaine paire contextEngineID/contextName, non seulement les informations sur la façon de transmettre la demande sont spécifiquement associées à ce contexte, mais le mandataire d'application de transmetteur n'a pas besoin d'une définition détaillée d'une vue de MIB (car le mandataire d'application de transmetteur transmet la demande sans égard aux types d'objets gérés).

À l'opposé, une application de répondeur de commande doit avoir la définition détaillée de la vue de MIB, et même si elle a besoin de produire des demandes à d'autres entités, via SNMP ou autrement, ce besoin dépend des instances individuelles d'objets gérés auxquelles on accède (c'est-à-dire, pas seulement sur le contexte).

Noter qu c'est un objectif de conception d'un mandataire d'application de transmetteur d'agir comme intermédiaire entre les points d'extrémité d'une transaction. En particulier, lors de la transmission de messages de classe de notification Confirmée, la réponse associée est transmise lorsque elle est reçue de la cible à laquelle le message de classe Notification a été transmis, plutôt que de générer une réponse immédiatement lorsque le message de classe Notification est reçu.

2. Cibles de la gestion

Certains types d'applications (en particulier générateurs de notification et transmetteurs mandataires) exigent un mécanisme pour déterminer où et comment envoyer les messages générés. Le présent document fournit un mécanisme et un module de MIB à cette fin. L'ensemble d'informations qui décrit où et comment envoyer un message est appelé une "cible de gestion", et consiste en deux sortes d'informations :
- Informations de destination , consistant en un domaine de transport et une adresse de transport. Ceci est aussi appelé un point d'extrémité de transport.
- Paramètres SNMP, consistant en un modèle de traitement de message, un modèle de sécurité, un niveau de sécurité, et des informations de nom de sécurité.

Le module de MIB SNMP TARGET décrit plus loin dans le présent document contient un tableau pour chacun de ces types d'informations. Il peut y avoir une relation de plusieurs à plusieurs dans la MIB entre ces deux types d'informations. C'est-à-dire qu'il peut y avoir plusieurs points d'extrémité de transport associés à un ensemble particulier de paramètres SNMP, ou un certain point d'extrémité de transport peut être associé à plusieurs ensembles de paramètres SNMP.

3. Éléments de procédure

Les paragraphes qui suivent décrivent les procédures de chaque type d'application lors de la génération de messages à transmettre ou lors du traitement des messages reçus. Les applications communiquent avec le répartiteur en utilisant les interfaces de service abstraites définies dans la [RFC3411].

3.1 Applications de générateur de commande

Un générateur de commandes initie une demande SNMP en invoquant le répartiteur à l'aide de l'interface de service abstraite (ASI, *Abstract Service Interface*) suivante :

```
statusInformation =   sendPdu(     -- sendPduHandle si succès, errorIndication si
                                   échec
  IN   transportDomain             -- domaine de transport à utiliser
  IN   transportAddress            -- adresse du réseau de destination
  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/à cette entité
  IN   contextName                 -- données de/dans ce contexte
  IN   pduVersion                  -- la version de la PDU
  IN   PDU                         -- unité de données de protocole SNMP
  IN   expectResponse              -- VRAI ou FAUX
      )
```

Où :
- Le transportDomain est celui de la destination du message.
- La transportAddress est celle de la destination du message.
- Le messageProcessingModel indique quel modèle de traitement de message l'application souhaite utiliser.
- Le securityModel est le modèle de sécurité que l'application souhaite utiliser.
- Le securityName est le nom indépendant du modèle de sécurité pour le principal au nom duquel l'application souhaite que le message soit généré.
- Le securityLevel est le niveau de sécurité que l'application souhaite utiliser.
- Le contextEngineID spécifie la localisation des informations de gestion qu'il demande. Noter que sauf si la demande est envoyée à un mandataire, cette valeur sera habituellement égale à la valeur du snmpEngineID du moteur auquel la demande est envoyée.
- Le contextName spécifie le nom du contexte local pour les informations de gestion qu'il demande.
- La pduVersion indique la version de la PDU à envoyer.
- La PDU est une valeur construite par le générateur de commandes contenant l'opération de gestion que le générateur de commandes souhaite effectuer.
- L'argument expectResponse indique qu'une réponse est attendue.

Le résultat de l'interface sendPdu indique si la PDU a été envoyée avec succès. Si elle l'a été, la valeur retournée sera une sendPduHandle. Le générateur de commandes devrait mémoriser la sendPduHandle afin de pouvoir corréler une réponse à la demande d'origine.

Le répartiteur est chargé de livrer la réponse à une certaine demande à l'application de générateur de commandes correcte. L'interface de service abstraite utilisée est :

```
processResponsePdu(                -- traiter la PDU de réponse
  IN   messageProcessingModel      -- normalement, la version SNMP
  IN   securityModel               -- modèle de sécurité utilisé
  IN   securityName                -- au nom de ce principal
  IN   securityLevel               -- niveau de sécurité
  IN   contextEngineID             -- données de/à 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           -- succès ou errorIndication
  IN   sendPduHandle               -- bride provenant de sendPdu
      )
```

Où :
- Le messageProcessingModel est la valeur provenant de la réponse reçue.
- Le securityModel est la valeur provenant de la réponse reçue.
- Le securityName est la valeur provenant de la réponse reçue.
- Le securityLevel est la valeur provenant de la réponse reçue.
- Le contextEngineID est la valeur provenant de la réponse reçue.
- Le contextName est la valeur provenant de la réponse reçue.
- La pduVersion indique la version de la PDU dans la réponse reçue.
- La PDU est la valeur provenant de la réponse reçue.
- Les statusInformation indiquent le succès ou l'échec de la réception de la réponse.

- La sendPduHandle est la valeur retournée par l'invocation de sendPdu qui a généré la demande d'origine à laquelle ceci est une réponse.

La procédure lorsque un générateur de commandes reçoit un message est la suivante :
(1) Si les valeurs reçues de messageProcessingModel, securityModel, securityName, contextEngineID, contextName, et pduVersion ne sont pas toutes égales aux valeurs utilisées dans la demande d'origine, la réponse est éliminée.
(2) Le type d'opération, le request-id, les error-status, l'error-index, et les variable-bindings sont extraits de la PDU et sauvegardés. Si le request-id n'est pas égal à la valeur utilisée dans la demande d'origine, la réponse est éliminée.
(3) À ce point, il appartient à l'application de prendre l'action appropriée. L'action spécifique dépend de la mise en œuvre. Si les statusInformation indiquent que la demande a échoué, une action appropriée pourrait tenter de transmettre à nouveau la demande, ou de notifier à la personne qui gère l'application qu'un échec s'est produit.

3.2 Applications de répondeur de commande

Avant qu'une application de répondeur de commande puisse traiter des messages, elle doit d'abord s'associer à un moteur SNMP. L'interface de service abstraite utilisée à cette fin est :

```
statusInformation =                                    -- succès ou errorIndication
registerContextEngineID(registerContextEngineID(
  IN   contextEngineID                                 -- prend la responsabilité de celui-ci
  IN   pduType                                         -- le ou les pduType à enregistrer
      )
```

Où :
- Les statusInformation indiquent le succès ou l'échec de la tentative d'enregistrement.
- Le contextEngineID est égal au snmpEngineID du moteur SNMP avec lequel le répondeur de commandes s'enregistre.
- Le pduType indique une PDU Read-Class et/ou Write-Class.

Noter que si une autre application de répondeur de commande est déjà enregistrée auprès du moteur SNMP, toute autre tentative d'enregistrement avec les mêmes contextEngineID et pduType sera refusée. Cela implique que des applications de répondeur de commandes séparées pourraient s'enregistrer séparément pour les divers types de PDU. Cependant, en pratique, ceci n'est pas souhaitable, et une seule application de répondeur de commande devrait être enregistrée auprès d'un moteur SNMP à un moment donné.
Une application de répondeur de commande peut se dissocier d'un moteur SNMP en utilisant l'interface de service abstraite suivante :

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

Où :
- Le contextEngineID est égal au snmpEngineID du moteur SNMP avec lequel le répondeur de commandes annule l'enregistrement.
- Le pduType indique une PDU Read-Class et/ou Write-Class.

Une fois que le répondeur de commandes s'est enregistré auprès du moteur SNMP, il attend de recevoir des messages SNMP. L'interface de service abstraite utilisée pour recevoir des messages est :

```
processPdu(                          -- traiter la PDU Request/Notification
  IN   messageProcessingModel        -- normalement, la version SNMP
  IN   securityModel                 -- modèle de sécurité utilisé
  IN   securityName                  -- au nom de ce principal
  IN   securityLevel                 -- niveau de sécurité
  IN   contextEngineID               -- données de/à 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 maximale de la PDU de réponse
  IN   stateReference                -- référence aux informations d'état nécessaires lors de l'envoi d'une
                                     réponse
  )
```

Où :
- Le messageProcessingModel indique quel modèle de traitement de message a reçu et traité le message.
- Le securityModel est la valeur provenant du message reçu.
- Le securityName est la valeur provenant du message reçu.
- Le securityLevel est la valeur provenant du message reçu.
- Le contextEngineID est la valeur provenant du message reçu.
- Le contextName est la valeur provenant du message reçu.
- La pduVersion indique la version de la PDU dans le message reçu.
- La PDU est la valeur provenant du message reçu.
- La maxSizeResponseScopedPDU est la taille maximum admissible d'une ScopedPDU contenant une PDU de réponse (sur la base de la taille maximum de message que peut accepter l'origine du message).
- La stateReference est une valeur qui fait référence aux informations en antémémoire sur chaque message de demande reçu. Cette valeur doit être retournée au répartiteur afin de générer une réponse.

La procédure à réception d'un message est la suivante :

(1) Le type d'opération est déterminé à partir de la valeur d'étiquette ASN.1 associée au paramètre PDU. Le type d'opération devrait toujours être d'un des types précédemment enregistrés par l'application.

(2) Le request-id est extrait de la PDU et sauvegardé.

(3) Tous les paramètres spécifiques d'un type de PDU sont extraits de la PDU et sauvegardés (par exemple, si le type de PDU est une PDU SNMPv2 GetBulk, les valeurs de non-repeaters et max-repetitions sont extraites).

(4) Les variable-bindings sont extraits de la PDU et sauvegardés.

(5) L'opération de gestion représentée par le type de PDU est effectuée par rapport à la vue de MIB pertinente dans le contexte désigné par le contextName (pour un type de PDU SNMPv2, l'opération est effectuée conformément aux procédures établies dans la [RFC1905]). La vue de MIB pertinente est déterminée par le securityLevel, le securityModel, le contextName, le securityName, et la classe du type de PDU. Pour déterminer si une instance d'objet particulière est dans la vue de MIB pertinente, on invoque l'interface de service abstraite suivante :

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

Où :
- Le securityModel est la valeur provenant du message reçu.
- Le securityName est la valeur provenant du message reçu.
- Le securityLevel est la valeur provenant du message reçu.
- Le viewType indique si le type de PDU est une opération de classe Lecture ou Écriture.
- Le contextName est la valeur provenant du message reçu.
- Le variableName est l'instance d'objet de la variable pour laquelle les droits d'accès sont vérifiés.

Normalement, le résultat de l'opération de gestion sera une nouvelle valeur de PDU, et le traitement se poursuit à l'étape (6) ci-dessous. Cependant, à tout moment durant le traitement de l'opération de gestion :
- Si l'ASI isAccessAllowed retourne une erreur noSuchView, noAccessEntry, ou noGroupName, le traitement de l'opération de gestion est arrêté, une valeur de PDU est construite en utilisant les valeurs de la PDU reçue à l'origine, mais en remplaçant l'état d'erreur par un code authorizationError, et une valeur d'indice d'erreur de 0, et le contrôle est passé à l'étape (6) ci-dessous.
- Si l'ASI isAccessAllowed retourne une otherError, le traitement de l'opération de gestion est arrêté, une valeur de PDU différente est construite en utilisant les valeurs de la PDU reçue à l'origine, mais en remplaçant l'état d'erreur par un code genError et l'indice d'erreur par l'indice du lien variable échoué, et le contrôle est passé à l'étape (6) ci-dessous.
- Si l'ASI isAccessAllowed retourne une erreur noSuchContext, le traitement de l'opération de gestion est arrêté,

aucune PDU de résultat n'est générée, le compteur snmpUnknownContexts est incrémenté, et le contrôle est passé à l'étape (6) ci-dessous pour générer un message de rapport.

- Si le contexte désigné par le paramètre contextName est indisponible, le traitement de l'opération de gestion est arrêté, aucune PDU de résultat n'est générée, le compteur snmpUnavailableContexts est incrémenté, et le contrôle est passé à l'étape (6) ci-dessous pour générer un message de rapport.

(6) Le répartiteur est invoqué pour générer un message de réponse ou de rapport. L'interface de service abstraite est :

```
returnResponsePdu(
  IN  messageProcessingModel      -- normalement, la version SNMP
  IN  securityModel               -- modèle de sécurité utilisé
  IN  securityName                -- au nom de ce principal
  IN  securityLevel               -- le même que pour une demande entrante
  IN  contextEngineID             -- données de/à 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 maximale de la PDU de réponse
  IN  stateReference              -- référence aux informations d'état telles que présentées avec la
                                  demande
  IN  statusInformation            -- succès ou errorIndication ; OID/valeur de compteur d'erreur si erreur
      )
```

Où :

- Le messageProcessingModel est la valeur provenant de l'invocation de processPdu.
- Le securityModel est la valeur provenant de l'invocation de processPdu.
- Le securityName est la valeur provenant de l'invocation de processPdu.
- Le securityLevel est la valeur provenant de l'invocation de processPdu.
- Le contextEngineID est la valeur provenant de l'invocation de processPdu.
- Le contextName est la valeur provenant de l'invocation de processPdu.
- La pduVersion indique la version de la PDU à retourner. Si il n'a pas été généré de PDU de résultat, la pduVersion est une valeur indéfinie.
- La PDU est le résultat généré à l'étape (5) ci-dessus. Si il n'a pas été généré de PDU de résultat, la PDU est une valeur indéfinie.
- La maxSizeResponseScopedPDU est une valeur locale qui indique la taille maximum d'une ScopedPDU que l'application peut accepter.
- La stateReference est la valeur provenant de l'invocation de la processPdu.
- Les statusInformation contiennent soit l'indication qu'aucune erreur n'est survenue et qu'une réponse devrait être générée, soit l'indication qu'une erreur s'est produite, avec l'OID et la valeur de compteur de l'objet de compteur d'erreur approprié.

Noter qu'une application de répondeur de commande devrait toujours invoquer l'interface de service abstraite returnResponsePdu, même dans le cas d'une erreur telle que d'allocation de ressource. Dans le cas d'une telle erreur, la valeur de PDU passée à returnResponsePdu devrait contenir les valeurs appropriées pour errorStatus et errorIndex.

Noter que le texte ci-dessus décrit des situations où le compteur snmpUnknownContexts est incrémenté, et où le compteur snmpUnavailableContexts est incrémenté. La différence entre ces compteurs est que snmpUnknownContexts est incrémenté lorsque une demande est reçue pour un contexte inconnu de l'entité SNMP. Le compteur snmpUnavailableContexts est incrémenté lorsque une demande est reçue pour un contexte connu de l'entité SNMP, mais qui est actuellement indisponible. Déterminer quand un contexte est indisponible est spécifique de la mise en œuvre, et certaines mises en œuvre peuvent ne jamais rencontrer cette situation, et donc ne jamais incrémenter le compteur snmpUnavailableContexts.

3.3 Applications de générateur de notification

Une application de générateur de notification génère des messages SNMP contenant des PDU de classe Notification (par exemple, des PDU SNMPv2 Trap ou Inform). Il n'y a pas d'exigence quant aux types spécifiques de PDU de classe Notification qu'une mise en œuvre particulière doit être capable de générer.

Les applications de générateur de notification exigent un mécanisme pour identifier les cibles de gestion auxquelles les notifications devraient être envoyées. Le mécanisme particulier utilisé dépend de la mise en œuvre. Cependant, si une mise en œuvre rend gérable la configuration de cibles de gestion SNMP, elle DOIT utiliser le module de MIB SNMP TARGET décrit dans le présent document.

Lorsque un générateur de notification souhaite générer une notification, il doit d'abord déterminer dans quel contexte les informations à transporter existent dans la notification, c'est-à-dire, il doit déterminer le contextEngineID et le contextName. Il doit ensuite déterminer l'ensemble de cibles de gestion auxquelles la notification devrait être envoyée. L'application doit aussi déterminer, pour chaque cible de gestion, quel type de PDU spécifique le message de notification devrait contenir, et si il doit contenir une PDU de classe Confirmée, le nombre d'essais et l'algorithme de retransmission.

Le mécanisme par lequel un générateur de notification détermine ces informations dépend de la mise en œuvre. Une fois que l'application a déterminé ces informations, la procédure suivante est effectuée pour chaque cible de gestion :

(1) Tout mécanisme de filtrage approprié est appliqué pour déterminer si la notification devrait être envoyée à la cible de gestion. Si un tel mécanisme de filtrage détermine que la notification ne devrait pas être envoyée, le traitement continue avec la prochaine cible de gestion. Autrement,

(2) L'ensemble approprié de liens de variables est restitué de l'instrumentation locale de MIB au sein de la vue de MIB pertinente. La vue de MIB pertinente est déterminée par le niveau de sécurité, le modèle de sécurité, le nom de contexte, et le nom de sécurité de la cible de gestion. Pour déterminer si une instance d'objet particulière est dans la vue de MIB pertinente, on utilise l'interface de service abstraite isAccessAllowed, de la même manière que décrit dans le paragraphe précédant, excepté que le viewType indique une opération de classe Notification. Si les statusInformation retournées par isAccessAllowed n'indiquent pas accessAllowed, la notification n'est pas envoyée à la cible de gestion.

(3) L'identifiant d'objet TYPE DE NOTIFICATION de la notification (c'est la valeur de l'élément de liens de variable dont le nom est snmpTrapOID.0, c'est-à-dire, le second lien de variable) est vérifié en utilisant l'interface de service abstraite isAccessAllowed, en utilisant les mêmes paramètres qu'utilisés dans l'étape précédente. Si les statusInformation retournées par isAccessAllowed n'indiquent pas accessAllowed, la notification n'est pas envoyée à la cible de gestion.

(4) Une PDU est construite en utilisant une valeur d'identifiant de demande localement unique, un type de PDU comme déterminé par la mise en œuvre, un état d'erreur et une valeur d'indice d'erreur de 0, et les liens de variable fournis précédemment à l'étape (2).

(5) Si la notification contient une PDU de classe Non confirmé, le répartiteur est invoqué en utilisant l'interface de service abstraite suivante :

```
statusInformation =  sendPdu(       -- sendPduHandle si succès ; errorIndication si
                                    échec
  IN   transportDomain              -- domaine de transport à utiliser
  IN   transportAddress             -- adresse du réseau de destination
  IN                                -- normalement, la version SNMP
messageProcessingModel
  IN   securityModel                -- modèle de sécurité à utiliser
  IN   securityName                 -- au nom de ce principal
  IN   securityLevel                -- niveau de sécurité demandé
  IN   contextEngineID              -- données de/à cette entité
  IN   contextName                  -- données de/dans ce contexte
  IN   pduVersion                   -- la version de la PDU
  IN   PDU                          -- unité de données de protocole SNMP
  IN   expectResponse               -- VRAI ou FAUX
    )
```

Où :
- Le transportDomain est celui de la cible de gestion.
- La transportAddress est celle de la cible de gestion.
- Le messageProcessingModel est celui de la cible de gestion.
- Le securityModel est celui de la cible de gestion.
- Le securityName est celui de la cible de gestion.
- Le securityLevel est celui de la cible de gestion.
- Le contextEngineID est la valeur déterminée à l'origine pour la notification.
- Le contextName est la valeur déterminée à l'origine pour la notification.
- La pduVersion est la version de la PDU à envoyer.
- La PDU est la valeur construite à l'étape (4) ci-dessus.
- L'argument expectResponse indique qu'aucune réponse n'est attendue.

Autrement,

(6) Si la notification contient une PDU de classe Confirmée, alors :
   a) Le répartiteur est invoqué en utilisant l'interface de service abstraite sendPdu comme décrit à l'étape (5) ci-dessus, excepté que l'argument expectResponse indique qu'une réponse est attendue.
   b) L'application met en antémémoire les informations sur la cible de gestion.
   c) Si une réponse est reçue dans un délai approprié du point d'extrémité de transport de la cible de gestion, la notification est considérée comme acquittée et les informations en antémémoire sont éliminées. Autrement,
   d) si une réponse n'est pas reçue dans un délai approprié, ou si une indication de rapport est reçue, les informations sur la cible de gestion sont restituées de l'antémémoire, et les étapes a) à d) sont répétées. Le nombre de fois que ces étapes sont répétées est égal au compte de réessais déterminé précédemment. Si ce compte de répétitions est excédé, l'accusé de réception de la notification est considéré avoir échoué, et le traitement de la notification pour cette cible de gestion est arrêté. Noter que certaines indications de rapport pourraient être considérées avoir échoué. De telles indications de rapport devraient être interprétées comme signifiant que l'accusé de réception de la notification a échoué, et que les étapes a) à d) ne doivent pas être répétées.

Les réponses aux notifications de PDU de classe Confirmée seront reçues via l'interface de service abstraite processResponsePdu.

Pour résumer, les étapes qu'un générateur de notification suit pour déterminer où envoyer une notification sont :
- Déterminer les cibles auxquelles la notification devrait être envoyée.
- Appliquer tout filtrage requis à la liste des cibles.
- Déterminer quelles cibles sont autorisées à recevoir la notification.

3.4 Applications de receveur de notification

Les applications de receveur de notification reçoivent des messages Notification SNMP du répartiteur. Avant qu'aucun message puisse être reçu, le receveur de notification doit s'enregistrer auprès du répartiteur en utilisant l'interface de service abstraite registerContextEngineID. Les paramètres utilisés sont :
- Le contextEngineID est une valeur de "caractère générique" indéfinie. Les notifications sont livrées à un receveur de notification enregistré sans égard au contextEngineID contenu dans le message de notification.
- Le pduType indique le type de notifications que l'application souhaite recevoir (par exemple, des PDU SNMPv2 Trap ou Inform).

Une fois que le receveur de notification s'est enregistré auprès du répartiteur, les messages sont reçus en utilisant l'interface de service abstraite processPdu. Les paramètres sont :
- Le messageProcessingModel indique quel modèle de traitement de message a reçu et traité le message.
- Le securityModel est la valeur provenant du message reçu.
- Le securityName est la valeur provenant du message reçu.
- Le securityLevel est la valeur provenant du message reçu.
- Le contextEngineID est la valeur provenant du message reçu.
- Le contextName est la valeur provenant du message reçu.
- La pduVersion indique la version de la PDU dans le message reçu.
- La PDU est la valeur provenant du message reçu.
- La maxSizeResponseScopedPDU est la taille maximum admissible d'une ScopedPDU contenant une PDU de réponse (sur la base de la taille maximum de message que le générateur du message peut accepter).
- Si le message contient une PDU de classe Non confirmée, la stateReference est indéfinie et inutilisée. Autrement, la stateReference est une valeur qui fait référence aux informations en antémémoire sur la notification. Cette valeur doit être retournée au répartiteur afin de générer une réponse.

Quand une PDU de classe Non confirmée est livrée à une application de receveur de notifications, elle extrait d'abord le type d'opération SNMP, l'identifiant de demande, l'état d'erreur, l'indice d'erreur, et les liens de variable de la PDU. Après cela, le traitement dépend de la mise en œuvre particulière.

Lorsque une PDU de classe Confirmée est reçue, l'application de receveur de notifications suit la procédure suivante :

(1) Le type de PDU, l'identifiant de demande, l'état d'erreur, l'indice d'erreur, et les liens de variable sont extraits de la PDU.

(2) Une PDU de classe Response est construite en utilisant l'identifiant de demande et les liens de variable extraits, et avec l'état d'erreur et l'indice d'erreur tous deux réglés à 0.

(3) Le répartiteur est invoqué pour générer un message de réponse en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :

- Le messageProcessingModel est la valeur provenant de l'invocation de processPdu.
- Le securityModel est la valeur provenant de l'invocation de processPdu.
- Le securityName est la valeur provenant de l'invocation de processPdu.
- Le securityLevel est la valeur provenant de l'invocation de processPdu.
- Le contextEngineID est la valeur provenant de l'invocation de processPdu.
- Le contextName est la valeur provenant de l'invocation de processPdu.
- La pduVersion indique la version de la PDU à retourner.
- La PDU est le résultat généré à l'étape (2) ci-dessus.
- La maxSizeResponseScopedPDU est une valeur locale qui indique la taille maximum d'une ScopedPDU que peut accepter l'application.
- La stateReference est la valeur provenant de l'invocation de processPdu.
- Les statusInformation indiquent qu'aucune erreur ne s'est produite et qu'une réponse devrait être générée.

(4) Après cela, le traitement dépend de la mise en œuvre.

3.5 Applications de transmetteur mandataire

Une application de transmetteur mandataire s'occupe de la transmission des messages SNMP. Il y a quatre types de base de messages qu'une application de transmetteur mandataire peut devoir transmettre. Ils sont groupés selon la classe de type de PDU contenue dans un message. Les quatre types de base de messages sont :

- Ceux qui contiennent des types de PDU de classe Lecture ou Écriture (par exemple, les types de PDU Get, GetNext, GetBulk, et Set). Ils se rapportent à la demande ou à la modification des informations situées dans un contexte particulier.

- Ceux qui contiennent des types de PDU de classe Notification (par exemple, les types de PDU SNMPv2 Trap et Inform). Ils se rapportent aux notifications concernant les informations situées dans un contexte particulier.

- Ceux qui contiennent un type de PDU de classe Response. La transmission des PDU de classe Response se produit toujours par suite de la réception d'une réponse à un message transmis précédemment.

- Ceux qui contiennent des types de PDU de classe Internal (par exemple, une PDU Report). La transmission de types de PDU de classe Internal survient toujours par suite de la réception d'une PDU de classe Internal en réponse à un message transmis précédemment.

Pour le premier type, le rôle du transmetteur mandataire est de livrer une demande d'informations de gestion à un moteur SNMP qui est "plus proche" ou "en aval sur le chemin" vers le moteur SNMP qui a accès à ces informations, et de livrer la réponse contenant les informations en retour au moteur SNMP duquel la demande a été reçue. Les informations de contexte dans une demande sont utilisées pour déterminer quel moteur SNMP a accès aux informations demandées, et ceci est utilisé pour déterminer où et comment transmettre la demande.

Pour le second type, le rôle du transmetteur mandataire est de déterminer quels moteurs SNMP devraient recevoir les notifications sur les informations de gestion d'une localisation particulière. Les informations de contexte dans un message de notification déterminent la localisation à laquelle s'appliquent les informations contenues dans la notification. Ceci est utilisé pour déterminer quels moteurs SNMP devraient recevoir la notification sur ces informations.

Pour le troisième type, le rôle du transmetteur mandataire est de déterminer à quelle demande ou notification transmise précédemment (s'il en est) correspond la réponse, et de retransmettre la réponse à l'initiateur de la demande ou notification.

Pour le quatrième type, le rôle du transmetteur mandataire est de déterminer à quelle demande ou notification précédemment transmise (s'il en est) correspond la PDU de classe Internal, et de retransmettre la PDU de classe Internal à l'initiateur de la demande ou notification.

Lors de la transmission des messages, une application de transmetteur mandataire doit effectuer une traduction des informations de cible de gestion entrante en informations de cible de gestion sortante. Comment cette traduction est effectuée est spécifique de la mise en œuvre. Dans de nombreux cas, cela sera piloté par un tableau de traduction préconfiguré. Si une application de transmetteur mandataire rend gérable le contenu de ce tableau SNMP, elle DOIT utiliser le module de MIB SNMP-PROXY défini dans le présent document.

3.5.1 Transmission de la demande

Il y a deux phases à la transmission de demande. D'abord, la demande entrante doit être passée à travers l'application mandataire. Ensuite, la réponse résultante doit être repassée. Ces phases sont décrites dans les deux paragraphes qui suivent.

3.5.1.1 Traitement d'une demande entrante

Une application de transmetteur mandataire qui souhaite transmettre des messages de demande doit d'abord s'enregistrer auprès du répartiteur en utilisant l'interface de service abstraite registerContextEngineID. Le transmetteur mandataire doit enregistrer chaque contextEngineID pour lequel il souhaite transmettre des messages, ainsi que pour chaque pduType. Noter que quand la configuration d'un transmetteur mandataire est changée, les valeurs particulières de contextEngineID pour lesquelles il transmet peuvent changer. Le transmetteur mandataire devrait invoquer les interfaces de service abstraites registerContextEngineID et unregisterContextEngineID comme nécessaire pour refléter sa configuration actuelle.

Une application de transmetteur mandataire ne devrait jamais tenter d'enregistrer une valeur de contextEngineID égale au snmpEngineID du moteur SNMP auquel le transmetteur mandataire est associé.

Une fois le transmetteur mandataire enregistré pour les valeurs appropriées de contextEngineID, il peut commencer le traitement des messages. La procédure suivante est utilisée:

(1) Un message est reçu en utilisant l'interface de service abstraite processPdu. Les informations de cible de gestion entrante reçues de l'interface processPdu sont traduites en informations de cible de gestion sortante. Noter que cette traduction peut varier pour des valeurs différentes de contextEngineID et/ou contextName. La traduction devrait résulter en une seule cible de gestion.

(2) Si des informations appropriées de cible de gestion sortante ne peuvent être trouvées, le transmetteur mandataire incrémente le compteur snmpProxyDrops [RFC1907], et invoque ensuite le répartiteur en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :
- Le messageProcessingModel est la valeur provenant de l'invocation de processPdu.
- Le securityModel est la valeur provenant de l'invocation de processPdu..
- Le securityName est la valeur provenant de l'invocation de processPdu.
- Le securityLevel est la valeur provenant de l'invocation de processPdu.
- Le contextEngineID est la valeur provenant de l'invocation de processPdu.
- Le contextName est la valeur provenant de l'invocation de processPdu.
- La pduVersion est la valeur provenant de l'invocation de processPdu.
- La PDU est une valeur indéfinie.
- La maxSizeResponseScopedPDU est une valeur locale indiquant la taille maximum de ScopedPDU que l'application peut accepter.
- La stateReference est la valeur provenant de l'invocation de processPdu.
- Les statusInformation indiquent qu'une erreur s'est produite et incluent l'OID et la valeur de l'objet snmpProxyDrops.

Le traitement du message s'arrête à ce point. Autrement,

(3) Une nouvelle PDU est construite. Une valeur unique d'identifiant de demande devrait être utilisée dans la nouvelle PDU (cette valeur va permettre à un message de réponse ultérieur d'être corrélé à cette demande). Le reste de la nouvelle PDU est identique à la PDU reçue, sauf si la version SNMP entrante et la version SNMP sortante prennent en charge des versions de PDU différentes, auquel cas le transmetteur mandataire peut avoir besoin d'effectuer une traduction sur la PDU. (Une méthode pour effectuer une telle traduction est décrite dans la [RFC2576].)

(4) Le transmetteur mandataire invoque le répartiteur pour générer le message transmis, en utilisant l'interface de service abstraite sendPdu. Les paramètres sont :
- Le transportDomain est celui de la cible de gestion sortante.
- La transportAddress est celle de la cible de gestion sortante.
- Le messageProcessingModel est celui de la cible de gestion sortante.
- Le securityModel est celui de la cible de gestion sortante.
- Le securityName est celui de la cible de gestion sortante.
- Le securityLevel est celui de la cible de gestion sortante.
- Le contextEngineID est la valeur provenant de l'invocation de processPdu.
- Le contextName est la valeur provenant de l'invocation de processPdu.
- La pduVersion est la version de la PDU à envoyer.
- La PDU est la valeur construite à l'étape (3) ci-dessus.
- L'argument expectResponse indique qu'une réponse est attendue. Si l'invocation de sendPdu échoue, le transmetteur mandataire effectue les étapes décrites en (2) ci-dessus.

Autrement:

(5) Le transmetteur mandataire met en antémémoire les informations suivantes afin de confronter une réponse entrante à la

demande transmise :
- la sendPduHandle retournée de l'invocation de sendPdu,
- le request-id de la PDU reçue.
- le contextEngineID,
- le contextName,
- la stateReference,
- les informations de la cible de gestion entrante,
- les informations de gestion sortantes,
- toutes les autres informations qui sont nécessaires pour confronter une réponse entrante à la demande transmise.

Si ces informations ne peuvent pas être mises en antémémoire (peut-être à cause d'un manque de ressources) le transmetteur mandataire effectue les étapes décrites en (2) ci-dessus. Autrement :

(6) Le traitement de la demande s'arrête jusqu'à ce que soit reçue une réponse à la demande transmise, ou jusqu'à ce qu'un intervalle de temps approprié se soit écoulé. Si cet intervalle de temps expire avant la réception d'une réponse, les informations en antémémoire sur cette demande sont supprimées.

3.5.1.2 Traitement d'une réponse entrante

Un transmetteur mandataire suit la procédure ci-dessous à réception d'une réponse entrante :

(1) La réponse entrante est reçue en utilisant l'interface processResponsePdu. Le transmetteur mandataire utilise les paramètres reçus pour localiser une entrée dans son antémémoire de demandes transmises en instance. Ceci est fait en confrontant les paramètres reçus avec les valeurs en antémémoire de sendPduHandle, contextEngineID, contextName, informations de cible de gestion sortante, et le request-id contenu dans la PDU reçue (le transmetteur mandataire doit extraire le request-id à cette fin). Si on ne peut pas trouver une entrée d'antémémoire appropriée, le traitement de la réponse s'arrête. Autrement

(2) Les informations en antémémoire sont extraites, et retirées de l'antémémoire.

(3) Une nouvelle PDU de classe Response est construite, en utilisant la valeur d'identifiant de demande provenant de la demande d'origine (telle qu'extraite de l'antémémoire). Toutes les autres valeurs sont identiques à celles de la PDU de classe Response reçue, sauf si la version SNMP entrante et la version SNMP sortante prennent en charge des versions de PDU différentes, et dans ce cas le transmetteur mandataire peut avoir besoin d'effectuer une traduction sur la PDU. (Une méthode pour effectuer une telle traduction est décrite dans la [RFC2576].)

(4) Le transmetteur mandataire appelle le répartiteur en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :
- Le messageProcessingModel indique le modèle de traitement de message par lequel le message entrant originel a été traité.
- Le securityModel est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le securityName est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le securityLevel est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le contextEngineID est la valeur extraite de l'antémémoire.
- Le contextName est la valeur extraite de l'antémémoire.
- La pduVersion indique la version de la PDU à retourner.
- La PDU est la PDU de réponse (éventuellement traduite).
- La maxSizeResponseScopedPDU est une valeur locale indiquant la taille maximum d'une ScopedPDU que l'application peut accepter.
- La stateReference est la valeur extraite de l'antémémoire.
- Les statusInformation indiquent qu'aucune erreur n'est survenue et qu'un message de PDU de réponse devrait être généré.

3.5.1.3 Traitement d'une PDU de classe Interne entrante

Un transmetteur mandataire suit la procédure suivante à réception d'une PDU de classe Interne entrante :

(1) La PDU de classe Interne entrante est reçue en utilisant l'interface processResponsePdu. Le transmetteur mandataire utilise les paramètres reçus pour localiser une entrée dans son antémémoire de demandes transmises en cours. Ceci est fait en confrontant les paramètres reçus aux valeurs en antémémoire de sendPduHandle. Si une entrée d'antémémoire appropriée ne peut être trouvée, le traitement de la PDU de classe Interne est arrêté. Autrement :

(2) Les informations sont extraites de l'antémémoire, et supprimées de l'antémémoire.

(3) Si les information de la cible de gestion entrante d'origine indiquent une version SNMP qui ne prend pas en charge les PDU Report, le traitement de la PDU de classe Internal est arrêté.

(4) Le transmetteur mandataire appelle le répartiteur en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :
- Le messageProcessingModel indique le modèle de traitement de message par lequel le message entrant originel a été traité.
- Le securityModel est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le securityName est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le securityLevel est celui de la cible de gestion originelle entrante extrait de l'antémémoire.
- Le contextEngineID est la valeur extraite de l'antémémoire.
- Le contextName est la valeur extraite de l'antémémoire.
- La pduVersion indique la version de la PDU à retourner.
- La PDU n'est pas utilisée.
- La maxSizeResponseScopedPDU est une valeur locale indiquant la taille maximum de ScopedPDU que l'application peut accepter.
- La stateReference est la valeur extraite de l'antémémoire.
- Les statusInformation contiennent des valeurs spécifiques du type de PDU de classe Interne (par exemple, pour une PDU Report, les statusInformation contiennent les valeurs de contextEngineID, contextName, counter OID, et de compteur reçues dans la PDU Report entrante).

3.5.2 Transmission de notification

Un transmetteur mandataire reçoit des notifications de la même manière qu'une application de receveur de notification, en utilisant l'interface de service abstraite processPdu. La procédure suivante est utilisée lorsque une notification est reçue :

(1) Les informations de cible de gestion entrantes reçues de l'interface processPdu sont traduites en informations de cible de gestion sortante. Noter que cette traduction peut varier pour des valeurs différentes de contextEngineID et/ou contextName. La traduction peut résulter en plusieurs cibles de gestion.

(2) Si les informations de cible de gestion sortante appropriées ne peuvent pas être trouvées et si la notification était une PDU de classe Non confirmée, le traitement de la notification s'arrête. Si les informations de cible de gestion sortante appropriées ne peuvent pas être trouvées et si la notification était une PDU de classe Confirmée, le transmetteur mandataire incrémente l'objet snmpProxyDrops, et appelle le répartiteur en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :
- Le messageProcessingModel est la valeur provenant de l'appel processPdu.
- Le securityModel est la valeur provenant de l'appel processPdu.
- Le securityName est la valeur provenant de l'appel processPdu.
- Le securityLevel est la valeur provenant de l'appel processPdu.
- Le contextEngineID est la valeur provenant de l'appel processPdu.
- Le contextName est la valeur provenant de l'appel processPdu.
- La pduVersion est la valeur provenant de l'appel processPdu.
- La PDU est une valeur indéfinie et non utilisée.
- La maxSizeResponseScopedPDU est une valeur locale qui indique la taille maximum de ScopedPDU que l'application peut accepter.
- La stateReference est la valeur provenant de l'appel processPdu.
- Les statusInformation indiquent qu'une erreur s'est produite et qu'un message Report devrait être généré.

Le traitement du message s'arrête à ce point. Autrement,

(3) Le transmetteur mandataire génère une notification en utilisant les procédures décrites dans le paragraphe précédant sur les générateurs de notification, avec les exceptions suivantes :
- on utilise les valeurs de contextEngineID et contextName provenant de la notification d'origine reçue ;
- on utilise les cibles de gestion sortantes déterminées précédemment ;
- aucun mécanisme de filtrage n'est appliqué ;
- on utilise les liens de variable provenant de la notification d'origine reçue, plutôt que de restituer des liens de variable de l'instrumentation de MIB locale. En particulier, aucun contrôle d'accès n'est appliqué à ces liens de variables, ni à la valeur du lien de variable qui contient snmpTrapOID.0 ;
- si la notification originelle contient une PDU de classe Confirmée, toutes les cibles de gestion sortantes pour lesquelles la version SNMP sortante ne prend pas en charge des types de PDU qui sont à la fois de classe Notification et de classe Confirmée ne seront pas utilisées pour générer les notifications transmises ;
- si, pour toute cible de gestion sortante, la version SNMP entrante et la version SNMP sortante prennent en charge des versions différentes de PDU, le transmetteur mandataire peut avoir besoin d'effectuer une traduction sur la PDU. (Une

méthode pour effectuer une telle traduction est décrite dans la [RFC2576].)

(4) Si la notification originelle reçue contient une PDU de classe Non confirmée, le traitement de la notification est maintenant achevé. Autrement, la notification originelle reçue doit contenir une PDU de classe Confirmée, et le traitement continue.

(5) Si les notifications transmises incluaient des PDU de classe Confirmée, le traitement continue lorsque les procédures décrites dans le paragraphe sur les générateurs de notification déterminent que soit :
- aucune des notifications générées contenant des PDU de classe Confirmée n'a été acquittée avec succès dans les plus long des intervalles de temps, auquel cas le traitement de la notification originelle s'arrête, soit
- au moins une des notifications générées contenant des PDU de classe Confirmée est acquittée avec succès, auquel cas une réponse à la notification originelle reçue contenant une PDU de classe Confirmée est générée comme décrit dans les étapes suivantes.

(6) On construit une PDU de classe Response en utilisant les valeurs d'identifiant de demande et de liens de variables de la PDU originelle de classe Notification reçue, et des valeurs d'état d'erreur et d'indice d'erreur de 0.

(7) Le répartiteur est appelé en utilisant l'interface de service abstraite returnResponsePdu. Les paramètres sont :
- Le messageProcessingModel est la valeur provenant de l'appel processPdu.
- Le securityModel est la valeur provenant de l'appel processPdu.
- Le securityName est la valeur provenant de l'appel processPdu.
- Le securityLevel est la valeur provenant de l'appel processPdu.
- Le contextEngineID est la valeur provenant de l'appel processPdu.
- Le contextName est la valeur provenant de l'appel processPdu.
- La pduVersion indique la version de la PDU construite dans l'étape (6) ci-dessus.
- La PDU est la valeur construite à l'étape (6) ci-dessus.
- La maxSizeResponseScopedPDU est une valeur locale qui indique la taille maximum de ScopedPDU que l'application peut accepter.
- La stateReference est la valeur provenant de l'appel processPdu.
- Les statusInformation indiquent qu'aucune erreur ne s'est produite et qu'un message PDU de classe Response devrait être généré.

4. Structure des modules de MIB

Le présent document décrit trois modules de MIB séparés, la MIB de cible de gestion, la MIB de notification, et la MIB de mandataire. Les paragraphes qui suivent décrivent la structure de ces trois modules de MIB.

L'utilisation de ces MIB par des types d'applications particuliers est décrite plus loin dans le présent document :
- l'utilisation de la MIB de cible de gestion et de la MIB de notification dans les applications de générateur de notifications est décrite dans la section 5,
- l'utilisation de la MIB de notification pour les notifications de filtrage dans les applications de générateur de notifications est décrite à la section 6,
- l'utilisation de la MIB de cible de gestion et de la MIB de mandataire dans les applications de mandataire de transmission est décrit à la section 7.

4.1 Module de MIB de cible de gestion

Le module SNMP-TARGET-MIB contient des objets pour définir les cibles de gestion. Il consiste en deux tableaux et des déclarations de conformité.

Le premier tableau, snmpTargetAddrTable, contient des informations sur les domaines et adresses de transport. Il contient aussi un objet, snmpTargetAddrTagList, qui fournit un mécanisme pour grouper les entrées.

Le second tableau, snmpTargetParamsTable, contient des informations sur la version SNMP et sur la sécurité à utiliser lors de l'envoi de messages à des domaines et adresses de transport particuliers.

La MIB Cible de gestion est destinée à fournir un mécanisme d'utilisation générale pour spécifier l'adresse de transport, et pour spécifier les paramètres des messages SNMP générés par une entité SNMP. Elle est utilisée dans le présent document pour générer des notifications et pour la transmission par mandataire. Cependant, elle peut être utilisée à d'autres fins. Si un autre document fait usage de cette MIB, il devra spécifier comment elle est utilisée. Par exemple, la [RFC2576] utilise cette MIB pour la validation de l'adresse de source des messages SNMPv1.

4.1.1 Listes d'étiquettes

L'objet snmpTargetAddrTagList est utilisé pour grouper des entrées dans le tableau snmpTargetAddrTable. La valeur de cet objet contient une liste de valeurs d'étiquettes qui sont utilisées pour choisir les adresses de cible à utiliser pour une opération particulière.

Une valeur d'étiquette, qui peut aussi être utilisée dans des objets de MIB autres que snmpTargetAddrTagList, est une chaîne arbitraire d'octets, mais ne doit pas contenir de caractère délimiteur. Les caractères délimiteurs sont définis comme étant un des caractères suivants :
- caractère ASCII espace (0x20) ;
- caractère ASCII tabulation (0x09).
- caractère ASCII retour chariot (CR) (0x0D).
- caractère ASCII saut à la ligne (LF) (0x0A).

De plus, une valeur d'étiquette au sein d'une liste d'étiquettes ne doit pas avoir une longueur zéro. Généralement, un objet de MIB particulier peut contenir soit
- une chaîne d'octets de longueur zéro qui représente une liste vide, soit
- une seule valeur d'étiquette, auquel cas la valeur de l'objet de MIB ne doit pas contenir de caractère délimiteur, soit
- une liste de valeurs d'étiquettes, séparées par un seul caractère délimiteur.

Pour une liste de valeurs d'étiquettes, ces contraintes impliquent certaines restrictions sur la valeur d'un objet de MIB :
- il ne doit pas y avoir de caractère délimiteur en tête ou en queue,
- il ne doit pas y avoir plusieurs caractères délimiteurs adjacents.

4.1.2 Définitions

```
DEFINITIONS  SNMP-TARGET-MIB ::= DÉBUT

IMPORTE
IDENTITÉ DE MODULE, TYPE D'OBJET, snmpModules, Counter32, Integer32,  DE SNMPv2-SMI
CONVENTION TEXTUELLE, TDomain, TAddress, TimeInterval, RowStatus, StorageType, TestAndInc,  DE SNMPv2-TC
SnmpSecurityModel, SnmpMessageProcessingModel, SnmpSecurityLevel, SnmpAdminString,  DE SNMP-FRAMEWORK-
MIB
CONFORMITÉ DE MODULE, GROUPE D'OBJETS,  DE SNMPv2-CONF;

IDENTITÉ DE MODULE   snmpTargetMIB
DERNIÈRE MISE À JOUR : "200210140000Z"
ORGANISATION : "Groupe de travail IETF SNMPv3"
INFORMATIONS DE CONTACT : "Adresse de messagerie du groupe de travail : snmpv3@lists.tislabs.com
pour s'abonner :  majordomo@lists.tislabs.com
mettre dans le corps du message :  subscribe snmpv3

Coprésident :   Russ Mundy
Adresse postale : Network Associates Laboratories, 15204 Omega Drive, Suite 300, Rockville, MD 20850-4601, USA
mél :  mundy@tislabs.com
téléphone :     +1 301-947-7107

Coprésident :   David Harrington
Adresse postale :   Enterasys Networks, 35 Industrial Way, P. O. Box 5004, Rochester, New Hampshire 03866-5005, USA
mél :     dbh@enterasys.com
téléphone :     +1 603-337-2614

Co-éditeur:  David B. Levi
Adresse postale :   Nortel Networks, 3505 Kesterwood Drive, Knoxville, Tennessee 37918, USA
mél :     dlevi@nortelnetworks.com
téléphone :     +1 865 686 0432

Co-éditeur :  Paul Meyer
Adresse postale :   Secure Computing Corporation, 2675 Long Lake Road, Roseville, Minnesota 55113, USA
mél :     paul_meyer@securecomputing.com
téléphone :     +1 651 628 1592

Co-éditeur:  Bob Stewart,  retraité."
```

DESCRIPTION : "Ce module de MIB définit des objets de MIB qui fournissent des mécanismes pour configurer à distance les paramètres utilisés par une entité SNMP pour la génération de messages SNMP. Copyright (C) The Internet Society (2002). Cette version de ce module de MIB fait partie de la RFC 3413 ; voir les notices légales complètes dans la RFC elle-même. "
REVISION : "200210140000Z" -- 14 octobre 2002
DESCRIPTION : "Correction du CONSEIL D'AFFICHAGE pour les chaînes UTF-8, correction de la valeur hexadécimale des caractères LF, précision de la signification des valeurs d'étiquettes de longueur zéro, amélioration des exemples de liste d'étiquette. Publiée comme RFC 3413."
REVISION : "199808040000Z" -- 4 août 1998
DESCRIPTION : "Clarifications, publiée comme RFC 2573."
REVISION : "199707140000Z" -- 14 juillet 1997
DESCRIPTION "Version initiale, publiée comme RFC2273."
::= { snmpModules 12 }

IDENTIFIANT D'OBJET snmpTargetObjects ::= { snmpTargetMIB 1 }
IDENTIFIANT D'OBJET snmpTargetConformance ::= { snmpTargetMIB 3 }

SnmpTagValue ::= CONVENTION TEXTUELLE
CONSEILS D'AFFICHAGE : "255t"
STATUT : actuel
DESCRIPTION : "Chaîne d'octets qui contient une valeur d'étiquette. Les valeurs d'étiquette sont de préférence sous une forme lisible par l'homme. Pour faciliter l'internationalisation, ces informations sont représentées en utilisant le jeu de caractères de la norme internationale ISO/CEI 10646-1, codé comme chaîne d'octets en utilisant le schéma de codage de caractères UTF-8 décrit dans la RFC 2279. Comme des codets supplémentaires sont ajoutés de temps en temps par des amendements à la norme 10646, les mises en œuvre doivent être prêtes à rencontrer tout codet de 0x00000000 à 0x7fffffff. L'utilisation de codes de contrôle devrait être évitée, et certains codes de contrôle ne sont pas permis, comme décrit ci-dessous.
Pour les codets non directement pris en charge par le matériel ou logiciel d'interface d'utilisateur, une solution de remplacement d'entrée et d'affichage, comme l'hexadécimal, peut être fournie.
Pour les informations codées en US-ASCII à 7 bits, la représentation UTF-8 est identique au codage US-ASCII.
Noter que lorsque cette convention textuelle est utilisée pour un objet utilisé ou dont l'utilisation est envisagée 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 [RFC1905].
Un objet de ce type contient une seule valeur d'étiquette qui est utilisée pour choisir un ensemble d'entrées dans un tableau. Une valeur d'étiquette est une chaîne arbitraire d'octets, mais ne peut pas contenir un caractère délimiteur.
Les caractères délimiteurs sont définis comme étant un des suivants :
- Un caractère espace ASCII (0x20).
- Un caractère TAB ASCII (0x09).
- Un caractère ASCII retour chariot (CR) (0x0D).
- Un caractère ASCII saut à la ligne (LF) (0x0A).
Les caractères délimiteurs sont utilisés pour séparer les valeurs d'étiquettes dans une liste d'étiquettes. Un objet de ce type peut ne contenir qu'une seule valeur d'étiquette, et donc les caractères délimiteurs ne sont pas permis dans une valeur de ce type.
Noter qu'une valeur d'étiquette de longueur 0 signifie qu'aucune étiquette n'est définie. En d'autres termes, une valeur d'étiquette de longueur 0 ne correspondra jamais à quelque chose dans une liste d'étiquettes, et ne va jamais choisir une entrée d'un tableau.
Des exemples de valeurs d'étiquettes valides sont :
- 'acme'
- 'router'
- 'host'
L'utilisation d'une valeur d'étiquette pour choisir des entrées de tableau est spécifique de l'application et de la MIB."
SYNTAXE : CHAINE D'OCTETS (TAILLE (0 à 255))

SnmpTagList ::= CONVENTION TEXTUELLE
CONSEIL D'AFFICHAGE : "255t"
STATUT : actuel
DESCRIPTION "Chaîne d'octets contenant une liste de valeurs d'étiquettes. Les valeurs d'étiquettes sont de préférence sous une forme lisible par l'homme. Pour faciliter l'internationalisation, ces informations sont représentées en utilisant le jeu de caractères de la norme internationale ISO/CEI 10646-1, codé comme chaîne d'octets en utilisant le schéma de codage de caractères UTF-8 décrit dans la RFC 2279.
Comme des codets supplémentaires sont ajoutés de temps en temps à la norme 10646, les mises en œuvre doivent être prêtes à rencontrer tout codet de 0x00000000 à 0x7fffffff.

L'utilisation de codes de contrôle devrait être évitée, sauf comme décrit ci-dessous.
Pour les codets non directement pris en charge par le matériel ou logiciel d'interface d'utilisateur, un moyen d'entrée et d'affichage de remplacement tel que l'hexadécimal peut être fourni.
Pour les informations codées en US-ASCII à 7 bits, la représentation UTF-8 est identique au codage US-ASCII.
Un objet de ce type contient une liste de valeurs d'étiquettes qui sont utilisées pour choisir un ensemble d'entrées dans un tableau.
Une valeur d'étiquette est une chaîne arbitraire d'octets, mais elle ne doit pas contenir de caractère délimiteur. Les caractères délimiteurs sont définis comme étant un des suivants :
- caractère espace ASCII (0x20),
- caractère TAB ASCII (0x09).
- caractère ASCII retour chariot (CR) (0x0D).
- caractère ASCII saut à la ligne (LF) (0x0A).
Les caractères délimiteurs sont utilisés pour séparer les valeurs d'étiquettes dans une liste d'étiquettes. Un seul caractère délimiteur doit apparaître entre deux valeurs d'étiquettes. Une valeur d'étiquette ne doit pas avoir une longueur de zéro. Ces contraintes impliquent certaines restrictions sur le contenu de cet objet :
- il ne doit pas y avoir de caractère délimiteur en tête ou en queue,
- il ne doit pas y avoir plusieurs caractères délimiteurs adjacents.
Des exemples de listes d'étiquettes valides sont :
- " -- liste vide
- 'acme' -- liste d'une seule étiquette
- 'host router bridge' -- liste de plusieurs étiquettes
Noter que bien qu'une valeur d'étiquette ne doive pas avoir une longueur de zéro, une chaîne vide est pourtant valide. Cela indique une liste vide (c'est-à-dire qu'il n'y a pas de valeurs d'étiquettes dans la liste).
L'utilisation de la liste d'étiquettes pour choisir des entrées de tableau est spécifique de l'application et de la MIB. Normalement, une application va fournir une ou plusieurs valeurs d'étiquettes, et toute entrée qui contient une combinaison de ces valeurs d'étiquettes sera retenue."
SYNTAXE : CHAINE D'OCTETS (TAILLE (0 à 255))

-- Groupe snmpTargetObjects

TYPE D'OBJET snmpTargetSpinLock
SYNTAXE : TestAndIncr
MAX-ACCESS : lecture-écriture
STATUT : actuel
DESCRIPTION : "Cet objet est utilisé pour faciliter la modification des entrées du tableau dans le module SNMP-TARGET-MIB par plusieurs gestionnaires. En particulier, il est utile pour modifier la valeur de l'objet snmpTargetAddrTagList.
La procédure pour modifier l'objet snmpTargetAddrTagList est :
1. Restituer la valeur de snmpTargetSpinLock et de snmpTargetAddrTagList.
2. Générer une nouvelle valeur pour snmpTargetAddrTagList.
3. Régler la valeur de snmpTargetSpinLock à la valeur restituée, et la valeur de snmpTargetAddrTagList à la nouvelle valeur. Si l'ensemble échoue pour l'objet snmpTargetSpinLock, retourner à l'étape 1."
::= { snmpTargetObjects 1 }

TYPE D'OBJET snmpTargetAddrTable
SYNTAXE : SÉQUENCE DE SnmpTargetAddrEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Tableau des adresses de transport à utiliser dans la génération des messages SNMP."
::= { snmpTargetObjects 2 }

TYPE D'OBJET snmpTargetAddrEntry
SYNTAXE : SnmpTargetAddrEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Adresse de transport à utiliser dans la génération des opérations SNMP. Les entrées dans le tableau snmpTargetAddrTable sont créées et supprimées en utilisant l'objet snmpTargetAddrRowStatus."
INDEX : { IMPLIED snmpTargetAddrName }
::= { snmpTargetAddrTable 1 }

SnmpTargetAddrEntry ::= SEQUENCE {
snmpTargetAddrName SnmpAdminString,

```
        snmpTargetAddrTDomain       TDomain,
        snmpTargetAddrTAddress      TAddress,
        snmpTargetAddrTimeout       TimeInterval,
        snmpTargetAddrRetryCount    Integer32,
        snmpTargetAddrTagList       SnmpTagList,
        snmpTargetAddrParams        SnmpAdminString,
        snmpTargetAddrStorageType   StorageType,
        snmpTargetAddrRowStatus     RowStatus
    }
```

TYPE D'OBJET snmpTargetAddrName
SYNTAXE : SnmpAdminString (TAILLE(1 à 32))
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Identifiant arbitraire local, mais unique, associé à cette entrée snmpTargetAddrEntry."
::= { snmpTargetAddrEntry 1 }

TYPE D'OBJET snmpTargetAddrTDomain
SYNTAXE : TDomain
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet indique le type de transport de l'adresse contenue dans l'objet snmpTargetAddrTAddress."
::= { snmpTargetAddrEntry 2 }

TYPE D'OBJET snmpTargetAddrTAddress
SYNTAXE : TAddress
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet contient une adresse de transport. Le format de cette adresse dépend de la valeur de l'objet snmpTargetAddrTDomain."
::= { snmpTargetAddrEntry 3 }

TYPE D'OBJET snmpTargetAddrTimeout
SYNTAXE : TimeInterval
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet devrait refléter le temps maximum attendu d'aller retour pour communiquer avec l'adresse de transport définie par cette rangée. Lorsque un message est envoyé à cette adresse, et qu'une réponse (si il en est attendu une) n'est pas reçue dans cet intervalle, une mise en œuvre peut supposer que la réponse ne sera pas livrée.
Noter que l'intervalle de temps pendant lequel une application attend une réponse peut en fait être déduit de la valeur de cet objet. La méthode pour déduire l'intervalle de temps réel dépend de la mise en œuvre. Une de ces méthodes est de déduire le temps d'aller-retour espéré sur la base d'un certain algorithme de retransmission et du nombre de temporisations qui se sont produites. Le type de message peut aussi être considéré dans la déduction des temps d'aller-retour espérés pour les retransmissions. Par exemple, si un message est envoyé avec un niveau de sécurité qui indique l'authentification et la confidentialité, la valeur déduite peut être augmentée pour compenser le temps de traitement supplémentaire pour le traitement de l'authentification et du chiffrement."
DEFVAL : { 1500 }
::= { snmpTargetAddrEntry 4 }

TYPE D'OBJET snmpTargetAddrRetryCount
SYNTAXE : Integer32 (0..255)
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet spécifie un nombre par défaut de tentatives de réessai lorsque une réponse n'est pas reçue pour un message généré. Une application peut fournir son propre compte de réessais, et dans ce cas la valeur de cet objet est ignorée."
DEFVAL : { 3 }
::= { snmpTargetAddrEntry 5 }

TYPE D'OBJET snmpTargetAddrTagList
SYNTAXE : SnmpTagList
MAX-ACCESS : lecture-création

STATUT : actuel
DESCRIPTION : "Cet objet contient une liste de valeurs d'étiquettes qui sont utilisées pour choisir les adresses cibles pour une certaine opération."
DEFVAL : { "" }
::= { snmpTargetAddrEntry 6 }

TYPE D'OBJET snmpTargetAddrParams
SYNTAXE : SnmpAdminString (TAILLE(1..32))
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "La valeur de cet objet identifie une entrée dans le tableau snmpTargetParamsTable. L'entrée identifiée contient les paramètres SNMP à utiliser pour générer les messages à envoyer à cette adresse de transport."
::= { snmpTargetAddrEntry 7 }

TYPE D'OBJET snmpTargetAddrStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur 'permanent' n'ont pas besoin de permettre l'accès en écriture aux objets des colonnes de cette rangée."
DEFVAL : { nonVolatile }
::= { snmpTargetAddrEntry 8 }

TYPE D'OBJET snmpTargetAddrRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet à createAndGo(4) ou à createAndWait(5). Jusqu'à ce que les instances de toutes les colonnes correspondantes soient configurées de façon appropriée, la valeur de l'instance correspondante de la colonne snmpTargetAddrRowStatus est 'notReady'. En particulier, une rangée nouvellement créée ne peut par être rendue active tant que les instances correspondantes de snmpTargetAddrTDomain, snmpTargetAddrTAddress, et snmpTargetAddrParams n'ont pas été réglées.
Les objets suivants ne peuvent pas être modifiés lorsque la valeur de cet objet est active(1) :
- snmpTargetAddrTDomain
- snmpTargetAddrTAddress
Une tentative de régler ces objets lorsque la valeur de snmpTargetAddrRowStatus est active(1) va résulter en une erreur "inconsistentValue" ."
::= { snmpTargetAddrEntry 9 }

TYPE D'OBJET snmpTargetParamsTable
SYNTAXE : SÉQUENCE DE SnmpTargetParamsEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION "Tableau des informations de cible SNMP à utiliser dans la génération de messages SNMP."
::= { snmpTargetObjects 3 }

TYPE D'OBJET snmpTargetParamsEntry
SYNTAXE : SnmpTargetParamsEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Ensemble d'informations de cible SNMP. Les entrées dans snmpTargetParamsTable sont créées et supprimées en utilisant l'objet snmpTargetParamsRowStatus."
INDEX : { IMPLIED snmpTargetParamsName }
::= { snmpTargetParamsTable 1 }

```
SnmpTargetParamsEntry ::= SEQUENCE {
    snmpTargetParamsName            SnmpAdminString,
    snmpTargetParamsMPModel         SnmpMessageProcessingModel,
    snmpTargetParamsSecurityModel   SnmpSecurityModel,
    snmpTargetParamsSecurityName    SnmpAdminString,
    snmpTargetParamsSecurityLevel   SnmpSecurityLevel,
```

```
    snmpTargetParamsStorageType     StorageType,
    snmpTargetParamsRowStatus       RowStatus
  }
```

TYPE D'OBJET snmpTargetParamsName
SYNTAXE : SnmpAdminString (TAILLE(1..32))
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Identifiant local arbitraire mais univoque associé à cette snmpTargetParamsEntry."
::= { snmpTargetParamsEntry 1 }

TYPE D'OBJET snmpTargetParamsMPModel
SYNTAXE : SnmpMessageProcessingModel
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Modèle de traitement de message à utiliser lors de la génération de messages SNMP utilisant cette entrée."
:= { snmpTargetParamsEntry 2 }

TYPE D'OBJET snmpTargetParamsSecurityModel
SYNTAXE : SnmpSecurityModel (1 à 2 147 483 647)
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Modèle de sécurité à utiliser pour générer des messages SNMP en utilisant cette entrée. Une mise en œuvre peut choisir de retourner une erreur inconsistentValue si une tentative est faite de régler cette variable à une valeur pour un modèle de sécurité que la mise en œuvre ne prend pas en charge."
::= { snmpTargetParamsEntry 3 }

TYPE D'OBJET snmpTargetParamsSecurityName
SYNTAXE : SnmpAdminString
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Le securityName qui identifie le principal au nom duquel les messages SNMP seront générés en utilisant cette entrée."
::= { snmpTargetParamsEntry 4 }

TYPE D'OBJET snmpTargetParamsSecurityLevel
SYNTAXE : SnmpSecurityLevel
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Niveau de sécurité à utiliser lors de la génération de messages SNMP utilisant cette entrée."
::= { snmpTargetParamsEntry 5 }

TYPE D'OBJET snmpTargetParamsStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur de 'permanent' n'ont pas besoin de permettre l'accès en écriture à des objets des colonnes de cette rangée."
DEFVAL : { nonVolatile }
::= { snmpTargetParamsEntry 6 }

TYPE D'OBJET snmpTargetParamsRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet à createAndGo(4) ou à createAndWait(5).
Jusqu'à ce que les instances de toutes les colonnes correspondantes soient configurées de façon appropriée, la valeur de l'instance correspondante de la colonne snmpTargetParamsRowStatus est 'notReady'.
En particulier, une rangée nouvellement créée ne peut pas être rendue active tant que les snmpTargetParamsMPModel, snmpTargetParamsSecurityModel, snmpTargetParamsSecurityName, et snmpTargetParamsSecurityLevel correspondants n'ont pas tous été réglés.

Les objets suivants ne doivent pas être modifiés lorsque la valeur de cet objet est active(1):
- snmpTargetParamsMPModel
- snmpTargetParamsSecurityModel
- snmpTargetParamsSecurityName
- snmpTargetParamsSecurityLevel

Une tentative de régler ces objets alors que la valeur de snmpTargetParamsRowStatus est active(1) va résulter en une erreur inconsistentValue."
::= { snmpTargetParamsEntry 7 }

TYPE D'OBJET snmpUnavailableContexts
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre total de paquets reçus par le moteur SNMP et qui ont été éliminés parce que le contexte contenu dans le message était indisponible."
::= { snmpTargetObjects 4 }

TYPE D'OBJET snmpUnknownContexts
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre total de paquets reçus par le moteur SNMP et qui ont été éliminés parce que le contexte contenu dans le message était inconnu."
::= { snmpTargetObjects 5 }

-- Information de conformité

IDENTIFIANT D'OBJET snmpTargetCompliances ::= { snmpTargetConformance 1 }
IDENTIFIANT D'OBJET snmpTargetGroups ::= { snmpTargetConformance 2 }

-- Déclarations de conformité

CONFORMITÉ DE MODULE snmpTargetCommandResponderCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP qui incluent une application de répondeur de commande."
MODULE : Ce module
GROUPES OBLIGATOIRES : { snmpTargetCommandResponderGroup }
::= { snmpTargetCompliances 1 }

GROUPE D'OBJETS snmpTargetBasicGroup
OBJETS { snmpTargetSpinLock, snmpTargetAddrTDomain, snmpTargetAddrTAddress, snmpTargetAddrTagList, snmpTargetAddrParams, snmpTargetAddrStorageType, snmpTargetAddrRowStatus, snmpTargetParamsMPModel, snmpTargetParamsSecurityModel, snmpTargetParamsSecurityName, snmpTargetParamsSecurityLevel, snmpTargetParamsStorageType, snmpTargetParamsRowStatus }
STATUT : actuel
DESCRIPTION "Collection d'objets fournissant une configuration à distance de cibles de gestion."
::= { snmpTargetGroups 1 }

GROUPE D'OBJETS snmpTargetResponseGroup
OBJETS { snmpTargetAddrTimeout, snmpTargetAddrRetryCount }
STATUT : actuel
DESCRIPTION : "Collection d'objets fournissant une configuration à distance de cibles de gestion pour les applications qui génèrent des messages SNMP pour lesquels un message de réponse serait attendu."
::= { snmpTargetGroups 2 }

GROUPE D'OBJETS snmpTargetCommandResponderGroup
OBJETS { snmpUnavailableContexts, snmpUnknownContexts }
STATUT : actuel
DESCRIPTION : "Collection d'objets exigés pour les applications de répondeur de commandes, utilisés pour compter les conditions d'erreur."
::= { snmpTargetGroups 3 }

FIN

4.2 Module de MIB Notification

Le module SNMP-NOTIFICATION-MIB contient des objets pour la configuration à distance des paramètres utilisés par une entité SNMP pour la génération de notifications. Il consiste en trois tableaux et des déclarations de conformité. Le premier tableau, snmpNotifyTable, contient des entrées qui choisissent quelles entrées dans snmpTargetAddrTable devraient être utilisées pour générer les notifications, et le type de notification à générer.

Le second tableau, snmpNotifyFilterProfileTable, augmente un peu snmpTargetParamsTable avec un objet qui est utilisé pour associer un ensemble de filtres à une cible de gestion particulière.

Le troisième tableau, snmpNotifyFilterTable, définit des filtres qui sont utilisés pour limiter le nombre de notifications qui sont générées en utilisant des cibles de gestion particulières.

4.2.1 Définitions

DEFINITIONS SNMP-NOTIFICATION-MIB ::= DÉBUT

IMPORTE :
IDENTITÉ DE MODULE, TYPE D'OBJET, snmpModules DE SNMPv2-SMI
RowStatus, StorageType DE SNMPv2-TC
SnmpAdminString DE SNMP-FRAMEWORK-MIB
SnmpTagValue, snmpTargetParamsName DE SNMP-TARGET-MIB
CONFORMITÉ DE MODULE, GROUPE D'OBJETS DE SNMPv2-CONF;

IDENTITÉ DE MODULE snmpNotificationMIB
DERNIÈRE MISE À JOUR : "200210140000Z"
ORGANISATION : "Groupe de travail IETF SNMPv3"
INFORMATIONS DE CONTACT : "Adresse de messagerie du groupe de travail : snmpv3@lists.tislabs.com
pour s'abonner : majordomo@lists.tislabs.com
mettre dans le corps du message : subscribe snmpv3

Coprésident : Russ Mundy
Adresse postale : Network Associates Laboratories, 15204 Omega Drive, Suite 300, Rockville, MD 20850-4601, USA
mél : mundy@tislabs.com
téléphone : +1 301-947-7107

Coprésident : David Harrington
Adresse postale : Enterasys Networks, 35 Industrial Way, P. O. Box 5004, Rochester, New Hampshire 03866-5005, USA
mél : dbh@enterasys.com
téléphone : +1 603-337-2614

Co-éditeur: David B. Levi
Adresse postale : Nortel Networks, 3505 Kesterwood Drive, Knoxville, Tennessee 37918, USA
mél : dlevi@nortelnetworks.com
téléphone : +1 865 686 0432

Co-éditeur : Paul Meyer
Adresse postale : Secure Computing Corporation, 2675 Long Lake Road, Roseville, Minnesota 55113, USA
mél : paul_meyer@securecomputing.com
téléphone : +1 651 628 1592

Co-éditeur: Bob Stewart, retraité."
DESCRIPTION : "Ce module de MIB définit des objets de MIB qui fournissent des mécanismes pour configurer à distance les paramètres utilisés par une entité SNMP pour la génération des notifications. Copyright (C) The Internet Society (2002). Cette version de ce module de MIB fait partie de la RFC 3413 ; voir les notices légales complètes dans la RFC elle-même. "
REVISION : "200210140000Z" -- 14 octobre 2002
DESCRIPTION "Clarifications, publiées comme RFC 3413."
REVISION "199808040000Z" -- 4 août 1998
DESCRIPTION "Clarifications, publiée comme RFC 2573."

```
        REVISION    "199707140000Z"          -- 14 juillet 1997
        DESCRIPTION "Version initiale, publiée comme RFC2273."
          ::= { snmpModules 13 }

        IDENTIFIANT D'OBJET   snmpNotifyObjects          ::= { snmpNotificationMIB 1 }
        IDENTIFIANT D'OBJET   snmpNotifyConformance  ::= { snmpNotificationMIB 3 }
```

-- Groupe snmpNotifyObjects

```
        TYPE D'OBJET  snmpNotifyTable
        SYNTAXE : SÉQUENCE DE SnmpNotifyEntry
        MAX-ACCESS : non accessible
        STATUT : actuel
        DESCRIPTION : "Ce tableau est utilisé pour choisir les cibles de gestion qui devraient recevoir des notifications, ainsi que le
                type de notification qui devrait être envoyé à chaque cible de gestion choisie."
          ::= { snmpNotifyObjects 1 }

        TYPE D'OBJET  snmpNotifyEntry
        SYNTAXE : SnmpNotifyEntry
        MAX-ACCESS : non accessible
        STATUT : actuel
        DESCRIPTION : "Une entrée de ce tableau choisit un ensemble de cibles de gestion qui devraient recevoir des notifications,
                ainsi que le type de notification qui devrait être envoyé à chaque cible de gestion choisie. Les entrées dans
                snmpNotifyTable sont créées et supprimées en utilisant l'objet snmpNotifyRowStatus."
        INDEX : { IMPLIED snmpNotifyName }
          ::= { snmpNotifyTable 1 }

        SnmpNotifyEntry ::= SEQUENCE {
            snmpNotifyName          SnmpAdminString,
            snmpNotifyTag           SnmpTagValue,
            snmpNotifyType          ENTIER,
            snmpNotifyStorageType   StorageType,
            snmpNotifyRowStatus     RowStatus
          }

        TYPE D'OBJET  snmpNotifyName
        SYNTAXE : SnmpAdminString (TAILLE(1..32))
        MAX-ACCESS : non accessible
        STATUT : actuel
        DESCRIPTION : "Identifiant arbitraire univoque local associé à cette snmpNotifyEntry."
          ::= { snmpNotifyEntry 1 }

        TYPE D'OBJET  snmpNotifyTag
        SYNTAXE : SnmpTagValue
        MAX-ACCESS : lecture-création
        STATUT : actuel
        DESCRIPTION : "Cet objet contient une seule valeur d'étiquette qui est utilisée pour choisir des entrées dans
                snmpTargetAddrTable. Toute entrée dans snmpTargetAddrTable qui contient une valeur d'étiquette égale à la valeur
                d'une instance de cet objet est choisie. Si cet objet contient une valeur de longueur zéro, aucune entrée n'est choisie."
        DEFVAL : { "" }
          ::= { snmpNotifyEntry 2 }

        TYPE D'OBJET  snmpNotifyType
        SYNTAXE : ENTIER { trap(1), inform(2) }
        MAX-ACCESS : lecture-création
        STATUT : actuel
        DESCRIPTION : "Cet objet détermine le type de notification à générer pour les entrées dans snmpTargetAddrTable choisies
                par l'instance de snmpNotifyTag correspondante. Cette valeur n'est utilisée que pour générer des notifications, et est
                ignorée lors de l'utilisation de snmpTargetAddrTable à d'autres fins. Si la valeur de cet objet est trap(1), alors tout
                message généré pour les rangées choisies contiendra une PDU de classe Non confirmée. Si la valeur de cet objet est
                inform(2), alors tout message généré pour les rangées choisies contiendra une PDU de classe Confirmée. Noter que
                si une entité SNMP prend en charge la génération de PDU de classe Non confirmée (et pas de PDU de classe
```

Confirmée) alors cet objet peut être en lecture seule."
DEFVAL : { trap }
::= { snmpNotifyEntry 3 }

TYPE D'OBJET snmpNotifyStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur 'permanent' n'ont pas besoin de permettre d'accès en écriture aux objets des colonnes de cette rangée."
DEFVAL : { nonVolatile }
::= { snmpNotifyEntry 4 }

TYPE D'OBJET snmpNotifyRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet à createAndGo(4) ou à createAndWait(5)."
::= { snmpNotifyEntry 5 }

TYPE D'OBJET snmpNotifyFilterProfileTable
SYNTAXE : SÉQUENCE DE SnmpNotifyFilterProfileEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Ce tableau est utilisé pour associer un profil de filtre de notification à un ensemble particulier de paramètres cibles."
::= { snmpNotifyObjects 2 }

TYPE D'OBJET snmpNotifyFilterProfileEntry
SYNTAXE : SnmpNotifyFilterProfileEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Une entrée dans ce tableau indique le nom du profil de filtre à utiliser lors de la génération de notifications en utilisant l'entrée correspondante dans snmpTargetParamsTable. Les entrées dans snmpNotifyFilterProfileTable sont créées et supprimées en utilisant l'objet snmpNotifyFilterProfileRowStatus."
INDEX : { IMPLIED snmpTargetParamsName }
::= { snmpNotifyFilterProfileTable 1 }

SnmpNotifyFilterProfileEntry ::= SEQUENCE {
snmpNotifyFilterProfileName SnmpAdminString,
snmpNotifyFilterProfileStorType StorageType,
snmpNotifyFilterProfileRowStatus RowStatus
}

TYPE D'OBJET snmpNotifyFilterProfileName
SYNTAXE : SnmpAdminString (TAILLE(1..32))
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Nom du profil de filtre à utiliser pour générer des notifications en utilisant l'entrée correspondante dans snmpTargetAddrTable."
::= { snmpNotifyFilterProfileEntry 1 }

TYPE D'OBJET snmpNotifyFilterProfileStorType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur de 'permanent' n'ont pas besoin de permettre l'accès en écriture aux objets des colonnes de cette rangée."
DEFVAL : { nonVolatile }
::= { snmpNotifyFilterProfileEntry 2 }

TYPE D'OBJET snmpNotifyFilterProfileRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet à createAndGo(4) ou à createAndWait(5). Tant que les instances de toutes les colonnes correspondantes ne sont pas configurées de façon appropriée, la valeur de l'instance correspondante de la colonne snmpNotifyFilterProfileRowStatus est 'notReady'. En particulier, une rangée qui vient d'être créée ne peut pas être rendue active tant que l'instance correspondante de snmpNotifyFilterProfileName n'a pas été établie."
::= { snmpNotifyFilterProfileEntry 3 }

TYPE D'OBJET snmpNotifyFilterTable
SYNTAXE : SÉQUENCE DE SnmpNotifyFilterEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Tableau des profils de filtre. Les profils de filtre sont utilisés pour déterminer si des cibles de gestion particulières devrait recevoir des notifications particulières. Lorsque une notification est générée, elle doit être comparée aux filtres associés à chaque cible de gestion qui est configurée à recevoir des notifications, afin de déterminer si elle peut être envoyée à chacune de ces cibles de gestion. Une discussion plus complète du filtrage de notifications se trouve à la section 6."
::= { snmpNotifyObjects 3 }

TYPE D'OBJET snmpNotifyFilterEntry
SYNTAXE : SnmpNotifyFilterEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Élément d'un profil de filtre. Les entrées dans snmpNotifyFilterTable sont créées et supprimées en utilisant l'objet snmpNotifyFilterRowStatus."
INDEX : { snmpNotifyFilterProfileName, IMPLIED snmpNotifyFilterSubtree }
::= { snmpNotifyFilterTable 1 }

```
SnmpNotifyFilterEntry ::= SEQUENCE {
    snmpNotifyFilterSubtree         IDENTIFIANT D'OBJET,
    snmpNotifyFilterMask            CHAINE D'OCTETS,
    snmpNotifyFilterType            ENTIER,
    snmpNotifyFilterStorageType     StorageType,
    snmpNotifyFilterRowStatus       RowStatus
  }
```

TYPE D'OBJET snmpNotifyFilterSubtree
SYNTAXE : IDENTIFIANT D'OBJET
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Sous arborescence de MIB qui, lorsque combinée avec les instances correspondantes de snmpNotifyFilterMask, définit une famille de sous arborescences qui sont incluses ou exclues du profil de filtre."
::= { snmpNotifyFilterEntry 1 }

TYPE D'OBJET snmpNotifyFilterMask
SYNTAXE : CHAINE D'OCTETS (TAILLE(0..16))
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Gabarit binaire qui, en combinaison avec l'instance correspondante de snmpNotifyFilterSubtree, définit une famille de sous arborescences qui sont incluses ou exclues du profil de filtre. Chaque bit de ce gabarit binaire correspond à un sous identifiant de snmpNotifyFilterSubtree, avec le bit de poids fort du ième octet de cette valeur de chaîne d'octets (étendue si nécessaire, voir ci-dessous) correspondant au (8*i - 7)ème sous identifiant, et le bit de moindre poids du ième octet de cette chaîne d'octets correspondant au (8*i)ème sous identifiant, où i est dans la gamme de 1 à 16. Chaque bit de ce gabarit binaire spécifie si les sous identifiants correspondants doivent ou non correspondre lorsque on détermine si un IDENTIFIANT D'OBJET correspond à cette famille de sous arborescences de filtres ; un '1' indique qu'une correspondance exacte doit se produire ; un '0' indique un 'caractère générique', c'est-à-dire que toute valeur de sous identifiant correspond. Donc, l'identifiant d'objet X d'une instance d'objet est contenue dans une famille de sous arborescences de filtres si, pour chaque sous identifiant de la valeur de snmpNotifyFilterSubtree, soit :

le ième bit de snmpNotifyFilterMask est 0, soit
le ième sous identifiant de X est égal à au ième sous identifiant de la valeur de snmpNotifyFilterSubtree.
Si la valeur de ce gabarit binaire est longue de M bits et si il y a plus de M sous identifiants dans l'instance correspondante de snmpNotifyFilterSubtree, le gabarit binaire est alors étendu avec des uns pour avoir la longueur requise.
Noter que lorsque la valeur de cet objet est la chaîne de longueur zéro, cette règle d'extension résulte en l'utilisation d'un gabarit entièrement constitué de uns (c'est-à-dire, sans 'caractère générique') et la famille de sous arborescences de filtres est la sous arborescence identifiée de façon univoque par l'instance correspondante de snmpNotifyFilterSubtree."

DEFVAL : { "H }
::= { snmpNotifyFilterEntry 2 }

TYPE D'OBJET snmpNotifyFilterType
SYNTAXE : ENTIER { included(1), excluded(2) }
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet indique si la famille de sous arborescences de filtre définie par cette entrée est incluse ou exclue d'un filtre. Une discussion plus détaillée de l'utilisation de cet objet se trouve à la section 6."
DEFVAL : { included }
::= { snmpNotifyFilterEntry 3 }

TYPE D'OBJET snmpNotifyFilterStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur de 'permanent' n'ont pas besoin de permettre l'accès en écriture à des objets des colonnes de la rangée."
DEFVAL : { nonVolatile }
::= { snmpNotifyFilterEntry 4 }

TYPE D'OBJET snmpNotifyFilterRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet soit à createAndGo(4) soit à createAndWait(5)."
::= { snmpNotifyFilterEntry 5 }

-- Informations de conformité

IDENTIFIANT D'OBJET snmpNotifyCompliances ::= { snmpNotifyConformance 1 }
IDENTIFIANT D'OBJET snmpNotifyGroups ::= { snmpNotifyConformance 2 }

-- Déclarations de conformité

CONFORMITÉ DE MODULE snmpNotifyBasicCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP minimales qui mettent en œuvre seulement les notifications SNMP de classe Non confirmée et les opérations de lecture création sur seulement le snmpTargetAddrTable."

GROUPES OBLIGATOIRES DU MODULE SNMP-TARGET-MIB { snmpTargetBasicGroup }

OBJET snmpTargetParamsMPModel
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé."

OBJET snmpTargetParamsSecurityModelMIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé."

OBJET snmpTargetParamsSecurityName
MIN-ACCESS : lecture seule

DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé."

OBJET snmpTargetParamsSecurityLevel
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé."

OBJET snmpTargetParamsStorageType
SYNTAX ENTIER { readOnly(5) }
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé. La prise en charge des valeurs other(1), volatile(2), nonVolatile(3), et permanent(4) n'est pas exigé."

OBJET snmpTargetParamsRowStatus
SYNTAX ENTIER { active(1) }
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification à snmpTargetParamsTable n'est pas exigé. La prise en charge des valeurs notInService(2), notReady(3), createAndGo(4), createAndWait(5), et destroy(6) n'est pas exigé."

MODULE -- Ce module
GROUPES OBLIGATOIRES { snmpNotifyGroup }

OBJET snmpNotifyTag
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé."

OBJET snmpNotifyType
SYNTAX ENTIER { trap(1) }
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé. La prise en charge de la valeur notify(2) n'est pas exigée."

OBJET snmpNotifyStorageType
SYNTAX ENTIER { readOnly(5) }
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification n'est pas exigé. La prise en charge des valeurs other(1), volatile(2), nonVolatile(3), et permanent(4) n'est pas exigée."

OBJET snmpNotifyRowStatus
SYNTAX ENTIER { active(1) }
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en création/suppression/modification à snmpNotifyTable n'est pas exigé. La prise en charge des valeurs notInService(2), notReady(3), createAndGo(4), createAndWait(5), et destroy(6) n'est pas exigée."
::= { snmpNotifyCompliances 1 }

CONFORMITÉ DE MODULE snmpNotifyBasicFiltersCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP qui mettent en œuvre les notifications SNMP de classe Non confirmée avec des opérations de filtrage, et de lecture-création sur tous les tableaux concernés."

MODULE SNMP-TARGET-MIB
GROUPES OBLIGATOIRES { snmpTargetBasicGroup }

MODULE -- Ce module
GROUPES OBLIGATOIRES { snmpNotifyGroup, snmpNotifyFilterGroup }
::= { snmpNotifyCompliances 2 }

CONFORMITÉ DE MODULE snmpNotifyFullCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP qui mettent en œuvre seulement les notifications SNMP de classe Confirmée, ou les notifications SNMP des deux classes Non confirmée et Confirmée, plus les opérations de filtrage et de lecture-création sur les tableaux concernés."

MODULE SNMP-TARGET-MIB
GROUPES OBLIGATOIRES { snmpTargetBasicGroup, snmpTargetResponseGroup }

MODULE -- Ce module
GROUPES OBLIGATOIRES { snmpNotifyGroup, snmpNotifyFilterGroup }
::= { snmpNotifyCompliances 3 }

GROUPE D'OBJETS snmpNotifyGroup
OBJETS { snmpNotifyTag, snmpNotifyType, snmpNotifyStorageType, snmpNotifyRowStatus }
STATUT : actuel
DESCRIPTION : "Collection d'objets pour choisir quelles cibles de gestion sont utilisées pour générer les notifications, et le type de notification à générer pour chaque cible de gestion choisie."
::= { snmpNotifyGroups 1 }

GROUPE D'OBJETS snmpNotifyFilterGroup
OBJETS { snmpNotifyFilterProfileName, snmpNotifyFilterProfileStorType, snmpNotifyFilterProfileRowStatus, snmpNotifyFilterMask, snmpNotifyFilterType, snmpNotifyFilterStorageType, snmpNotifyFilterRowStatus }
STATUT : actuel
DESCRIPTION : "Collection d'objets qui fournit la configuration à distance des filtres de notification."
::= { snmpNotifyGroups 2 }

FIN

4.3 Module de MIB Proxy

Le module SNMP-PROXY-MIB, qui définit les objets de MIB qui fournissent les mécanismes pour configurer à distance les paramètres utilisés par une entité SNMP pour les opérations de transmission par mandataire, contient un seul tableau. Ce tableau, snmpProxyTable, est utilisé pour définir les traductions entre les cibles de gestion à utiliser lors de la transmission des messages.

4.3.1 Définitions

DEFINITIONS DE SNMP-PROXY-MIB ::= DÉBUT

IMPORTE :
IDENTITÉ DE MODULE, TYPE D'OBJET, snmpModules DE SNMPv2-SMI
RowStatus, StorageType DE SNMPv2-TC
SnmpEngineID, SnmpAdminString DE SNMP-FRAMEWORK-MIB
SnmpTagValue DE SNMP-TARGET-MIB
CONFORMITÉ DE MODULE, GROUPE D'OBJETS DE SNMPv2-CONF.

IDENTITÉ DE MODULE snmpProxyMIB
DERNIÈRE MISE À JOUR "200210140000Z"
ORGANISATION "Groupe de travail IETF SNMPv3"
INFORMATIONS DE CONTACT : "Adresse de messagerie du groupe de travail : snmpv3@lists.tislabs.com
pour s'abonner : majordomo@lists.tislabs.com
mettre dans le corps du message : subscribe snmpv3

Coprésident : Russ Mundy
Adresse postale : Network Associates Laboratories, 15204 Omega Drive, Suite 300, Rockville, MD 20850-4601, USA
mél : mundy@tislabs.com
téléphone : +1 301-947-7107

Coprésident : David Harrington
Adresse postale : Enterasys Networks, 35 Industrial Way, P. O. Box 5004, Rochester, New Hampshire 03866-5005, USA
mél : dbh@enterasys.com
téléphone : +1 603-337-2614

Co-éditeur: David B. Levi
Adresse postale : Nortel Networks, 3505 Kesterwood Drive, Knoxville, Tennessee 37918, USA
mél : dlevi@nortelnetworks.com
téléphone : +1 865 686 0432

Co-éditeur : Paul Meyer
Adresse postale : Secure Computing Corporation, 2675 Long Lake Road, Roseville, Minnesota 55113, USA
mél : paul_meyer@securecomputing.com
téléphone : +1 651 628 1592

Co-éditeur: Bob Stewart, retraité."
DESCRIPTION : "Ce module de MIB définit les objets de MIB fournissant les mécanismes pour configurer à distance les paramètres utilisés par une application de transmission mandataire. Copyright (C) The Internet Society (2002). Cette version de module de MIB fait partie de la RFC 3413 ; voir les notices légales complètes dans la RFC elle-même. "
REVISION : "200210140000Z" -- 14 octobre 2002
DESCRIPTION "Clarifications, publiée comme RFC 3413."
REVISION "199808040000Z" -- 4 août 1998
DESCRIPTION "Clarifications, publiée comme RFC 2573."
REVISION "199707140000Z" -- 14 juillet 1997
DESCRIPTION "Version initiale, publiée comme RFC2273."
::= { snmpModules 14 }

IDENTIFIANT D'OBJET snmpProxyObjects ::= { snmpProxyMIB 1 }
IDENTIFIANT D'OBJET snmpProxyConformance ::= { snmpProxyMIB 3 }

-- Groupe snmpProxyObjects

TYPE D'OBJET snmpProxyTable
SYNTAXE : SÉQUENCE DE SnmpProxyEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Tableau des paramètres de traduction utilisés par les applications de transmetteur mandataire pour transmettre les messages SNMP."
::= { snmpProxyObjects 2 }

TYPE D'OBJET snmpProxyEntry
SYNTAXE : SnmpProxyEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Ensemble de paramètres de traduction utilisés par une application de transmetteur mandataire pour transmettre les messages SNMP. Les entrées du snmpProxyTable sont créées et supprimées en utilisant l'objet snmpProxyRowStatus."
INDEX : { IMPLIED snmpProxyName }
::= { snmpProxyTable 1 }

```
SnmpProxyEntry ::= SEQUENCE {
    snmpProxyName               SnmpAdminString,
    snmpProxyType               ENTIER,
    snmpProxyContextEngineID    SnmpEngineID,
    snmpProxyContextName        SnmpAdminString,
    snmpProxyTargetParamsIn     SnmpAdminString,
    snmpProxySingleTargetOut    SnmpAdminString,
    snmpProxyMultipleTargetOut  SnmpTagValue,
    snmpProxyStorageType        StorageType,
    snmpProxyRowStatus          RowStatus
  }
```

TYPE D'OBJET snmpProxyName
SYNTAXE : SnmpAdminString (TAILLE(1..32))
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Identifiant local arbitraire, mais univoque associé à cette snmpProxyEntry."
::= { snmpProxyEntry 1 }

TYPE D'OBJET snmpProxyType
SYNTAXE : ENTIER { read(1), write(2), trap(3), inform(4) }

MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type du message qui peut être transmis en utilisant les paramètres de traduction définis par cette entrée."
::= { snmpProxyEntry 2 }

TYPE D'OBJET snmpProxyContextEngineID
SYNTAXE : SnmpEngineID
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Le contextEngineID contenu dans les messages qui peuvent être transmis en utilisant les paramètres de traduction définis par cette entrée."
::= { snmpProxyEntry 3 }

TYPE D'OBJET snmpProxyContextName
SYNTAXE : SnmpAdminString
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Le contextName contenu dans les messages qui peuvent être transmis en utilisant les paramètres de traduction définis par cette entrée. Cet objet est facultatif, et si il n'est pas pris en charge, le contextName contenu dans un message est ignoré lors du choix d'une entrée dans le snmpProxyTable."
::= { snmpProxyEntry 4 }

TYPE D'OBJET snmpProxyTargetParamsIn
SYNTAXE : SnmpAdminString
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet choisit une entrée dans le snmpTargetParamsTable. L'entrée choisie est utilisée pour déterminer quelle rangée du snmpProxyTable utiliser pour transmettre les messages reçus."
::= { snmpProxyEntry 5 }

TYPE D'OBJET snmpProxySingleTargetOut
SYNTAXE : SnmpAdminString
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet choisit une cible de gestion définie dans le snmpTargetAddrTable (dans la SNMP-TARGET-MIB). La cible choisie est définie par une entrée dans le snmpTargetAddrTable dont la valeur d'indice (snmpTargetAddrName) est égale à cet objet. Cet objet n'est utilisé que lorsque le choix d'une seule cible est requis (c'est-à-dire lors de la transmission d'une demande entrante de lecture ou d'écriture)."
::= { snmpProxyEntry 6 }

TYPE D'OBJET snmpProxyMultipleTargetOut
SYNTAXE : SnmpTagValue
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Cet objet choisit un ensemble de cibles de gestion définies dans le snmpTargetAddrTable (dans SNMP-TARGET-MIB). Cet objet n'est utilisé que lorsque le choix de plusieurs cibles est exigé (c'est-à-dire lors de la transmission d'une notification entrante)."
::= { snmpProxyEntry 7 }

TYPE D'OBJET snmpProxyStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation de cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur de 'permanent' n'ont pas besoin de permettre l'accès en écriture à un objet d'une colonne de cette rangée ."
DEFVAL : { nonVolatile }
::= { snmpProxyEntry 8 }

TYPE D'OBJET snmpProxyRowStatus
SYNTAXE : RowStatus
MAX-ACCESS : lecture-création
STATUT : actuel

DESCRIPTION : "Statut de cette rangée conceptuelle. Pour créer une rangée dans ce tableau, un gestionnaire doit régler cet objet soit à createAndGo(4) soit à createAndWait(5). Les objets suivants ne peuvent pas être modifiés lorsque la valeur de cet objet est active(1) :
- snmpProxyType
- snmpProxyContextEngineID
- snmpProxyContextName
- snmpProxyTargetParamsIn
- snmpProxySingleTargetOut
- snmpProxyMultipleTargetOut"
::= { snmpProxyEntry 9 }

-- Informations de conformité

IDENTIFIANT D'OBJET snmpProxyCompliances ::= { snmpProxyConformance 1 }
IDENTIFIANT D'OBJET snmpProxyGroups ::= { snmpProxyConformance 2 }

-- Déclarations de conformité

CONFORMITÉ DE MODULE snmpProxyCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP qui comportent une application de transmetteur mandataire."
GROUPES OBLIGATOIRES DU MODULE SNMP-TARGET-MIB { snmpTargetBasicGroup, snmpTargetResponseGroup }
MODULE -- Ce module
GROUPES OBLIGATOIRES { snmpProxyGroup }
::= { snmpProxyCompliances 1 }

GROUPE D'OBJETS snmpProxyGroup
OBJETS { snmpProxyType, snmpProxyContextEngineID, snmpProxyContextName, snmpProxyTargetParamsIn, snmpProxySingleTargetOut, snmpProxyMultipleTargetOut, snmpProxyStorageType, snmpProxyRowStatus }
STATUT : actuel
DESCRIPTION : "Collection d'objets qui fournissent la configuration à distance des paramètres de traduction de cible de gestion à utiliser par les applications de transmetteur mandataire."
::= { snmpProxyGroups 3 }

FIN

5. Identification des cibles de gestion chez les générateurs de notification

Cette section décrit les mécanismes utilisés par une application de générateur de notifications lorsque elle utilise le module de MIB décrit dans le présent document pour déterminer l'ensemble de cibles de gestion à utiliser pour générer une notification.

Un générateur de notification utilise toutes les entrées actives dans snmpNotifyTable pour trouver les cibles de gestion à utiliser pour générer les notifications. Chaque entrée active dans ce tableau choisit zéro, une ou plusieurs entrées dans le snmpTargetAddrTable. Lorsque une notification est générée, elle est envoyée à toutes les cibles spécifiées par les entrées de snmpTargetAddrTable choisies (sous réserve de l'application de contrôle d'accès et de filtrage de notification).

Toute entrée dans le snmpTargetAddrTable dont l'objet snmpTargetAddrTagList contient une valeur d'étiquette qui est égale à une valeur de snmpNotifyTag est choisie par la snmpNotifyEntry qui contient cette instance de snmpNotifyTag. Noter qu'une snmpTargetAddrEntry particulière peut être choisie par plusieurs entrées dans le snmpNotifyTable, résultant en la génération de plusieurs notifications en utilisant cette snmpTargetAddrEntry (cela permet, par exemple, d'envoyer à la fois des filtres et des informations à la même cible).

Chaque snmpTargetAddrEntry contient un pointeur sur le snmpTargetParamsTable (snmpTargetAddrParams). Ce pointeur choisit un ensemble de paramètres SNMP à utiliser pour générer les notifications. Si l'entrée choisie dans le tableau snmpTargetParamsTable n'existe pas, la cible de gestion n'est pas utilisée pour générer des notifications.

La décision qu'une notification devrait contenir une PDU de classe Non confirmée ou une PDU de classe Confirmée est déterminée par la valeur de l'objet snmpNotifyType. Si la valeur de cet objet est trap(1), la notification devrait contenir une PDU de classe Non confirmée.

Si la valeur de cet objet est inform(2), la notification devrait alors contenir une PDU de classe Confirmée, et la durée de temporisation et le nombre d'essais de la notification sont les valeurs de snmpTargetAddrTimeout et de snmpTargetAddrRetryCount. Noter que l'exception à ces règles est lorsque l'objet snmpTargetParamsMPModel indique une version SNMP qui prend en charge une version de PDU différente. Dans ce cas, la notification peut être envoyée en utilisant un type de PDU différent. (La [RFC2576] définit le type de PDU dans le cas où la version SNMP sortante est SNMPv1).

6. Filtrage de notification

Cette section décrit les mécanismes utilisés par une application de générateur de notifications lorsque on utilise le module de MIB décrit dans le présent document pour filtrer la génération de notifications.

Un générateur de la notification utilise le snmpNotifyFilterTable pour filtrer les notifications. Un profil de filtre de notification peut être associé à une entrée particulière dans le snmpTargetParamsTable. Le profil de filtre associé est identifié par une entrée dans le snmpNotifyFilterProfileTable dont l'indice est égal à l'indice de l'entrée dans le snmpTargetParamsTable. Si aucune entrée telle n'existe dans le snmpNotifyFilterProfileTable, aucun filtrage n'est effectué pour cette cible de gestion.

Si une telle entrée existe, la valeur de snmpNotifyFilterProfileName de l'entrée est comparée à la portion correspondante de l'indice de toutes les entrées actives dans le snmpNotifyFilterTable. Toutes les entrées pour lesquelles cette comparaison résulte en un correspondance exacte sont utilisées pour filtrer une notification générée en utilisant la snmpTargetParamsEntry associée. Si il n'existe pas de telles entrées, aucun filtrage n'est effectué, et une notification peut être envoyée à la cible de gestion.

Autrement, si il existe des entrées correspondantes, une notification peut être envoyée si l'IDENTIFIANT D'OBJET de TYPE DE NOTIFICATION de la notification (c'est la valeur de l'élément du lien de variable dont le nom est snmpTrapOID.0, c'est-à-dire, le second lien de variable) est spécifiquement inclus, et si aucune des instances d'objet à inclure dans les liens de variable de la notification n'est spécifiquement exclue par les entrées qui correspondent.

Chaque ensemble d'entrées de snmpNotifyFilterTable est divisé en deux collections de sous arborescences de filtres : les sous arborescences de filtres incluses, et les sous arborescences de filtre exclues. L'objet snmpNotifyFilterType définit la collection à laquelle appartient chaque entrée correspondante.

Pour déterminer si un nom ou instance d'objet particulier de notification est exclu par l'ensemble des entrées correspondantes, on compare le nom de la notification ou l'IDENTIFIANT D'OBJET de l'instance d'objet à chacune des entrées correspondantes. Pour un nom de notification, si aucun ne correspond, le nom de notification est alors considéré comme exclu, et la notification ne devrait pas être envoyée à cette cible de gestion. Pour une instance d'objet, si aucune ne correspond, l'instance d'objet est considérée incluse, et la notification peut être envoyée à cette cible de gestion. Si une ou plusieurs correspondent, le nom de notification ou l'instance d'objet est inclus ou exclu, selon la valeur du snmpNotifyFilterType dans l'entrée dont la valeur de snmpNotifyFilterSubtree a le plus de sous identifiants. Si plusieurs entrées correspondent et ont le même nombre de sous identifiants, alors la valeur du snmpNotifyFilterType, dans l'entrée parmi celle qui correspondent, et dont l'instance est lexicographiquement la plus grande, détermine l'inclusion ou l'exclusion.

Un nom de notification ou l'IDENTIFIANT D'OBJET X de l'instance d'objet correspond à une entrée dans snmpNotifyFilterTable lorsque le nombre de sous identifiants dans X est au moins autant que dans la valeur de snmpNotifyFilterSubtree pour l'entrée, et chaque sous identifiant dans la valeur de snmpNotifyFilterSubtree correspond à son sous identifiant correspondant dans X. Deux sous identifiants correspondent si le bit correspondant de snmpNotifyFilterMask est zéro (la valeur de 'caractère générique') ou si les deux sous identifiants sont égaux.

7. Traduction de cible de gestion pour les applications de mandataire de transmission

Cette section décrit les mécanismes utilisés par une application de transmetteur mandataire lorsque on utilise le module de MIB décrit dans le présent document pour traduire les informations entrantes de cible de gestion en informations sortantes de cible de gestion pour les besoins de la transmission de messages. Il y a en fait deux mécanismes qu'un transmetteur mandataire peut utiliser, un pour transmettre les messages de demande, et un pour transmettre les messages de notification.

7.1 Traduction de cible de gestion pour la transmission de demande

Lors de la transmission de messages de demande, le transmetteur mandataire va choisie une seule entrée dans le snmpProxyTable. Pour choisir cette entrée, il va effectuer les comparaisons suivantes :

- Le snmpProxyType doit être read(1) si la demande est une PDU de classe Lecture. Le snmpProxyType doit être write(2) si

la demande est une PDU de classe Écriture.
- Le contextEngineID doit être égal à l'objet snmpProxyContextEngineID.
- Si l'objet snmpProxyContextName est pris en charge, il doit être égal au contextName.
- L'objet snmpProxyTargetParamsIn identifie une entrée dans le snmpTargetParamsTable. Le messageProcessingModel, le securityModel, le securityName, et le securityLevel doivent correspondre aux valeurs de snmpTargetParamsMPModel, de snmpTargetParamsSecurityModel, de snmpTargetParamsSecurityName, et de snmpTargetParamsSecurityLevel de l'entrée identifiée dans le snmpTargetParamsTable.

Il peut y avoir plusieurs entrées dans le snmpProxyTable pour lesquelles ces comparaisons réussissent. L'entrée dont le snmpProxyName a la plus petite valeur lexicographique et pour laquelle la comparaison a réussi sera choisie par le transmetteur mandataire.

Les informations sortantes de cible de gestion sont identifiées par la valeur de l'objet snmpProxySingleTargetOut de l'entrée choisie. Cet objet identifie une entrée dans le snmpTargetAddrTable. L'entrée identifiée dans le snmpTargetAddrTable contient aussi une référence au snmpTargetParamsTable (snmpTargetAddrParams). Si l'entrée identifiée dans le snmpTargetAddrTable n'existe pas, ou si l'entrée identifiée dans le snmpTargetParamsTable n'existe pas, alors cette snmpProxyEntry n'identifie pas des informations de transmission valides, et le transmetteur mandataire devrait tenter d'identifier une autre rangée.

Si il n'y a pas d'entrée dans le snmpProxyTable pour laquelle toutes les conditions ci-dessus peuvent être satisfaites, il n'y a alors pas d'informations de transmission appropriées, et le transmetteur mandataire devrait prendre les dispositions appropriées.

Autrement, le snmpTargetAddrTDomain, le snmpTargetAddrTAddress, le snmpTargetAddrTimeout, et le snmpTargetRetryCount de la snmpTargetAddrEntry identifiée, et le snmpTargetParamsMPModel, le snmpTargetParamsSecurityModel, le snmpTargetParamsSecurityName, et le snmpTargetParamsSecurityLevel de la snmpTargetParamsEntry identifiée sont utilisées comme cible de gestion de destination.

7.2 Traduction de cible de gestion pour la transmission de notification

Lors de la transmission de messages de notification, le transmetteur mandataire va choisir plusieurs entrées dans le snmpProxyTable. Pour choisir ces entrées, il va effectuer les comparaisons suivantes :
- Le snmpProxyType doit être trap(3) si la notification est une PDU de classe Non confirmée. Le snmpProxyType doit être inform(4) si la demande est une PDU de classe Confirmée.
- Le contextEngineID doit être égal à l'objet snmpProxyContextEngineID.
- Si l'objet snmpProxyContextName est pris en charge, il doit être égal au contextName.
- L'objet snmpProxyTargetParamsIn identifie une entrée dans le snmpTargetParamsTable. Le messageProcessingModel, le modèle de sécurité, le securityName, et le securityLevel doivent correspondre à la valeur de snmpTargetParamsMPModel, de snmpTargetParamsSecurityModel, de snmpTargetParamsSecurityName, et de snmpTargetParamsSecurityLevel de l'entrée identifiée dans le snmpTargetParamsTable.

Toutes les entrées pour lesquelles ces conditions sont satisfaites sont retenues. L'objet snmpProxyMultipleTargetOut de chacune de ces entrées est utilisé pour choisir un ensemble d'entrées dans snmpTargetAddrTable. Toute snmpTargetAddrEntry dont l'objet snmpTargetAddrTagList contient une valeur d'étiquette égale à la valeur de snmpProxyMultipleTargetOut, et dont l'objet snmpTargetAddrParams fait référence à une entrée existante dans snmpTargetParamsTable, est choisi comme destination pour la notification transmise.

8. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait 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 copyrights, 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 .

9. 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 s'est pu 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.).

10. Considérations pour la sécurité

Les applications SNMP décrites dans le présent document ont normalement un accès direct aux instrumentations de MIB. Donc, il est très important que ces applications soient strictes dans leur application du contrôle d'accès comme décrit dans le présent document.

De plus, il peut y avoir certains types d'applications de générateur de notifications qui, plutôt que d'accéder à une instrumentation de MIB en utilisant le contrôle d'accès, vont obtenir les informations de MIB par d'autres moyens (comme à partir d'une ligne de commande). Les mises en œuvre et les utilisateurs de telles applications doivent être responsables de la non divulgation des informations de MIB qui seraient normalement inaccessibles à cause du contrôle d'accès.

Finalement, les MIB décrites dans le présent document contiennent des informations potentiellement sensibles. Un administrateur de la sécurité peut souhaiter limiter l'accès à ces MIB.

11. Références

11.1 Références normatives

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

[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)

[RFC3411] D. Harrington, R. Presuhn, B. Wijnen, "Architecture de description des cadres de gestion du protocole simple de

gestion de réseau (SNMP)", décembre 2002. (*MàJ par* RFC5343) (STD0062)

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

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

11.2 Références pour information

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

[RFC1213] K. McCloghrie et M. Rose, "Base de données d'informations de gestion pour la gestion de réseau des internets fondés sur TCP/IP : MIB-II", STD 17, mars 1991.

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

Appendice A Exemple de configuration de Trap

Cette section décrit un exemple de configuration pour une application de générateur de notification qui met en œuvre le niveau de sécurité snmpNotifyBasicCompliance. L'exemple de configuration spécifie que le générateur de notification devrait envoyer les notifications à trois gestionnaires séparés, en utilisant l'authentification et pas de confidentialité pour les deux premiers gestionnaires, et en utilisant à la fois l'authentification et la confidentialité pour le troisième gestionnaire.

La configuration consiste en trois rangées dans snmpTargetAddrTable, deux rangées dans snmpTargetTable, et deux rangées dans snmpNotifyTable.

```
* snmpTargetAddrName       = "addr1"
  snmpTargetAddrTDomain    = snmpUDPDomain
  snmpTargetAddrTAddress   = 128.1.2.3/162
  snmpTargetAddrTagList    = "group1"
  snmpTargetAddrParams     = "AuthNoPriv-joe"
  snmpTargetAddrStorageType = readOnly(5)
  snmpTargetAddrRowStatus  = active(1)

* snmpTargetAddrName       = "addr2"
  snmpTargetAddrTDomain    = snmpUDPDomain
  snmpTargetAddrTAddress   = 128.2.4.6/162
  snmpTargetAddrTagList    = "group1"
  snmpTargetAddrParams     = "AuthNoPriv-joe"
  snmpTargetAddrStorageType = readOnly(5)
  snmpTargetAddrRowStatus  = active(1)

* snmpTargetAddrName       = "addr3"
  snmpTargetAddrTDomain    = snmpUDPDomain
  snmpTargetAddrTAddress   = 128.1.5.9/162
  snmpTargetAddrTagList    = "group2"
  snmpTargetAddrParams     = "AuthPriv-bob"
  snmpTargetAddrStorageType = readOnly(5)
  snmpTargetAddrRowStatus  = active(1)

* snmpTargetParamsName              = "AuthNoPriv-joe"
  snmpTargetParamsMPModel            = 3
```

```
   snmpTargetParamsSecurityModel      = 3 (USM)
   snmpTargetParamsSecurityName       = "joe"
   snmpTargetParamsSecurityLevel      = authNoPriv(2)
   snmpTargetParamsStorageType        = readOnly(5)
   snmpTargetParamsRowStatus          = active(1)

* snmpTargetParamsName               = "AuthPriv-bob"
   snmpTargetParamsMPModel            = 3
   snmpTargetParamsSecurityModel      = 3 (USM)
   snmpTargetParamsSecurityName       = "bob"
   snmpTargetParamsSecurityLevel      = authPriv(3)  snmpTargetParamsStorageType        = readOnly(5)
   snmpTargetParamsRowStatus          = active(1)

* snmpNotifyName       = "group1"
   snmpNotifyTag        = "group1"
   snmpNotifyType       = trap(1)
   snmpNotifyStorageType  = readOnly(5)
   snmpNotifyRowStatus    = active(1)

* snmpNotifyName       = "group2"
   snmpNotifyTag        = "group2"
   snmpNotifyType       = trap(1)
   snmpNotifyStorageType  = readOnly(5)
   snmpNotifyRowStatus    = active(1)
```

Ces entrées définissent deux groupes de cibles de gestion. Le premier groupe contient deux cibles de gestion :

	première cible	seconde cible
messageProcessingModel	SNMPv3	SNMPv3
securityModel	3 (USM)	3 (USM)
securityName	"joe"	"joe"
securityLevel	authNoPriv(2)	authNoPriv(2)
transportDomain	snmpUDPDomain	snmpUDPDomain
transportAddress	128.1.2.3/162	128.2.4.6/162

Et le second groupe contient une seule cible de gestion :

messageProcessingModel	SNMPv3
securityLevel	authPriv(3)
securityModel	3 (USM)
securityName	"bob"
transportDomain	snmpUDPDomain
transportAddress	128.1.5.9/162

Adresse des éditeurs

David B. Levi
Nortel Networks
3505 Kesterwood Drive
Knoxville, TN 37918
U.S.A.
Téléphone : +1 865 686 0432
mél : dlevi@nortelnetworks.com

Paul Meyer
Secure Computing Corporation
2675 Long Lake Road
Roseville, MN 55113
U.S.A.
Téléphone : +1 651 628 1592
mél : paul_meyer@securecomputing.com

Bob Stewart
Retraité

Déclaration complète de droits de reproduction

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