Groupe de travail Réseau
Request for Comments : 3412
STD : 62
RFC rendue obsolète : 2572
Catégorie : Norme
Traduction Claude Brière de L'Isle

J. Case, SNMP Research, Inc.
D. Harrington, Enterasys Networks
R. Presuhn, BMC Software, Inc.
B. Wijnen, Lucent Technologies
décembre 2002

Traitement et répartition des messages pour le protocole simple de gestion de réseau (SNMP)

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

Résumé
Le présent document décrit le traitement et la répartition des messages pour le protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*) au sein de l'architecture SNMP. Il définit les procédures de répartition de versions potentiellement multiples de messages SNMP aux modèles de traitement de message SNMP appropriés, et de répartition des PDU aux applications SNMP. Le présent document décrit aussi un modèle de traitement de message - le modèle de traitement de message SNMPv3. Le présent document rend obsolète la [RFC2572].

Table des Maztières

1. Introduction

L'architecture pour décrire les cadres de gestion de l'Internet [RFC3411] précise qu'un moteur SNMP se compose :
1) d'un répartiteur *(Dispatcher)*
2) d'un sous système de traitement de message *(Message Processing Subsystem)*
3) d'un sous système de sécurité, et
4) d'un sous système de contrôle d'accès.

Les applications utilisent les services de ces sous systèmes.

Il est important de comprendre l'architecture de SNMP et sa terminologie pour comprendre que le sous système de traitement de message et le répartiteur décrits dans le présent document trouvent leur place dans l'architecture et interagissent avec les autres sous systèmes au sein de l'architecture. On suppose que le lecteur a pris connaissance et compris la description de l'architecture SNMP, définie dans la [RFC3411].

Le répartiteur dans le moteur SNMP envoie et reçoit les messages SNMP. Il répartit aussi les PDU SNMP aux applications SNMP. Lorsque un message SNMP doit être préparé ou lorsque des données doivent être extraites d'un message SNMP, le répartiteur délègue ces tâches à un modèle de traitement de message spécifique de la version de message au sein du sous système de traitement de message.

Un modèle de traitement de message est chargé de traiter un message spécifique de la version SNMP et de coordonner l'interaction avec le sous système de sécurité pour s'assurer qu'une sécurité appropriée est appliquée au message SNMP en cours de traitement.

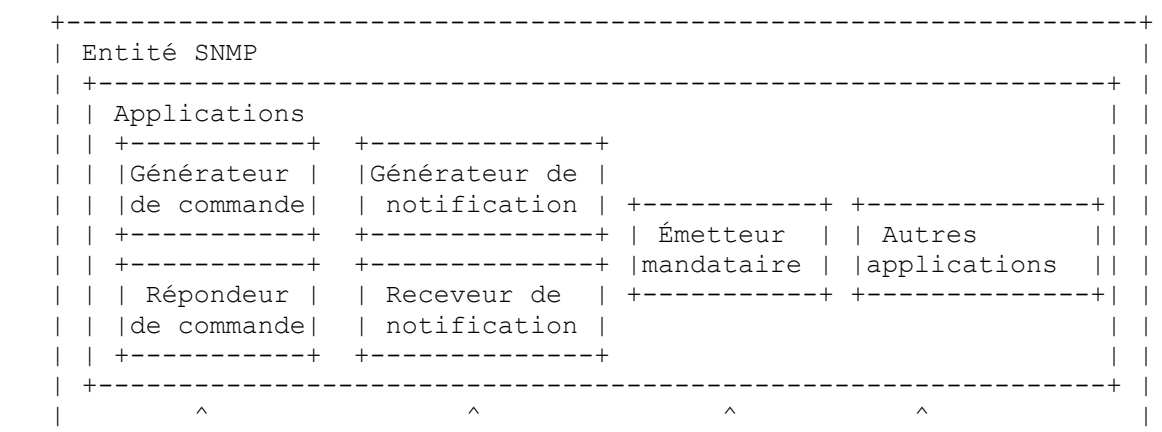
Les interactions entre le répartiteur, le sous système de traitement de message, et les applications sont modélisées en utilisant des éléments de données abstraits et des primitives d'interface de service abstraites définis par l'architecture SNMP.

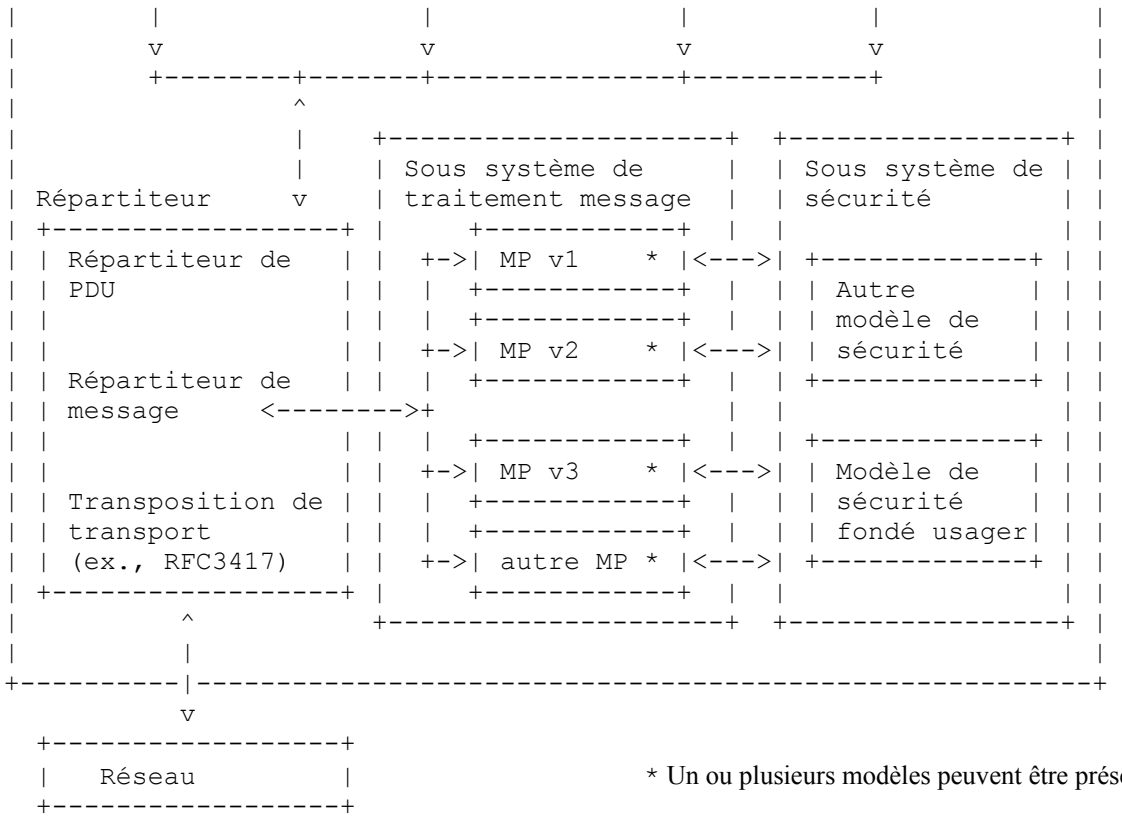
De même, les interactions entre le sous système de traitement de message et le sous système de sécurité sont modélisées en utilisant des éléments de données abstraits et des primitives d'interface de service abstraites définis par l'architecture SNMP.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" dans le présent document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

2. Généralités

L'illustration suivante dépeint le traitement de message en relation avec les applications SNMP, le sous système de sécurité et la transposition de transport.





2.1 Répartiteur

Le répartiteur est une pièce clé d'un moteur SNMP. Il y en a seulement un dans un moteur SNMP ; son travail est de répartir les tâches entre les multiples modèles de traitement de message spécifiques de version, et de répartir les PDU aux diverses applications.

Pour les messages sortants, une application fournit une PDU à envoyer, plus les données nécessaires pour préparer et envoyer le message, et l'application spécifie quel modèle de traitement de messages spécifique d'une version sera utilisé pour préparer le message avec le traitement de sécurité désiré. Une fois le message prêt, le répartiteur l'envoie.

Pour les messages entrants, le répartiteur détermine la version SNMP du message entrant et le passe au modèle de traitement de messages spécifique de la version pour extraire les composants du message et coordonner le traitement des services de sécurité pour le message. Après le traitement spécifique de la version, le répartiteur de PDU détermine quelle application, s'il en est, devrait recevoir la PDU pour la traiter et la transmettre en conséquence.

Le répartiteur, tout en envoyant et recevant les messages SNMP, collecte des statistiques sur les messages SNMP et le comportement du moteur SNMP dans les objets gérés pour les rendre accessibles aux entités SNMP distantes. Le présent document définit ces objets gérés, le module de MIB qui les contient, et comment ces objets gérés peuvent être utilisés pour une gestion utile.

2.2 Sous système de traitement de message

Le sous système de traitement de messages SNMP est la partie d'un moteur SNMP qui interagit avec le répartiteur pour traiter les messages SNMP spécifiques d'une version. Il contient un ou plusieurs modèles de traitement de messages.

Le présent document décrit un modèle de traitement de messages, le modèle de traitement de messages SNMPv3, à la Section 6. Le modèle de traitement de messages SNMPv3 est défini dans une autre section pour montrer que plusieurs modèles de traitement de messages (indépendants) peuvent exister au même moment et que de tels modèles peuvent être décrits dans différents documents. Le modèle de traitement de messages SNMPv3 pourra être remplacé ou complété par d'autres modèles de traitement de messages à l'avenir. Deux modèles de traitement de messages dont on attend le développement prochain sont le format de message SNMPv1 [RFC1157] et le format de message SNMPv2c [RFC1901]. D'autres peuvent être développés en tant que de besoin.

3. Éléments de traitement et de répartition de message

Voir dans la [RFC3411] la définition de : contextEngineID, contextName, scopedPDU, maxSizeResponseScopedPDU, securityModel, securityName, securityLevel, messageProcessingModel.

Pour les messages entrants, un module de traitement de message spécifique de la version fournit ces valeurs au répartiteur. Pour les messages sortants, une application fournit ces valeurs au répartiteur.

Pour certains traitements spécifiques de la version, les valeurs peuvent être extraites des messages reçus ; pour d'autres versions, les valeurs peuvent être déterminées par un algorithme, ou par un mécanisme défini par la mise en œuvre. Le mécanisme par lequel la valeur est déterminée n'est pas pertinent pour le répartiteur.

Les définitions supplémentaires ou étendues suivantes sont à utiliser au sein du répartiteur.

3.1 messageProcessingModel

La valeur de messageProcessingModel identifie un modèle de traitement de messages. Un modèle de traitement de messages décrit les procédures spécifiques de la version pour extraire les données des messages, générer les messages, invoquer un securityModel pour appliquer ses services de sécurité aux messages, pour convertir les données d'un format de message spécifique de la version en un format générique utilisable par le répartiteur, et pour convertir les données d'un format de répartiteur en un format de message spécifique de la version.

3.2 pduVersion

La valeur de pduVersion représente une version spécifique d'une opération du protocole et de ses formats de PDU associés, comme SNMPv1 ou SNMPv2 [RFC3416]. Les valeurs de pduVersion sont spécifiques de la version de la PDU contenue dans un message, et des PDU traitées par les applications. Le répartiteur n'utilise pas directement la valeur de pduVersion.

Une application spécifie la pduVersion lorsque elle demande au répartiteur de PDU d'envoyer une PDU à un autre moteur SNMP. Le répartiteur passe la pduVersion à un modèle de traitement de messages, afin qu'il sache comment traiter correctement cette PDU.

Pour les messages entrants, la pduVersion est fournie au répartiteur par un module de traitement de message spécifique de la version. Le répartiteur de PDU passe la pduVersion à l'application afin qu'elle sache comment traiter correctement la PDU. Par exemple, une application qui répond à une commande a besoin de savoir si elle doit utiliser des éléments de procédure et la syntaxe de la [RFC3416] au lieu de ceux spécifiés pour SNMPv1.

3.3 pduType

Une valeur de pduType représente un type spécifique d'opération de protocole. Les valeurs de pduType sont spécifiques de la version de la PDU contenue dans un message. Les applications s'enregistrent pour prendre en charge des pduType particuliers pour des contextEngineID particuliers. Pour les messages entrants, pduType est fourni au répartiteur par un module de traitement de message spécifique de la version. Il est utilisé ultérieurement pour attribuer la PDU à l'application qui s'est enregistrée pour le pduType pour le contextEngineID de la scopedPDU associée.

3.4 sendPduHandle

Cette bride est générée pour coordonner le traitement des demandes et réponses entre le moteur SNMP et une application. La bride doit être unique à travers tous les modèles de traitement de messages spécifiques de la version, et n'a qu'une signification locale.

4. Éléments de procédure du répartiteur

Cette section décrit les procédures suivies par le répartiteur pour générer et traiter les messages SNMP.

4.1 Envoi d'un message SNMP au réseau

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il envoie un message SNMP.

4.1.1 Envoi d'une demande ou d'une notification

Les procédures suivantes sont respectées par le répartiteur lorsque une application veut envoyer une PDU SNMP à une autre application (distante) c'est-à-dire, pour initier une communication en générant un message, contenant par exemple une demande ou une notification.

1) L'application demande cela en utilisant la primitive de service abstraite :

```
statusInformation =  sendPdu(          -- sendPduHandle si réussite, errorIndication si échec
      IN   transportDomain              -- domaine de transport à utiliser
      IN   transportAddress             -- adresse du réseau de destination
      IN   messageProcessingModel       -- normalement, 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                   -- version de la PDU
      IN   PDU                          -- unité de données de protocole SNMP
      IN   expectResponse               -- VRAI ou FAUX
        )
```

2) Si la valeur du messageProcessingModel ne représente pas un modèle de traitement de message connu du répartiteur, une errorIndication (selon la mise en œuvre) est alors retournée à l'application appelante. Aucun autre traitement n'est effectué.

3) Le répartiteur génère un sendPduHandle pour coordonner la suite du traitement.

4) Le répartiteur de messages envoie la demande au module de traitement de message spécifique de la version identifié par messageProcessingModel en utilisant la primitive de service abstraite :

```
statusInformation =                     -- indication de succès ou d'échec
   prepareOutgoingMessage(
      IN   transportDomain              -- comme spécifié par l'application
      IN   transportAddress             -- comme spécifié par l'application
      IN   messageProcessingModel       -- comme spécifié par l'application
      IN   securityModel                -- comme spécifié par l'application
      IN   securityName                 -- comme spécifié par l'application
      IN   securityLevel                -- comme spécifié par l'application
      IN   contextEngineID              -- comme spécifié par l'application
      IN   contextName                  -- comme spécifié par l'application
      IN   pduVersion                   -- comme spécifié par l'application
      IN   PDU                          -- comme spécifié par l'application
      IN   expectResponse               -- comme spécifié par l'application
      IN   sendPduHandle                -- comme déterminé à l'étape 3.
     OUT   destTransportDomain          -- domaine de transport de destination
     OUT   destTransportAddress         -- adresse de transport de destination
     OUT   outgoingMessage              -- le message à envoyer
     OUT   outgoingMessageLength        -- longueur du message
       )
```

5) Si les statusInformation indiquent une erreur, l'errorIndication est retournée à l'application appelante. Aucun autre traitement n'est effectué.

6) Si les statusInformation indiquent la réussite, la sendPduHandle est retournée à l'application, et le outgoingMessage est envoyé. Le transport utilisé pour envoyer le outgoingMessage est retourné via destTransportDomain, et l'adresse à laquelle il a été envoyé est retournée via destTransportAddress.

Le traitement du message sortant est achevé.

4.1.2 Envoi d'un réponse au réseau

La procédure suivante est suivie lorsque une application veut retourner une réponse au générateur d'une demande SNMP.

1) Une application peut demander cela en utilisant la primitive de service abstraite :

```
result =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 sur une demande entrante
    IN  contextEngineID             -- données de/à cette entité SNMP
    IN  contextName                 -- données de/dans ce contexte
    IN  pduVersion                  -- version de la PDU
    IN  PDU                         -- unités de données de protocole SNMP
    IN  maxSizeResponseScopedPDU    -- taille maximum de la PDU de réponse
    IN  stateReference              -- référence aux informations d'état comme présentées dans la demande
    IN  statusInformation           -- indication de succès ou d'erreur (OID de compteur d'erreur et valeur
                                    -- quand c'est une indication d'erreur)
    )
```

2) Le répartiteur de messages envoie la demande au modèle de traitement de messages approprié indiqué par la valeur reçue de messageProcessingModel en utilisant la primitive de service abstraite :

```
result =prepareResponseMessage(     -- indication de succès ou d'erreur
    IN  messageProcessingModel      -- spécifié par l'application
    IN  securityModel               -- spécifié par l'application
    IN  securityName                -- spécifié par l'application
    IN  securityLevel               -- spécifié par l'application
    IN  contextEngineID             -- spécifié par l'application
    IN  contextName                 -- spécifié par l'application
    IN  pduVersion                  -- spécifié par l'application
    IN  PDU                         -- spécifié par l'application
    IN  maxSizeResponseScopedPDU    -- spécifié par l'application
    IN  stateReference              -- spécifié par l'application
    IN  statusInformation           -- spécifié par l'application
   OUT  destTransportDomain         -- domaine de transport de destination
   OUT  destTransportAddress        -- adresse de transport de destination
   OUT  outgoingMessage             -- le message à envoyer
   OUT  outgoingMessageLength       -- longueur du message
    )
```

3) Si le résultat est une errorIndication, l'errorIndication est retournée à l'application appelante. Aucun autre traitement n'est effectué.
4) Si le résultat est réussi, le outgoingMessage est envoyé. Le transport utilisé pour envoyer le message sortant est retourné via destTransportDomain, et l'adresse à laquelle il a été envoyé est retournée via destTransportAddress.

Le traitement du message est achevé.

4.2 Réception d'un message SNMP du réseau

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il reçoit un message SNMP.

On notera que, pour être précis et pour ne pas rallonger et compliquer le texte, certains détails ont été omis dans les étapes qui suivent. En particulier, les éléments de procédure n'indiquent pas toujours explicitement quand les informations d'état doivent être libérées. La règle générale est que si les informations d'état sont disponibles lorsque un message doit être "éliminé sans autre traitement", les informations d'état doivent aussi être libérées au même moment.

4.2.1 Répartition des messages SNMP reçus

1) Le compteur snmpInPkts [RFC3418] est incrémenté.
2) La version du message SNMP est déterminée d'une manière qui dépend de la mise en œuvre. Si le paquet ne peut pas être

suffisamment analysé pour déterminer la version du message SNMP, le compteur snmpInASNParseErrs [RFC3418] est incrémenté, et le message est éliminé sans autre traitement. Si la version n'est pas prise en charge, le compteur snmpInBadVersions [RFC3418] est alors incrémenté, et le message est éliminé sans autre traitement.

3) Le transportDomain et la transportAddress d'origine sont déterminés.
4) Le message est passé au modèle de traitement de messages spécifique de la version qui retourne les éléments de données abstraits requis par le répartiteur. Cela est effectué en utilisant la primitive de service abstraite :

```
result = prepareDataElements(              -- indication de succès ou d'erreur
    IN   transportDomain                   -- origine comme déterminé à l'étape 3.
    IN   transportAddress                  -- origine comme déterminé à l'étape 3.
    IN   wholeMsg                          -- comme reçu par le réseau
    IN   wholeMsgLength                    -- comme reçu par le réseau
   OUT   messageProcessingModel            -- normalement, la version SNMP
   OUT   securityModel                     -- modèle de sécurité spécifié
   OUT   securityName                      -- au nom de ce principal
   OUT   securityLevel                     -- niveau de sécurité spécifié
   OUT   contextEngineID                   -- données de/à cette entité
   OUT   contextName                       -- données de/dans ce contexte
   OUT   pduVersion                        -- version de la PDU
   OUT   PDU                               -- unités de données de protocole SNMP
   OUT   pduType                           -- type de PDU SNMP
   OUT   sendPduHandle                     -- bride pour une demande qui correspond
   OUT   maxSizeResponseScopedPDU          -- taille maximum de la PDU de réponse
   OUT   statusInformation                 -- indication de succès ou d'erreur (OID et valeur de compteur d'erreur
                                           -- si errorIndication)
   OUT   stateReference                    -- référence aux informations d'état à utiliser pour une réponse possible
   )
```

5) Si le résultat est une errorIndication ÉCHEC, le message est éliminé sans autre traitement.

6) À ce point, les éléments de données abstraits ont été préparés et le traitement continue comme décrit au paragraphe 4.2.2 "Répartition des PDU pour les messages entrants".

4.2.2 Répartition des PDU pour les messages entrants

Les éléments de procédure pour la répartition des PDU dépendent de la valeur de sendPduHandle. Si la valeur de sendPduHandle est <aucune>, c'est alors une demande ou une notification et les procédures spécifiées au paragraphe 4.2.2.1 s'appliquent. Si la valeur de snmpPduHandle n'est pas <aucune>, c'est alors une réponse et les procédures spécifiées au paragraphe 4.2.2.2 s'appliquent.

4.2.2.1 Demandes et notifications entrantes

Les procédures suivantes sont suivies pour la répartition des PDU lorsque la valeur de sendPduHandle est <aucune>, indiquant que c'est une demande ou une notification.

1) La combinaison de contextEngineID et de pduType est utilisée pour déterminer quelle application s'est enregistrée pour cette demande ou notification.

2) Si aucune application ne s'est enregistrée pour la combinaison, alors :
   a) Le compteur snmpUnknownPDUHandlers est incrémenté.
   b) Un message de réponse est généré en utilisant la primitive de service abstraite :

```
result =                                   -- SUCCÈS ou ÉCHEC
    prepareResponseMessage(
    IN   messageProcessingModel            -- comme fourni par le module MP
    IN   securityModel                     -- comme fourni par le module MP
    IN   securityName                      -- comme fourni par le module MP
    IN   securityLevel                     -- comme fourni par le module MP
    IN   contextEngineID                   -- comme fourni par le module MP
    IN   contextName                       -- comme fourni par le module MP
    IN   pduVersion                        -- comme fourni par le module MP
    IN   PDU                               -- comme fourni par le module MP
    IN   maxSizeResponseScopedPDU          -- comme fourni par le module MP
```

```
    IN  stateReference              -- comme fourni par le module MP
    IN  statusInformation           -- paire de valeurs d'OID errorIndication plus snmpUnknownPDUHandlers
   OUT  destTransportDomain         -- destination transportDomain
   OUT  destTransportAddress        -- destination transportAddress
   OUT  outgoingMessage             -- le message à envoyer
   OUT  outgoingMessageLength       -- sa longueur
   )
```

c) Si le résultat est SUCCÈS, le message préparé est alors envoyé au générateur de la demande tel qu'identifié par le transportDomain et la transportAddress. Le transport utilisé pour envoyer le outgoingMessage est retourné via destTransportDomain, et l'adresse à laquelle il a été envoyé est retournée via destTransportAddress.

d) Le message entrant est éliminé sans autre traitement. Le traitement de message est achevé pour ce message.

3) La PDU est fournie à l'application, en utilisant la primitive de service abstraite :

```
processPdu(                         -- traiter la demande/notification
    IN  messageProcessingModel      -- comme fourni par le module MP
    IN  securityModel               -- comme fourni par le module MP
    IN  securityName                -- comme fourni par le module MP
    IN  securityLevel               -- comme fourni par le module MP
    IN  contextEngineID             -- comme fourni par le module MP
    IN  contextName                 -- comme fourni par le module MP
    IN  pduVersion                  -- comme fourni par le module MP
    IN  PDU                         -- comme fourni par le module MP
    IN  maxSizeResponseScopedPDU    -- comme fourni par le module MP
    IN  stateReference              -- comme fourni par le module MP, nécessaire à l'envoi d'une réponse
     )
```

Le traitement de message est terminé pour ce message.

4.2.2.2 Réponses entrantes

Les procédures suivantes sont suivies pour la répartition des PDU lorsque la valeur de sendPduHandle n'est pas <aucune>, indiquant que c'est une réponse.

1) La valeur de sendPduHandle est utilisée pour déterminer, d'une manière définie par la mise en œuvre, quelle application attend une réponse associée à cette sendPduHandle.
2) Si on ne trouve aucune application en attente, le message est éliminé sans autre traitement, et la stateReference est libérée. Le compteur snmpUnknownPDUHandlers est incrémenté. Le traitement de message est terminé pour ce message.
3) Toutes les informations en antémémoire sur le message, y compris la stateReference, sont éliminées.
4) La réponse est fournie à l'application en utilisant la primitive de service abstraite :

```
processResponsePdu(                 -- traiter la PDU de réponse
    IN  messageProcessingModel      -- fourni par le module MP
    IN  securityModel               -- fourni par le module MP
    IN  securityName                -- fourni par le module MP
    IN  securityLevel               -- fourni par le module MP
    IN  contextEngineID             -- fourni par le module MP
    IN  contextName                 -- fourni par le module MP
    IN  pduVersion                  -- fourni par le module MP
    IN  PDU                         -- fourni par le module MP
    IN  statusInformation           -- fourni par le module MP
    IN  sendPduHandle               -- fourni par le module MP
      )
```

Le traitement de message est terminé pour ce message.

4.3 Enregistrement d'application pour le traitement des types de PDU

Les applications qui veulent traiter certaines PDU doivent s'enregistrer auprès du répartiteur de PDU. Les applications spécifient la combinaison de contextEngineID et de pduType qu'elles veulent prendre en charge.

1) Une application s'enregistre conformément à la primitive d'interface abstraite :

```
statusInformation =             -- indication de succès ou d'erreur
 registerContextEngineID(
   IN   contextEngineID         -- prendre en charge celui-ci
   IN   pduType                 -- le ou les pduType à enregistrer
    )
```

Note : Les mises en œuvre peuvent fournir un moyen de demander l'enregistrement pour plusieurs valeurs simultanées de contextEngineID, par exemple, toutes les valeurs de contextEngineID, et peuvent aussi fournir un moyen de demander l'enregistrement simultané de plusieurs valeurs du pduType.

2) La validité des paramètres peut être vérifiée ; si ils ne sont pas valides, une indication d'erreur (invalidParameter) est retournée à l'application.
3) Chaque combinaison de contextEngineID et de pduType ne peut être enregistrée qu'une seule fois. Si une autre application s'est déjà enregistrée pour la combinaison spécifiée, une indication d'erreur (alreadyRegistered) est alors retournée à l'application.
4) Autrement, l'enregistrement est sauvegardé afin que ces PDU SNMP puissent être envoyées à cette application.

4.4 Désenregistrement d'application pour le traitement de types de PDU

Les applications qui ne veulent plus traiter certaines PDU doivent se désenregistrer auprès du répartiteur de PDU.

1) Une application se désenregistre en utilisant la primitive de service abstraite :

```
unregisterContextEngineID(
   IN   contextEngineID         -- abandonner la prise en charge de ceci
   IN   pduType                 -- le ou les pduType à désenregistrer
     )
```

Note : Les mises en œuvre peuvent fournir un moyen pour demander le désenregistrement simultané de plusieurs valeurs de contextEngineID, par exemple, toutes les valeurs de contextEngineID, et peuvent aussi fournir un moyen de demander simultanément le désenregistrement de plusieurs valeurs de pduType.

2) Si la combinaison contextEngineID et pduType a été enregistrée, l'enregistrement est alors supprimé.

Si un tel enregistrement n'existe pas, la demande est alors ignorée.

5. Définitions

5.1 Définitions pour le traitement et la répartition des messages SNMP

```
DEFINITIONS SNMP-MPD-MIB ::= DEBUT

IMPORTATIONS
CONFORMITÉ-DE-MODULE, GROUPE-D'OBJET :  DE SNMPv2-CONF
IDENTITÉ-DE-MODULE, TYPE-D'OBJET,
snmpModules, Counter32 :  DE SNMPv2-SMI;

IDENTITÉ-DE-MODULE snmpMPDMIB
DERNIERE MISE A JOUR : "200210140000Z"
ORGANISATION : "Groupe de travail SNMPv3"
INFORMATIONS DE CONTACT : "WG-EMail:   snmpv3@lists.tislabs.com
                S'abonner à :  snmpv3-request@lists.tislabs.com

        Co-Chair :   Russ Mundy
                     Network Associates Laboratories
adresse postale :    15204 Omega Drive, Suite 300
                     Rockville, MD 20850-4601
                     USA
             mèl :     mundy@tislabs.com
       téléphone :     +1 301-947-7107
```

Co-Chair & Co-editeur : David Harrington
Enterasys Networks
adresse postale: 35 Industrial Way
P. O. Box 5005
Rochester NH 03866-5005
USA
mèl: dbh@enterasys.com
téléphone : +1 603-337-2614

Co-editeur : Jeffrey Case
SNMP Research, Inc.
adresse postale : 3001 Kimberlin Heights Road
Knoxville, TN 37920-9716
USA
mèl : case@snmp.com
téléphone : +1 423-573-1434

Co-éditeur : Randy Presuhn
BMC Software, Inc.
adresse postale : 2141 North First Street
San Jose, CA 95131
USA
mèl : randy_presuhn@bmc.com
téléphone : +1 408-546-1006

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

DESCRIPTION : "MIB pour le traitement et l'expédition de messages. Copyright (C) The Internet Society (2002). Cette version de ce module de MIB fait partie de la RFC 3412 ; voir dans la RFC elle-même les notices légales complètes. "
REVISION : "200210140000Z" -- 14 octobre 2002
DESCRIPTION : "Adresses mises à jour, publié comme RFC 3412."
REVISION : "199905041636Z" -- 4 mai 1999
DESCRIPTION : "Adresses mises à jour, publié comme RFC 2572."
REVISION : "199709300000Z" -- 30 septembre 1997
DESCRIPTION : "Version originale, publiée comme RFC 2272."
::= { snmpModules 11 }

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

IDENTIFIANT D'OBJET snmpMPDAdmin ::= { snmpMPDMIB 1 }
IDENTIFIANT D'OBJET snmpMPDMIBObjects ::= { snmpMPDMIB 2 }
IDENTIFIANT D'OBJET snmpMPDMIBConformance ::= { snmpMPDMIB 3 }

-- Statistiques des messages SNMP **********

IDENTIFIANT D'OBJET snmpMPDStats ::= { snmpMPDMIBObjects 1 }

TYPE D'OBJET snmpUnknownSecurityModels
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre total de paquets reçus par le moteur SNMP qui ont été éliminés à cause de leur référence à un modèle de sécurité inconnu ou non pris en charge par le moteur SNMP."
::= { snmpMPDStats 1 }

TYPE D'OBJET snmpInvalidMsgs
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre total de paquets reçus par le moteur SNMP qui ont été éliminés à cause de composants invalides ou incohérents dans le message SNMP."
::= { snmpMPDStats 2 }

TYPE D'OBJET snmpUnknownPDUHandlers
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce que la PDU contenue dans le paquet n'a pas pu être passée à une application prenant en charge le traitement du pduType, par exemple. aucune application SNMP ne s'est enregistrée pour la bonne combinaison de contextEngineID et pduType."
::= { snmpMPDStats 3 }

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

IDENTIFIANT D'OBJET snmpMPDMIBCompliances ::= {snmpMPDMIBConformance 1}
IDENTIFIANT D'OBJET snmpMPDMIBGroups ::= {snmpMPDMIBConformance 2}

-- Déclarations de conformité

CONFORMITÉ-DE-MODULE snmpMPDCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les entités SNMP qui mettent en œuvre la SNMP-MPD-MIB. "
MODULE : ce module
GROUPES OBLIGATOIRES { snmpMPDGroup }
::= { snmpMPDMIBCompliances 1 }

GROUPE-D'OBJET snmpMPDGroup
OBJETS : { snmpUnknownSecurityModels, snmpInvalidMsgs, snmpUnknownPDUHandlers }
STATUT : actuel
DESCRIPTION : "Collection d'objets assurant la surveillance à distance du processus de traitement et d'expédition du message SNMP. "
::= { snmpMPDMIBGroups 1 }

FIN

6. Format de message SNMPv3

Cette section définit le format de message SNMPv3 et le modèle de traitement de messages SNMP version 3 (v3MP, *SNMPv3 Message Processing*) correspondant.

ÉTIQUETTES IMPLICITE DE DÉFINITIONS de SSNMPv3MessageSyntax ::= DÉBUT

SNMPv3Message ::= SEQUENCE { -- identifie la disposition du message SNMPv3, cet élément est dans la même position que dans SNMPv1 et SNMPv2c, permettant la reconnaissance de la valeur 3 qui est utilisée pour snmpv3.
msgVersion ENTIER (0 à 2 147 483 647), -- paramètres administratifs.
msgGlobalData HeaderData, -- format de paramètre spécifique du modèle de sécurité défini par le modèle de sécurité.
msgSecurityParameters CHAINE D'OCTETS,
msgData ScopedPduData
}

HeaderData ::= SEQUENCE {
msgID ENTIER (0 à 2 147 483 647),
msgMaxSize ENTIER (484 à 2 147 483 647),
msgFlags CHAINE D'OCTETS (TAILLE(1)),
-- ....1 authFlag

```
            -- .... ..1.   privFlag
            -- .... .1..   reportableFlag
            --             On observera que :
            -- .... ..00   va, il signifie noAuthNoPriv
            -- .... ..01   va, il signifie authNoPriv
            -- .... ..10   réservé, NE DOIT PAS être utilisé.
            -- .... ..11   va, il signifie authPriv
msgSecurityModel ENTIER (1 à 2 147 483 647)
    }

ScopedPduData ::= CHOIX {
      plaintext    ScopedPDU,
      encryptedPDU CHAINE D'OCTETS          -- valeur chiffrée de scopedPDU
   }

ScopedPDU ::= SEQUENCE {
      contextEngineID  CHAINE D'OCTETS,
      contextName      CHAINE D'OCTETS,
      data             ANY – par exemple, les PDU définies dans la [RFC3416]
   }
FIN
```

6.1 msgVersion

Le champ msgVersion est réglé à snmpv3(3) et identifie le message comme SNMP version 3.

6.2 msgID

Le msgID est utilisé entre deux entités SNMP pour coordonner les demandes et les réponses des messages, et par v3MP pour coordonner le traitement du message par différents modèles de sous système au sein de l'architecture.

Les valeurs pour msgID DEVRAIENT être générées d'une manière qui évite la réutilisation de valeurs en cours d'usage. Le faire assure la protection contre des attaques en répétition. Une stratégie de mise en œuvre possible serait d'utiliser les bits de moindre poids de snmpEngineBoots [RFC3411] comme portion de poids fort de la valeur de msgID et un entier à croissance monotone pour la portion de moindre poids de msgID.

Noter que l'identifiant de demande dans une PDU peut être utilisé par les applications SNMP pour identifier la PDU ; le msgID est utilisé par le moteur pour identifier le message qui porte une PDU. Le moteur a besoin d'identifier le message même si le déchiffrement de la PDU (et de l'identifiant de demande) échoue. On ne devrait pas supposer que la valeur du msgID et celle du request-id sont équivalentes.

La valeur du champ msgID pour une réponse prend la valeur du champ msgID du message dont il est une réponse. En utilisant la valeur de msgID, un moteur peut distinguer les demandes en cours (potentiellement multiples) et par là corréler les réponses entrantes avec les demandes en cours. Dans les cas d'utilisation d'un service de datagrammes non fiable, le msgID fournit aussi un moyen simple pour identifier les messages dupliqués par le réseau. Si une demande est retransmise, une nouvelle valeur de msgID DEVRAIT être utilisée pour chaque retransmission.

6.3 msgMaxSize

Le champ msgMaxSize du message porte la taille maximum de message prise en charge par l'envoyeur du message, c'est-à-dire, la taille maximum de message que l'envoyeur peut accepter quand un autre moteur SNMP envoie un message SNMP (que ce soit une réponse ou tout autre message) à l'envoyeur de ce message sur le transport utilisé pour ce message.

Lorsque un message SNMP est généré, la msgMaxSize est fournie par le moteur SNMP qui génère le message. Au moteur SNMP receveur, la msgMaxSize est utilisée pour déterminer la taille maximum de message dont l'envoyeur peut s'accommoder.

6.4 msgFlags

Le champ msgFlags du message contient plusieurs champs de bits qui contrôlent le traitement du message.

Le fanion reportableFlag est une aide secondaire pour déterminer si une PDU Report DOIT être envoyée. Il n'est utilisé que dans le cas où une portion de PDU d'un message ne peut pas être décodée, du fait, par exemple, d'une clé de chiffrement incorrecte. Si la PDU peut être décodée, le type de PDU forme la base de la décision d'envoyer une PDU Report.

Lorsque reportableFlag est utilisé, si sa valeur est un, une PDU Report DOIT être retournée à l'envoyeur dans les conditions qui peuvent causer la génération de PDU Report. De même, lorsque reportableFlag est utilisé et que sa valeur est zéro, une PDU Report NE DOIT PAS être envoyée. Le fanion reportableFlag DOIT toujours être à zéro lorsque le message contient une PDU de classe Non confirmé, telle qu'une PDU Report, une PDU de type réponse (comme une PDU Response) ou une PDU de type notification sans accusé de réception (comme une PDU SNMPv2-trap). Le fanion reportableFlag DOIT toujours être à un pour une PDU de la classe Confirmed, incluant les PDU de type de demande (comme une PDU Get) et les PDU de type de notification à accusé de réception (tels que une PDU Inform).

Si le fanion reportableFlag est réglé à un pour un message qui contient une PDU de la classe Non confirmé, comme une PDU Report, une PDU de type réponse (comme une PDU Response), ou une PDU de type de notification sans accusé de réception (comme une PDU SNMPv2-trap) le receveur d'un tel message DOIT alors le traiter bien que le fanion reportableFlag ait été réglé à zéro.

Si le fanion reportableFlag est réglé à zéro pour un message qui contient une PDU de type Demande (comme une PDU Get) ou une PDU de type Notification avec accusé de réception (comme une PDU Inform) le receveur de ce message DOIT alors le traiter bien que le fanion reportableFlag ait été réglé à un.

Les PDU Report sont générées directement par le modèle de traitement de messages SNMPv3, et prennent en charge les communications de moteur à moteur, mais peuvent être passées aux applications pour être traitées.

Un moteur SNMP qui reçoit une reportPDU peut l'utiliser pour déterminer quelle sorte de problème a été détecté par le moteur SNMP distant. Il peut le faire sur la base du compteur d'erreurs inclus comme première (et seule) varBind de la reportPDU. Sur la base de l'erreur détectée, le moteur SNMP peut essayer d'envoyer un message SNMP corrigé. Si ce n'est pas possible, il peut passer une indication de l'erreur à l'application au nom de laquelle a été produite la demande SNMP défaillante.

Les portions authFlag et privFlag du champ msgFlags sont réglées par l'envoyeur de façon à indiquer le niveau de sécurité qui a été appliqué au message avant son envoi dans le réseau. Le receveur du message DOIT appliquer le même niveau de sécurité lorsque le message est reçu et lorsque le contenu est traité.

Il y a trois niveaux de sécurité : noAuthNoPriv, qui est moins que authNoPriv, qui à son tour est moins que authPriv. Voir le document d'architecture SNMP [RFC3411] pour les détails sur le niveau de sécurité.

a) authFlag
Si le fanion authFlag est réglé à un, le modèle de sécurité utilisé par le moteur SNMP qui envoie le message DOIT alors identifier le securityName au nom duquel le message SNMP a été généré et DOIT fournir, d'une manière spécifique du modèle de sécurité, des données suffisantes pour que le receveur du message soit capable d'authentifier cette identification. En général, cette authentification va permettre au receveur de déterminer avec une certitude raisonnable que le message a été :
- envoyé au nom du principal associé au securityName,
- non redirigé,
- non modifié dans le transit, et
- non répété.

Si le authFlag est à zéro, le securityModel utilisé par le moteur SNMP qui a envoyé le message DOIT alors identifier le securityName au nom duquel le message SNMP a été généré mais il n'a pas besoin de fournir des données suffisantes pour que le receveur du message authentifie l'identification, car dans ce cas il n'est pas nécessaire d'authentifier le message.

b) privFlag
Si le fanion privFlag est établi, le securityModel utilisé par le moteur SNMP qui a envoyé le message DOIT alors aussi protéger contre la divulgation la scopedPDU dans un message SNMP, c'est-à-dire, il DOIT chiffrer/déchiffrer la scopedPDU. Si le privFlag est à zéro, le securityModel utilisé n'a alors pas besoin de protéger les données de la divulgation.

C'est une exigence explicite de l'architecture SNMP que si la confidentialité est choisie, l'authentification est aussi exigée. Cela signifie que si le privFlag est établi, le authFlag DOIT alors aussi être réglé à un.

La combinaison de authFlag et de privFlag comprend un niveau de sécurité comme suit :
authFlag zéro, privFlag zéro -> securityLevel est noAuthNoPriv.
authFlag zéro, privFlag un -> combinaison invalide , voir ci-dessous.

authFlag un, privFlag zéro -> securityLevel est authNoPriv.
authFlag un, privFlag un -> securityLevel est authPriv.

Les éléments de procédure (voir ci-dessous) décrivent l'action à effectuer lorsque une combinaison invalide de authFlag égal à zéro et privFlag égal à un est rencontrée.

Les bits restants dans msgFlags sont réservés, et DOIVENT être réglés à zéro à l'envoi d'un message et DEVRAIENT être ignorés à réception d'un message.

6.5 msgSecurityModel

Le v3MP prend en charge l'existence concurrente de plusieurs modèles de sécurité pour fournir des services de sécurité pour les messages SNMPv3. Le champ msgSecurityModel dans un message SNMPv3 identifie quel modèle de sécurité a été utilisé par l'envoyeur pour générer le message et donc quel securityModel DOIT être utilisé par le receveur pour effectuer le traitement de sécurité pour le message. La transposition en la mise en œuvre appropriée de securityModel au sein d'un moteur SNMP est accomplie d'une manière qui dépend de la mise en œuvre.

6.6 msgSecurityParameters

Le champ msgSecurityParameters du message SNMPv3 est utilisé pour la communication entre les modules du modèle de sécurité dans les moteurs SNMP d'envoi et de réception. Les données dans le champ msgSecurityParameters sont utilisées exclusivement par le modèle de sécurité, le contenu et le format des données sont définis par le modèle de sécurité. Cette CHAINE D'OCTETS n'est pas interprétée par v3MP, mais est passée à la mise en œuvre locale du modèle de sécurité indiqué par le champ msgSecurityModel dans le message.

6.7 scopedPduData

Le champ scopedPduData représente soit la scopedPDU en texte source si le privFlag dans les msgFlags est zéro, soit il représente une encryptedPDU (codée comme CHAINE D'OCTETS) qui DOIT être déchiffrée par le securityModel utilisé pour produire une scopedPDU en texte source.

6.8 scopedPDU

La scopedPDU contient des informations pour identifier un contexte administrativement unique et une PDU. Les identifiants d'objet dans la PDU se réfèrent aux objets gérés qui sont (supposés être) accessibles dans le contexte spécifié.

6.8.1 contextEngineID

Le contextEngineID dans le message SNMPv3 identifie de façon univoque, au sein d'un domaine administratif, une entité SNMP qui peut réaliser une instance d'un contexte avec un contextName particulier.

Pour les messages entrants, le contextEngineID est utilisé en conjonction avec le pduType pour déterminer à quelle application le scopedPDU sera envoyé pour traitement.

Pour les messages sortants, le v3MP règle le contextEngineID à la valeur fournie par l'application dans la demande d'un message à envoyer.

6.8.2 contextName

Le champ contextName dans un message SNMPv3, en conjonction avec le champ contextEngineID, identifie le contexte particulier associé aux informations de gestion contenues dans la portion PDU du message. Le contextName est unique au sein de l'entité SNMP spécifiée par le contextEngineID, qui peut réaliser les objets gérés référencés dans la PDU. Une application qui génère un message fournit la valeur pour le champ contextName et cette valeur peut être utilisée durant le traitement par une application au moteur SNMP receveur.

6.8.3 data

Le champ Données du message SNMPv3 contient la PDU. Entre autres choses, la PDU contient le type de PDU utilisé par le v3MP pour déterminer le type du message SNMP entrant. Le v3MP spécifie que la PDU DOIT être une de celles spécifiées dans la [RFC3416].

7. Éléments de procédure pour v3MP

Cette section décrit les procédures suivies par un moteur SNMP lorsque il génère et traite les messages SNMP conformément au modèle de traitement de messages SNMPv3.

Prière de noter que pour des besoins de clarté et pour que le texte ne devienne pas trop long et plus compliqué, certains détails ont été omis des étapes qui suivent.

a) Certaines étapes spécifient que lorsque des conditions d'erreur se rencontrent lors du traitement d'un message reçu, un message qui contient une PDU Report est généré et le message reçu est éliminé sans autre traitement. Cependant, une PDU Report NE DOIT PAS être générée si il peut être déterminé que la PDU qui cause la génération de la PDU Report est un membre de la classe Confirmed, ou si le reportableFlag est réglé à un et que la classe de la PDU ne peut pas être déterminée.

b) Les éléments de procédure n'indiquent pas toujours explicitement quand les informations d'état doivent être libérées. La règle générale est que si les informations d'état sont disponibles lorsque un message va être "éliminé sans autre traitement", les informations d'état devraient alors aussi être éliminées en même temps .

7.1 Préparation d'un message SNMP sortant

Ce paragraphe décrit la procédure suivie pour préparer un message SNMPv3 à partir des éléments de données passés par le répartiteur de messages.

1) Le répartiteur de messages peut demander qu'un message SNMPv3 contenant une PDU de classe écriture, de classe lecture ou de classe notification soit préparé pour envoi.
   a) Il fait une telle demande conformément à la primitive de service abstrait :

```
statusInformation =                 -- indication de succès ou d'erreur
      prepareOutgoingMessage(
        IN   transportDomain        -- domaine de transport demandé
        IN   transportAddress       -- adresse de destination demandée
        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             -- version de la PDU *
        IN   PDU                    -- unités de données de protocole SNMP
        IN   expectResponse         -- VRAI ou FAUX *
        IN   sendPduHandle          -- bride pour faire correspondre les réponses entrantes
       OUT   destTransportDomain    -- domaine de transport de destination
       OUT   destTransportAddress   -- adresse de transport de destination
       OUT   outgoingMessage        -- le message à envoyer
       OUT   outgoingMessageLength  -- la longueur du message
        )
```

* Le modèle de traitement de messages SNMPv3 n'utilise pas les valeurs de expectResponse ou pduVersion.

   b) Un msgID unique est généré. Le numéro utilisé pour msgID ne devrait pas avoir été utilisé récemment, et NE DOIT PAS être le même que celui utilisé pour une demande en cours.

2) Le répartiteur de messages peut demander qu'un message SNMPv3 contenant une PDU de classe Response ou Internal soit préparé pour envoi.
   a) Il fait une telle demande conformément à la primitive de service abstrait :

```
result = prepareResponseMessage(    -- SUCCÈS ou ÉCHEC
      IN   messageProcessingModel   -- normalement, la version SNMP
      IN   securityModel            -- le même que sur une demande entrante
      IN   securityName             -- le même que sur une demande entrante
      IN   securityLevel            -- le même que sur une demande entrante
      IN   contextEngineID          -- données de/à cette entité SNMP
```

```
    IN  contextName                 -- données de/dans ce contexte
    IN  pduVersion                  -- version de la PDU
    IN  PDU                         -- unités de données de protocole SNMP
    IN  maxSizeResponseScopedPDU    -- taille maximum que l'envoyeur peut accepter
    IN  stateReference              -- référence aux informations d'état présentée avec la demande
    IN  statusInformation           -- indication de succès ou d'erreur ;
                                    -- OID et valeur de compteur d'erreur si errorIndication
   OUT  destTransportDomain         -- domaine de transport de destination
   OUT  destTransportAddress        -- adresse de transport de destination
   OUT  outgoingMessage             -- le message à envoyer
   OUT  outgoingMessageLength       -- la longueur du message
   )
```

b) Les information en antémémoire pour la demande d'origine sont restituées via les stateReference, incluant :
- msgID, *(identifiant de message)*
- contextEngineID, *(identifiant de moteur du contexte)*
- contextName, *(nom du contexte)*
- securityModel, *(modèle de sécurité)*
- securityName, *(nom de sécurité)*
- securityLevel, *(niveau de sécurité)*
- securityStateReference, *(référence de l'état de sécurité)*
- reportableFlag, *(fanion rapportable)*
- transportDomain, *(domaine de transport)* et
- transportAddress *(adresse de transport)*.

Le modèle de traitement de messages SNMPv3 ne permet pas que des données en antémémoire soient écrasées, sauf par des indications d'erreur comme précisé en (3) ci-dessous.

3) Si statusInformation contient des valeurs pour une combinaison OID/valeur (contenant aussi potentiellement une valeur de securityLevel, de contextEngineID, ou de contextName) alors :
   a) Si une PDU est fournie, c'est la PDU de la demande d'origine. Si possible, extraire le request-id et le pduType.
   b) Si le pduType est déterminé comme n'étant pas membre de la classe Confirmed, ou si le reportableFlag est à zéro et le pduType ne peut pas être déterminé, alors le message d'origine est éliminé, et aucun autre traitement n'est effectué. Un résultat de ÉCHEC est retourné. Le traitement du message SNMPv3 est terminé.
   c) Une PDU Report est préparée :
      1) la varBindList est construite avec l'OID et la valeur provenant des statusInformation.
      2) error-status est réglé à 0.
      3) error-index est réglé à 0.
      4) request-id est réglé à la valeur extraite de l'étape b). Autrement, request-id est réglé à 0.
   d) errorIndication dans statusInformation peut être accompagné d'une valeur de securityLevel, de contextEngineID, ou de contextName.
      1) Si statusInformation contient une valeur de securityLevel, securityLevel est alors réglé à cette valeur, autrement il est réglé à noAuthNoPriv.
      2) Si statusInformation contient une valeur de contextEngineID, contextEngineID est alors réglé à cette valeur, autrement il est réglé à la valeur du snmpEngineID de cette entité.
      3) Si statusInformation contient une valeur de contextName, alors contextName est réglé à cette valeur, autrement, il est réglé au contexte par défaut de "" (chaîne de longueur zéro).
   e) La PDU est réglée à se référer à la nouvelle PDU de rapport. La vieille PDU est éliminée.
   f) Le traitement se continue à l'étape 6) ci-dessous.
4) Si le contextEngineID n'est pas encore déterminé, alors le contextEngineID est déterminé, d'une manière qui dépend de la mise en œuvre, éventuellement en utilisant le transportDomain et le transportAddress.
5) Si le contextName n'est pas encore déterminé, le contextName est réglé au contexte par défaut.
6) Une scopedPDU est préparée à partir de contextEngineID, contextName, et de la PDU.
7) msgGlobalData est construit comme suit :
   a) Le champ msgVersion est réglé à snmpv3(3).
   b) msgID est réglé comme déterminé à l'étape 1 ou 2 ci-dessus.
   c) msgMaxSize est réglé à une valeur qui dépend de la mise en œuvre.
   d) les msgFlag sont réglés comme suit :
      - si securityLevel spécifie noAuthNoPriv, alors authFlag et privFlag sont tous deux réglés à zéro.
      - si securityLevel spécifie authNoPriv, alors authFlag est réglé à un et privFlag à zéro.
      - si securityLevel spécifie authPriv, alors authFlag est réglé à un et privFlag à un.
      - si la PDU est de la classe Unconfirmed, le reportableFlag est alors réglé à zéro.

- si la PDU est de la classe Confirmed, le reportableFlag est alors réglé à un.
- tous les autres bits de msgFlags sont réglés à zéro.

e) msgSecurityModel est réglé à la valeur de securityModel.

8) Si la PDU est de la classe Response ou de la classe Internal, alors :

a) Le modèle de sécurité spécifié est invoqué pour générer le message conformément à la primitive :

```
statusInformation = generateResponseMsg(
    IN   messageProcessingModel     -- modèle de traitement de messages SNMPv3
    IN   globalData                 -- msgGlobalData de l'étape 7
    IN   maxMessageSize             -- d'après msgMaxSize (étape 7c)
    IN   securityModel              -- comme déterminé à l'étape 7e
    IN   securityEngineID           -- valeur de snmpEngineID
    IN   securityName               -- au nom de ce principal
    IN   securityLevel              -- pour le message sortant
    IN   scopedPDU                  -- comme préparé à l'étape 6)
    IN   securityStateReference     -- comme déterminé à l'étape 2
   OUT   securityParameters         -- rempli par le module de sécurité
   OUT   wholeMsg                   -- message généré complet
   OUT   wholeMsgLength             -- longueur du message généré
  )
```

Si, au retour du modèle de sécurité, les statusInformation incluent une errorIndication, toutes les informations en antémémoire sur le message de demande en cours sont alors éliminées, et une errorIndication est retournée, afin qu'elle puisse être retournée à l'application appelante. Le traitement de message SNMPv3 est achevé.

b) Un résultat de SUCCÈS est retourné. Le traitement de message SNMPv3 est achevé.

9) Si la PDU est de la classe Confirmed ou de la classe Notification, alors :

a) Si la PDU est de la classe Unconfirmed, alors securityEngineID est réglé à la valeur du snmpEngineID de cette entité. Autrement, le snmpEngineID de l'entité cible est déterminé, d'une manière qui dépend de la mise en œuvre, éventuellement en utilisant transportDomain et transportAddress. La valeur du securityEngineID est réglée à la valeur du snmpEngineID de l'entité cible.

b) Le modèle de sécurité spécifié est invoqué pour générer le message conformément à la primitive :

```
statusInformation =
    generateRequestMsg(
    IN  messageProcessingModel   -- modèle de traitement de messages SNMPv3
    IN  globalData               -- msgGlobalData, d'après l'étape 7
    IN  maxMessageSize           -- d'après msgMaxSize à l'étape 7 c)
    IN  securityModel            -- comme fourni par l'appelant
    IN  securityEngineID         -- entité SNMP d'autorité d'après l'étape 9 a)
    IN  securityName             -- comme fourni par l'appelant
    IN  securityLevel            -- comme fourni par l'appelant
    IN  scopedPDU                -- comme préparé à l'étape 6
    OUT securityParameters       -- rempli par le module de sécurité
    OUT wholeMsg                 -- message généré complet
    OUT wholeMsgLength           -- longueur du message généré
    )
```

Si, au retour du modèle de sécurité, statusInformation inclut une errorIndication, le message est alors éliminé, et la errorIndication est retournée, afin qu'elle puisse être retournée à l'application appelante, et aucun autre traitement n'est effectué. Le traitement du message SNMPv3 est achevé.

c) Si la PDU est de la classe Confirmed, les informations sur le message sortant sont mises en antémémoire, et une stateReference spécifique de la mise en œuvre est créée. Les informations à mettre en antémémoire incluent les valeurs de :

- sendPduHandle
- msgID
- snmpEngineID
- securityModel
- securityName
- securityLevel
- contextEngineID

- contextName

d) Un résultat SUCCÈS est retourné. Le traitement du message SNMPv3 est achevé.

7.2 Préparer les éléments de données à partir d'un message SNMP entrant

Ce paragraphe décrit la procédure suivie pour extraire les données d'un message SNMPv3, et pour préparer les éléments de données requis pour la suite du traitement du message par le répartiteur de message.

1) Le message est passé du répartiteur de message conformément à la primitive de service abstrait :

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

2) Si le message reçu n'est pas la mise en série (conformément aux conventions de la [RFC3417]) d'une valeur de SNMPv3Message, le compteur snmpInASNParseErrs [RFC3418] est alors incrémenté, le message est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.

3) Les valeurs pour msgVersion, msgID, msgMaxSize, msgFlags, msgSecurityModel, msgSecurityParameters, et msgData sont extraites du message.

4) Si la valeur du composant msgSecurityModel ne correspondent pas à un securityModel pris en charge, le compteur snmpUnknownSecurityModels est alors incrémenté, le message est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.

5) Le securityLevel est déterminé à partir des bits authFlag et privFlag du composant msgFlags comme suit :
   a) Si authFlag n'est pas établi et si privFlag n'est pas établi, securityLevel est réglé à noAuthNoPriv.
   b) Si authFlag est établi et privFlag n'est pas établi, securityLevel est réglé à authNoPriv.
   c) Si authFlag est établi et privFlag est établi, securityLevel est réglé à authPriv.
   d) Si authFlag n'est pas établi et privFlag est établi, le compteur snmpInvalidMsgs est alors incrémenté, le message est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.
   e) Tous les autres bits dans msgFlags sont ignorés.

6) Le module de sécurité qui met en œuvre le modèle de sécurité comme spécifié par le composant securityModel est invoqué pour les services d'authentification et de confidentialité. Ceci est fait conformément à la primitive de service abstrait :

```
statusInformation =                   -- indication d'erreur ou succès ; OID de compteur d'erreur et valeur si erreur
processIncomingMsg(
   IN  messageProcessingModel         -- modèle de traitement de messages SNMPv3
   IN  maxMessageSize                 -- de l'entité SNMP envoyeuse
   IN  securityParameters             -- pour le message reçu
   IN  securityModel                  -- pour le message reçu
   IN  securityLevel                  -- niveau de sécurité
```

```
    IN  wholeMsg                    -- comme reçu du réseau
    IN  wholeMsgLength              -- longueur telle que reçue du réseau
   OUT securityEngineID             -- entité SNMP d'autorité
   OUT securityName                 -- identification du principal
   OUT scopedPDU                    -- charge utile du message (en texte source)
   OUT maxSizeResponseScopedPDU     -- taille maximum que l'envoyeur peut accepter
   OUT securityStateReference       -- référence aux informations d'état de sécurité nécessaires pour la réponse
   )
```

Si une errorIndication est retournée par le module de sécurité, alors :

a) Si statusInformation contient des valeurs pour une paire OID/valeur, la génération d'une PDU Report est alors tentée (voir l'étape 3 du paragraphe 7.1).

1) Si la scopedPDU a été retournée de processIncomingMsg, déterminer alors le contextEngineID, le contextName, et la PDU.

2) Les informations sur le message sont mises en antémémoire et une stateReference est créée (spécifique de la mise en œuvre). Les informations à mettre en antémémoire incluent les valeurs de :
   msgVersion,
   msgID,
   securityLevel,
   msgFlags,
   msgMaxSize,
   securityModel,
   maxSizeResponseScopedPDU,
   securityStateReference

3) Demander qu'une PDU Report soit préparée et envoyée, conformément à la primitive de service abstrait :

```
result =returnResponsePdu(          -- SUCCÈS ou ÉCHEC
    IN  messageProcessingModel      -- SNMPv3(3)
    IN  securityModel               -- le même que sur une demande entrante
    IN  securityName                -- de processIncomingMsg
    IN  securityLevel               -- le même que sur une demande entrante
    IN  contextEngineID             -- de l'étape 6 a) 1)
    IN  contextName                 -- de l'étape 6 a) 1)
    IN  pduVersion                  -- SNMPv2-PDU
    IN  PDU                         -- de l'étape 6 a) 1)
    IN  maxSizeResponseScopedPDU    -- de processIncomingMsg
    IN  stateReference              -- de l'étape 6 a) 2)
    IN  statusInformation           -- de processIncomingMsg
    )
```

b) Le message entrant est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.

7) La scopedPDU est analysée pour extraire le contextEngineID, le contextName et la PDU. Si une erreur d'analyse survient, le compteur snmpInASNParseErrs [RFC3418] est alors incrémenté, les informations d'état de sécurité sont éliminées, le message est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé. Traiter un type de PDU inconnu comme une erreur d'analyse est une option de la mise en œuvre.

8) La pduVersion est déterminée d'une manière qui dépend de la mise en œuvre. Pour SNMPv3, la pduVersion serait une PDU SNMPv2.

9) Le pduType est déterminé, d'une manière qui dépend de la mise en œuvre. Pour la [RFC3416], les pduType incluent :
   - GetRequest-PDU,
   - GetNextRequest-PDU,
   - GetBulkRequest-PDU,
   - SetRequest-PDU,
   - InformRequest-PDU,
   - SNMPv2-Trap-PDU,
   - Response-PDU,
   - Report-PDU.

10) Si le pduType est de la classe Response ou Internal, alors :
   a) La valeur du composant msgID est utilisée pour trouver les informations en antémémoire sur un message de demande correspondant en cours. Si un tel message Request en cours n'est pas trouvé, les informations d'état de sécurité sont alors

éliminées, le message est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.

b) sendPduHandle est restitué des informations en antémémoire.

Autrement, sendPduHandle est réglé à <none>, une valeur définie par la mise en œuvre.

11) Si le pduType est de la classe Internal, alors :

a) les statusInformation sont créées en utilisant le contenu de la PDU Report, d'une manière qui dépend de la mise en œuvre. Ces statusInformation seront transmises à l'application associée à la sendPduHandle.

b) Les données en antémémoire pour le message en cours, auxquelles se réfère stateReference, sont restituées. Si les valeurs de securityModel ou securityLevel diffèrent de celles en antémémoire, il est important de reconnaître que les PDU de la classe Internal livrées au niveau de sécurité de noAuthNoPriv ouvrent une fenêtre d'opportunité d'attaques en usurpation d'identité ou en répétition. Si le receveur de tels messages est conscient de ces risques, l'utilisation de tels messages non authentifiés est acceptable et peut fournir une fonction utile pour découvrir les identifiants de moteurs ou pour détecter des défauts de configuration chez les nœuds distants.
Lorsque les valeurs de securityModel ou securityLevel diffèrent de celles en antémémoire, une mise en œuvre peut conserver les informations de l'antémémoire sur le message Request en cours, en anticipation de la possibilité que la PDU de classe Internal reçue serait illégitime. Autrement, toutes les informations en antémémoire sur le message Request en cours sont éliminées.

c) Les informations d'état de sécurité pour ce message entrant sont éliminées.

d) stateReference est réglé à <none>.

e) Un résultat de SUCCÈS est retourné. Le traitement du message SNMPv3 est achevé.

12) Si le pduType est de la classe Response, alors :

a) Les données en antémémoire pour la demande en cours, auxquelles se réfère stateReference, sont restituées, incluant :
- snmpEngineID
- securityModel
- securityName
- securityLevel
- contextEngineID
- contextName

b) Si les valeurs extraites du message entrant diffèrent des données en antémémoire, toutes les informations de l'antémémoire sur le message Request en cours sont alors éliminées, le message entrant est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.
Lorsque les valeurs de securityModel ou securityLevel diffèrent de celles en antémémoire, une mise en œuvre peut conserver les informations en antémémoire sur le message Request en cours, en anticipation de la possibilité que la PDU de classe Response reçue serait illégitime.

c) Autrement, toutes les informations en antémémoire sur le message Request en cours sont éliminées, et la stateReference est réglée à <none>.

d) Un résultat de SUCCÈS est retourné. Le traitement du message SNMPv3 est achevé.

13) Si le pduType est de la classe Confirmed, alors :

a) Si la valeur de securityEngineID n'est pas égale à la valeur de snmpEngineID, les informations d'état de sécurité sont alors éliminées, toutes les informations en antémémoire sur ce message sont éliminées, le message entrant est éliminé sans autre traitement, et un résultat ÉCHEC est retourné. Le traitement du message SNMPv3 est achevé.

b) Les informations sur le message sont mises en antémémoire et une stateReference est créée (spécifique de la mise en œuvre). Les informations à mettre en antémémoire incluent les valeurs de :
msgVersion,
msgID,
securityLevel,
msgFlags,
msgMaxSize,
securityModel,
maxSizeResponseScopedPDU,
securityStateReference

c) Un résultat de SUCCÈS est retourné. Le traitement du message SNMPv3 est achevé.

14) Si le pduType est de la classe Unconfirmed, un résultat de SUCCÈS est retourné. Le traitement du message SNMPv3 est achevé.

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 le 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 se peut des précédentes RFC et projets. Il en résulte que des remerciements particuliers sont dus aux auteurs des projets précédents connus sous les noms de SNMPv2u et de SNMPv2* : Jeff Case (SNMP Research, Inc.), David Harrington (Cabletron Systems Inc.), David Levi (SNMP Research, Inc.), Keith McCloghrie (Cisco Systems), Brian O'Keefe (Hewlett Packard), Marshall T. Rose (Dover Beach Consulting), Jon Saperia (BGS Systems Inc.), Steve Waldbusser (International Network Services), Glenn W. Waters (Bell-Northern Research Ltd.)

10. Considérations pour la sécurité

Le répartiteur coordonne le traitement des messages pour fournir un niveau de sécurité aux messages de gestion et pour diriger les PDU SNMP à la ou aux applications SNMP appropriées.

Un modèle de traitement de message, et en particulier le v3MP défini dans le présent document, interagit au titre du traitement de message avec les modèles de sécurité dans le sous système de sécurité via les primitives d'interface de service abstraites définies dans la [RFC3411] et développées ci-dessus.

Le niveau de sécurité réellement fourni est principalement déterminé par la ou les mises en œuvre spécifiques de modèle de sécurité et la ou les mises en œuvre spécifiques d'application SNMP incorporées dans ce cadre. Les applications ont accès aux données qui ne sont pas sécurisées. Les applications devraient prendre des mesures raisonnables pour protéger les données de

la divulgation, et lorsque elles envoient des données sur le réseau, elles devraient respecter le niveau de sécurité et invoquer les services d'un modèle de contrôle d'accès lorsque elles appliquent le contrôle d'accès.

Les valeurs de l'élément msgID utilisé dans la communication entre les entités SNMP DOIVENT être choisies de façon à éviter les attaques en répétition. Il n'est pas nécessaire que les valeurs soient imprévisibles ; il est suffisant qu'elle ne soient pas répétées.
Lorsque les échanges sont effectués sur un réseau non sûr, cela offre à des tiers l'opportunité d'espionner ou répéter des messages lorsque un message d'un échange est classé au niveau de sécurité de noAuthNoPriv. Pour la plupart des échanges, tous les messages sont au même niveau de sécurité. Dans le cas où le message final est une PDU de classe Interne, ce message peut être délivré au niveau de noAuthNoPriv ou authNoPriv, indépendamment du niveau de sécurité des messages précédants. Les PDU de classe Interne délivrés au niveau de authNoPriv ne sont pas considérées poser de risque pour la sécurité. Les PDU de classe Interne délivrées au niveau de sécurité de noAuthNoPriv ouvrent une fenêtre d'opportunité d'espionnage ou d'attaque en répétition. Si le receveur d'un tel message est conscient de ces risques, l'utilisation de tels messages non authentifiés est acceptable et peut fournir une fonction utile pour la découverte des identifiants de moteurs ou pour détecter des fautes de configuration sur des nœuds distants.

Le présent document contient aussi un module de définition de MIB. Aucun des objets définis n'est inscriptible, et les informations qu'ils représentent ne sont pas réputées particulièrement sensibles. Cependant, si dans un environnement particulier elles étaient considérées comme sensibles, leur accès devrait être restreint par l'utilisation des modèles de sécurité et de contrôle d'accès configurés de façon appropriée.

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)

[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)

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

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

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

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

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

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

11.2 Références pour information

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

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

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

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

12. Adresse des éditeurs

Jeffrey Case SNMP Research, Inc. 3001 Kimberlin Heights Road Knoxville, TN 37920-9716 USA	David Harrington Enterasys Networks 35 Industrial Way PO Box 5005 Rochester, NH 03866-5005 USA	Randy Presuhn BMC Software, Inc. 2141 North First Street San Jose, CA 95131 USA	Bert Wijnen Lucent Technologies Schagen 33 3461 GL Linschoten Netherlands
téléphone : +1 423-573-1434 mél : case@snmp.com	+1 603-337-2614 dbh@enterasys.com	+1 408-546-1006 randy_presuhn@bmc.com	+31 348-680-485 bwijnen@lucent.com

13. Déclaration complète de droits de reproduction

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