Groupe de travail Réseau
Request for Comments : 3416
STD 62
RFC rendue obsolète : 1905
Catégorie : Norme

Traduction Claude Brière de L'Isle

Éditeur de cette version : R. Presuhn, BMC Software, Inc.
Auteurs de la version précédente :
J. Case, SNMP Research, Inc.
K. McCloghrie, Cisco Systems, Inc.
M. Rose, Dover Beach Consulting, Inc.
S. Waldbusser, International Network Services

décembre 2002

Version 2 des opérations de protocole pour le protocole simple de gestion de réseau (SNMP)

Résumé
Le présent document définit la version 2 des opérations de protocole pour le protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*). Il définit la syntaxe et les éléments de procédure pour l'envoi, la réception, et le traitement des PDU SNMP. Le présent document rend obsolète la RFC 1905.

Table des Matières

1. Introduction

Le cadre de gestion SNMP comporte au moment de la rédaction du présent texte cinq composantes majeures :

- Une architecture globale, décrite dans le STD 62, [RFC3411].
- Des mécanismes de description et de dénomination des objets et événements pour les besoins de la gestion. La première version de cette structure d'informations de gestion (SMI, *Structure of Management Information*) est appelée SMIv1 et elle est décrite dans le STD 16, [RFC1155], [RFC1212] et [RFC1215]. La seconde version, appelée SMIv2, est décrite dans le STD 58, [RFC2578], [RFC2579] et [RFC2580].

- Des protocoles de message pour le transfert des informations de gestion. La première version du protocole de message SNMP est appelée SNMPv1 et elle est décrite dans le STD 15, [RFC1157]. Une seconde version du protocole de message SNMP, qui n'est pas un protocole Internet en cours de normalisation, est appelée SNMPv2c et elle est décrite dans la [RFC1901] et dans le STD 62, [RFC3417]. La troisième version du protocole de message est appelée SNMPv3 et elle est décrite dans le STD 62, [RFC3417], [RFC3412] et [RFC3414].

- Des opérations de protocole pour accéder aux informations de gestion. Le premier ensemble d'opérations de protocole et de formats de PDU associées est décrit dans le STD 15, [RFC1157]. Un second ensemble d'opérations de protocole et de formats de PDU associées est décrit dans le présent document.

- Un ensemble d'applications fondamentales décrites dans le STD 62, [RFC3413] et le mécanisme de contrôle d'accès fondé sur la vue décrit dans le STD 62, [RFC3415].

Une introduction plus détaillée au cadre de gestion SNMP au moment de la rédaction se trouve dans la [RFC3410].

On accède aux objets gérés via un magasin virtuel d'informations, appelée la base de données d'informations de gestion (MIB, *Management Information Base*). Les objets de la MIB sont définis en utilisant les mécanismes définis dans la SMI.

Le présent document, version 2 des opérations de protocole pour le protocole simple de gestion de réseau, définit les opérations du protocole par rapport à l'envoi et la réception des PDU à transporter par le protocole de message.

2. Généralités

Les entités SNMP qui prennent en charge les applications de générateur de commandes ou de receveur de notification (traditionnellement appelées "gestionnaires") communiquent avec des entités SNMP qui prennent en charge des applications de réponse aux commandes ou de générateur de notifications (traditionnellement appelés "agents"). L'objet de ce protocole est le transport des informations et opérations de gestion.

2.1 Informations de gestion

Le terme "variable" se réfère à une instance de type d'objet non agrégeable défini conformément aux conventions établies dans la SMI [RFC2578] ou aux conventions textuelles fondées sur la SMI [RFC2579]. Le terme "liaison de variable" se réfère normalement à l'appariement du nom de la variable et de sa valeur associée. Cependant, si certaines sortes de conditions exceptionnelles surviennent durant le traitement d'une demande de restitution, une liaison de variable va apparier un nom et une indication de cette exception.

Une liste de liaison de variable est une simple liste de liaisons de variables.

Le nom d'une variable est un IDENTIFIANT D'OBJET qui est l'enchaînement de l'IDENTIFIANT D'OBJET du type d'objet correspondant avec un fragment d'IDENTIFIANT D'OBJET qui identifie l'instance. L'IDENTIFIANT D'OBJET du type d'objet correspondant est appelé le préfixe d'IDENTIFIANT D'OBJET de la variable.

2.2 Retransmission des demandes

Pour tous les types de demandes de ce protocole, le receveur est obligé, dans des circonstances normales, de générer et transmettre une réponse à l'origine de la demande. Qu'une demande doive être retransmise, si aucune réponse correspondante n'est reçue dans l'intervalle de temps approprié, est à la discrétion de l'application qui génère la demande. Cela va normalement dépendre de l'urgence de la demande. Cependant, une telle application doit agir de façon responsable par rapport à la fréquence et à la durée des retransmissions. Voir le BCP 41 [RFC2914] pour un exposé des principes pertinents de contrôle de l'encombrement.

2.3 Tailles des messages

La taille maximum d'un message SNMP est limitée au minimum de :
(1) la taille maximum de message que l'entité de destination SNMP peut accepter ; et,
(2) la taille maximum de message que l'entité SNMP de source peut générer.

La première peut s'apprécier sur la base du receveur ; et en son absence, elle est indiquée par le domaine de transport utilisé lors de l'envoi du message. La seconde est imposée par des contraintes locales spécifiques de la mise en œuvre.

Chaque transposition de transport pour SNMP indique la taille de message minimum qu'une mise en œuvre SNMP doit être capable de produire ou consommer. Bien que les mises en œuvre soient invitées à prendre en charge de plus grandes valeurs chaque fois que possible, une mise en œuvre conforme ne doit jamais générer de messages plus grands que ce qui est autorisé par l'entité SNMP receveuse.

Un des buts de la PDU GetBulkRequest, spécifiée dans le présent protocole, est de minimiser le nombre d'échanges de protocole nécessaires pour restituer une grande quantité d'informations de gestion. À ce titre, ce type de PDU permet à une entité SNMP qui prend en charge des applications de générateur de commandes de demander que la réponse soit aussi grande que possible étant données les contraintes de taille de message. Ces contraintes incluent les limites de taille des messages que peut générer l'entité SNMP qui prend en charge les applications de réponse aux commandes, et que peut recevoir l'entité SNMP qui prend en charge les applications de générateur de commandes.

Cependant, il est possible que de tels messages de taille maximum soient plus grands que la MTU du chemin traversé par les messages à travers le réseau. Dans une telle situation, de tels messages sont soumis à la fragmentation. La fragmentation est généralement considérée comme dommageable [FRAG], parce que entre autres problèmes, elle conduit à une diminution de la fiabilité du transfert des messages. Et donc, une entité SNMP qui envoie une PDU GetBulkRequest doit veiller à régler ses paramètres en conséquence, de façon à réduire le risque de fragmentation. En particulier, dans des conditions de tensions sur le réseau, seules de petites valeurs devraient être utilisées pour max-repetitions.

2.4 Transpositions de transport

Il est important de noter que l'échange des messages SNMP exige seulement un service non fiable de datagrammes, chaque message étant entièrement contenu dans un seul datagramme de transport et ce, de façon indépendante. Les règles spécifiques de transposition de transport et de codage sont spécifiée ailleurs [RFC3417]. Cependant, la transposition préférée est l'utilisation du protocole de datagramme d'utilisateur [RFC0768].

2.5 Transpositions de type de données SMIv2

La SMIv2 [RFC2578] définit onze types de base (INTEGER *(entier)*, OCTET STRING *(chaîne d'octets)*, OBJECT IDENTIFIER *(identifiant d'objet)*, Integer32, IpAddress, Counter32, Gauge32, Unsigned32, TimeTicks, Opaque, Counter64) et la construction BITS. Les types de base de SMIv2 sont transposés dans les types choisis correspondants dans les choix SimpleSyntax et ApplicationSyntax de la définition ASN.1 du protocole SNMP. Noter que les types de base INTEGER et Integer32 de SMIv2 sont transposés dans le type de choix de valeur d'entier du choix SimpleSyntax. De même, les types de base Gauge32 et Unsigned32 de SMIv2 sont transposés dans le type de choix unsigned-integer-value *(valeur d'entier non signé)* du choix ApplicationSyntax.

La construction BITS de SMIv2 est transposée dans le type de choix string-value du choix SimpleSyntax. Une valeur BITS est codée comme une CHAÎNE D'OCTETS, dans laquelle tous les bits désignés dans (la définition de) la chaîne binaire, en commençant par le premier bit et en continuant jusqu'au dernier bit, sont placés du bit 8 (bit de plus fort poids) au bit 1 (bit de moindre poids) du premier octet, suivis par les bits 8 à 1 de chaque octet suivant à son tour, suivis par autant de bits que nécessaire pour l'octet final suivant, en commençant par le bit 8. Les bits restants, s'il en est, de l'octet final sont mis à zéro à leur génération et ignorés à réception.

3. Définitions

La syntaxe de PDU est définie en utilisant la notation ASN.1 [ASN1].

```
DÉFINITIONS SNMPv2-PDU ::= DÉBUT

ObjectName ::= IDENTIFIANT D'OBJET

ObjectSyntax ::= CHOIX {
        simple      SimpleSyntax,
        application-wide    ApplicationSyntax }

SimpleSyntax ::= CHOIX {
        integer-value       ENTIER (-2 147 483 648 à 2 147 483 647),
        string-value        CHAINE D'OCTETS (TAILLE (0 à 65 535)),
        objectID-value      IDENTIFIANT D'OBJET }
```

```
ApplicationSyntax ::= CHOIX {
        ipAddress-value             IpAddress,
        counter-value               Counter32,
        timeticks-value             TimeTicks,
        arbitrary-value             Opaque,
        big-counter-value           Counter64,
        unsigned-integer-value      Unsigned32 }

IpAddress ::= [APPLICATION 0] CHAINE D'OCTETS IMPLICITE (TAILLE (4))

Counter32 ::= [APPLICATION 1] ENTIER IMPLICITE (0 à 4 294 967 295)

Unsigned32 ::= [APPLICATION 2] ENTIER IMPLICITE (0 à 4 294 967 295)

Gauge32 ::= Unsigned32

TimeTicks ::= [APPLICATION 3] ENTIER IMPLICITE (0 à 4 294 967 295)

Opaque ::= [APPLICATION 4] CHAINE D'OCTETS IMPLICITE

Counter64 ::= [APPLICATION 6] ENTIER IMPLICITE (0 à 18 446 744 073 709 551 615)

-- Unités de données de protocole

PDU ::= CHOIX {
        get-request             PDU GetRequest,
        get-next-request        PDU GetNextRequest,
        get-bulk-request        PDU GetBulkRequest,
        response                PDU Response,
        set-request             PDU SetRequest,
        inform-request          PDU InformRequest,
        snmpV2-trap             PDU SNMPv2-Trap,
        report                  PDU Report }

-- PDU

PDU GetRequest ::= [0] PDU IMPLICITE

PDU GetNextRequest ::= [1] PDU IMPLICITE

PDU Response ::= [2] PDU IMPLICITE

PDU SetRequest ::= [3] PDU IMPLICITE

-- [4] est obsolète

PDU GetBulkRequest ::= [5] BulkPDU IMPLICITE

PDU InformRequest ::= [6] PDU IMPLICITE

PDU SNMPv2-Trap ::= [7] PDU IMPLICITE

-- L'usage et la sémantique précise de la PDU Report ne sont pas définis dans ce document. Tout cadre administratif de
    SNMP utilisant cette PDU doit définir son usage et sa sémantique.

PDU Report ::= [8] PDU IMPLICITE

max-bindings ENTIER ::= 2 147 483 647

PDU ::= SEQUENCE { request-id ENTIER (-214 783 648 à 214 783 647),

error-status        -- parfois ignoré
```

```
            ENTIER {
                noError(0),
                tooBig(1),
                noSuchName(2),      -- pour la compatibilité des mandataires
                badValue(3),        -- pour la compatibilité des mandataires
                readOnly(4),        -- pour la compatibilité des mandataires
                genErr(5),
                noAccess(6),
                wrongType(7),
                wrongLength(8),
                wrongEncoding(9),
                wrongValue(10),
                noCreation(11),
                inconsistentValue(12),
                resourceUnavailable(13),
                commitFailed(14),
                undoFailed(15),
                authorizationError(16),
                notWritable(17),
                inconsistentName(18)
            },

    error-index             -- parfois ignoré
            ENTIER (0 à max-bindings),

    variable-bindings       -- les valeurs sont parfois ignorées
            VarBindList
        }

    BulkPDU ::=                 -- doit avoir une structure identique à la PDU
        SEQUENCE {
            request-id      ENTIER (-214 783 648 à 214 783 647),
            non-repeaters   ENTIER (0 à max-bindings),
            max-repetitions ENTIER (0 à max-bindings),
            variable-bindings       -- les valeurs sont ignorées
                VarBindList
        }

    -- liaison de variables

    VarBind ::= SEQUENCE {
            nom  ObjectName,

            CHOIX {
                valeur          ObjectSyntax,
                unSpecified     NUL,    -- dans les demandes de restitution

                                -- exceptions dans les réponses
                noSuchObject    [0] NUL IMPLICITE,
                noSuchInstance  [1] NUL IMPLICITE,
                endOfMibView    [2] NUL IMPLICITE.
            }
        }

    -- liste de liaisons de variables

    VarBindList ::= SEQUENCE (TAILLE (0 à max-bindings)) DE VarBind

    FIN
```

4. Spécification du protocole

4.1 Constructions communes

La valeur du champ request-id dans une PDU Response prend la valeur du champ request-id dans la PDU de demande à laquelle elle répond. Par l'utilisation de la valeur du request-id, une application peut distinguer les demandes en cours (qui peuvent être nombreuses) et corréler par là les réponses entrantes avec les demandes en cours. Dans les cas d'utilisation d'un service de datagrammes non fiable, l'identifiant de demande donne un moyen simple pour identifier les messages dupliqués par le réseau. L'utilisation du même identifiant de demande sur la retransmission d'une demande permet à la réponse de satisfaire aussi bien la transmission originale que la retransmission de la demande. Cependant, afin de calculer le délai d'aller retour de la transmission et du traitement d'une transaction de demande-réponse, l'application doit utiliser une valeur d'identifiant de demande différente sur une demande retransmise. Il est recommandée d'utiliser cette dernière stratégie dans la majorité des situations.

Une valeur différente de zéro dans le champ État d'erreur dans une PDU de réponse est utilisée pour indiquer qu'une erreur est intervenue qui a empêché le traitement de la demande. Dans ces cas, une valeur différente de zéro du champ Indice d'erreur de la PDU de réponse donnera des informations supplémentaires en identifiant sur la liste la liaison de variable qui a causé l'erreur. Une liaison de variable est identifiée par sa valeur d'indice. La première liaison de variable dans une liste de liaisons de variables est l'indice un, la seconde est l'indice deux, etc.

SNMP limite les valeurs d'IDENTIFIANT D'OBJET à un maximum de 128 sous identifiants, et chaque sous identifiant a une valeur maximum de 2**32-1 *(2^{32} -1)*.

4.2 Traitement des PDU

Dans les éléments de procédure ci-dessous, tout champ d'une PDU qui n'est pas référencé par la procédure pertinente est ignoré par l'entité SNMP receveuse. Cependant, tous les composants d'une PDU, y compris ceux dont les valeurs sont ignorées par l'entité SNMP receveuse, doivent avoir une syntaxe et un codage ASN.1 valides. Par exemple, certaines PDU (par exemple, la PDU GetRequest) ne sont concernées que par le nom d'une variable et pas par sa valeur. Dans ce cas, la portion valeur de la liaison de variable est ignorée par l'entité SNMP receveuse. La valeur unSpecified est définie pour être utilisée comme portion de valeur de telles liaisons.

Lors de la génération d'une communication de gestion, "l'enveloppe" de message pour encapsuler la PDU est générée conformément aux "Éléments de procédure" du cadre administratif utilisé. La définition de "max-bindings" impose une limite supérieure au nombre de liaisons de variables. En pratique, la taille d'un message est aussi limitée par des contraintes sur la taille maximum de message. Une mise en œuvre conforme doit prendre en charge autant de liaisons de variables dans une PDU ou BulkPDU qu'il en tient dans la limite globale de taille maximum de message du moteur SNMP, mais pas plus de 2 147 483 647 liaisons de variables.

À réception d'une communication de gestion, les "Éléments de procédure" du cadre administratif utilisé sont suivis, et si ces procédures indiquent que l'opération contenue dans le message est à effectuer en local, ces procédures indiquent alors aussi la vue de MIB qui est visible pour l'opération.

4.2.1 PDU GetRequest

Une PDU GetRequest est généré et transmise à la demande d'une application.

À réception d'une PDU GetRequest, l'entité SNMP receveuse traite chaque liaison de variable dans la liste de liaisons de variables pour produire une PDU Response. Tous les champs de la PDU Response ont les mêmes valeurs que les champs correspondants de la demande reçue excepté comme indiqué ci-dessous. Chaque liaison de variable est traitée comme suit :

(1) Si le nom de la liaison de variable correspond exactement au nom d'une variable accessible par cette demande, le champ de valeur de la liaison de variable est réglé à la valeur de la variable désignée.

(2) Autrement, si le nom de la liaison de variable n'a pas un préfixe IDENTIFIANT D'OBJET qui correspond exactement au préfixe IDENTIFIANT D'OBJET de toute (potentielle) variable accessible par cette demande, son champ de valeur est alors réglé à "noSuchObject".

(3) Autrement, le champ de valeur de la liaison de variable est réglé à "noSuchInstance".

Si le traitement de toute liaison de variable échoue pour une raison autre que celles énumérées ci-dessus, la Response-PDU

est alors reformatée avec les mêmes valeurs dans ses champs request-id et variable-bindings que dans la PDU GetRequest reçue, avec la valeur de son champ État d'erreur réglé à "genErr", et la valeur de son champ indice d'erreur réglé à l'indice de la liaison de variable en échec.

Autrement, la valeur du champ état d'erreur de la PDU Response est réglé à "noError", et la valeur de son champ indice d'erreur est zéro.

La PDU Response générée est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à une contrainte locale et à la taille maximum de message de l'origine, il est transmis à l'origine de la PDU GetRequest.
Autrement, une PDU Response de remplacement est générée. Cette PDU Response de remplacement est formatée avec la même valeur dans son champ d'identifiant de demande que celle de la PDU GetRequest reçue, avec la valeur de son champ État d'erreur réglé à "tropGros", la valeur de son champ d'indice d'erreur réglé à zéro, et un champ vide de liaisons de variables. Cette PDU Response de remplacement est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à une contrainte locale et à la taille maximum de message de l'origine, il est transmis à l'origine de la PDU GetRequest. Autrement, le compteur snmpSilentDrops [RFC3418] est incrémenté et le message résultant est éliminé.

4.2.2 PDU GetNextRequest

Une PDU GetNextRequest est générée et transmise à la demande d'une application.

À réception d'une PDU GetNextRequest, l'entité SNMP receveuse traite chaque liaison de variable de la liste de variable-binding pour produire une PDU Response. Tous les champs de la PDU Response ont les mêmes valeurs que les champs correspondants de la demande reçue excepté comme indiqué ci-dessous. Chaque liaison de variable est traitée comme suit :

(1) On localise la variable qui est dans la liste ordonnée lexicographiquement des noms de toutes les variables qui sont accessibles par cette demande et dont le nom est le premier successeur lexicographique du nom de la liaison de variable dans la PDU GetNextRequest entrante. Les champs de nom et de valeur de la liaison de variable correspondante dans la PDU Response sont mis au nom et à la valeur de la variable localisée.

(2) Si le nom de la liaison de variable demandée ne précède pas lexicographiquement le nom d'une variable accessible par cette demande, c'est-à-dire, si il n'a pas de successeur lexicographique, la liaison de variable correspondante produite dans la PDU Response a son champ de valeur réglé à "endOfMibView", et son champ de nom est réglé au nom de la liaison de variable de la demande.

Si le traitement d'une liaison de variable échoue pour une raison autre que celles énumérées ci-dessus, la PDU Response est alors reformatée avec les mêmes valeurs dans ses champs request-id et variable-bindings que dans la PDU GetNextRequest reçue, avec la valeur de son champ error-status réglé à "genErr", et la valeur de son champ error-index réglée à l'indice de la liaison de variable qui a échoué.

Autrement, la valeur du champ error-status de la PDU Response est réglé à "noError", et la valeur de son champ error-index est à zéro.

La PDU Response générée est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à une contrainte locale et à la taille maximum de message du générateur, il est transmis au générateur de la PDU GetNextRequest.

Autrement, une PDU Response de remplacement est générée. Cette PDU Response de remplacement est formatée avec les mêmes valeurs que dans le champ request-id de la PDU GetNextRequest reçue, avec la valeur de son champ error-status réglé à "tooBig", la valeur de son champ error-index réglée à zéro, et un champ variable-bindings vide. Cette PDU Response de remplacement est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à une contrainte locale et à la taille maximum de message du générateur, il est transmis au générateur de la PDU GetNextRequest. Autrement, le compteur snmpSilentDrops [RFC3418] est incrémenté et le message résultant est éliminé.

4.2.2.1 Exemple de traversée de tableau

Une utilisation importante de la PDU GetNextRequest est la traversée des tableaux conceptuels d'informations au sein d'une MIB. La sémantique de ce type de demande, ainsi qu'avec la méthode d'identification des instances individuelles des objets dans la MIB, fournit l'accès aux objets en rapport dans la MIB comme si ils bénéficiaient d'une organisation en tableaux.

Dans le scénario d'échange de protocole ci-dessous, une application restitue l'adresse physique dépendante du support et le

type de transposition d'adresse pour chaque entrée dans le tableau de traduction d'adresse IP du réseau au support [RFC1213] d'un certain élément de réseau. Elle restitue aussi la valeur de sysUpTime [RFC3418], à laquelle les transpositions ont eu lieu. Supposons que le tableau de réseau à support du répondeur de commandes ait trois entrées :

Numéro d'interface	Adresse réseau	Adresse physique	Type
1	10.0.0.51	00:00:10:01:23:45	statique
1	9.2.3.4	00:00:10:54:32:10	dynamique
2	10.0.0.15	00:00:10:98:76:54	dynamique

L'entité SNMP qui prend en charge une application de générateur de commandes commence par envoyer une PDU GetNextRequest contenant les valeurs d'IDENTIFIANT D'OBJET indiquées comme noms de variable demandés :

GetNextRequest (sysUpTime, ipNetToMediaPhysAddress, ipNetToMediaType)

L'entité SNMP qui prend en charge une application de répondeur de commande répond par une PDU Response :

Response ((sysUpTime.0 = "123456"),
(ipNetToMediaPhysAddress.1.9.2.3.4 = "000010543210"),
(ipNetToMediaType.1.9.2.3.4 = "dynamic"))

L'entité SNMP qui prend en charge l'application de générateur de commande continue avec :

GetNextRequest (sysUpTime, ipNetToMediaPhysAddress.1.9.2.3.4, ipNetToMediaType.1.9.2.3.4)

L'entité SNMP qui prend en charge l'application de répondeur de commande répond par :

Response ((sysUpTime.0 = "123461"),
(ipNetToMediaPhysAddress.1.10.0.0.51 = "000010012345"),
(ipNetToMediaType.1.10.0.0.51 = "static"))

L'entité SNMP qui prend en charge l'application de générateur de commande continue avec :

GetNextRequest (sysUpTime, ipNetToMediaPhysAddress.1.10.0.0.51, ipNetToMediaType.1.10.0.0.51)

L'entité SNMP qui prend en charge l'application de répondeur de commande répond par :

Response ((sysUpTime.0 = "123466"),
(ipNetToMediaPhysAddress.2.10.0.0.15 = "000010987654"),
(ipNetToMediaType.2.10.0.0.15 = "dynamic"))

L'entité SNMP qui prend en charge l'application de générateur de commande continue avec :

GetNextRequest (sysUpTime, ipNetToMediaPhysAddress.2.10.0.0.15, ipNetToMediaType.2.10.0.0.15)

Comme il n'y a plus d'autre entrée dans le tableau, l'entité SNMP qui prend en charge l'application de répondeur de commande répond avec les variables qui suivent dans l'ordre lexicographique des noms d'objets accessibles, par exemple :

Response ((sysUpTime.0 = "123471"),
(ipNetToMediaNetAddress.1.9.2.3.4 = "9.2.3.4"),
(ipRoutingDiscards.0 = "2"))

Noter comment, ayant atteint la fin de la colonne pour ipNetToMediaPhysAddress, la seconde liaison de variable provenant de l'application de répondeur de commande est maintenant "revenue" à la première rangée de la colonne suivante. De plus, noter comment, ayant atteint la fin de ipNetToMediaTable pour la troisième liaison de variable, l'application de répondeur de commandes a répondu par le prochain objet disponible, qui est en dehors de ce tableau. Cette réponse signale la fin du tableau à l'application de générateur de commande.

4.2.3 PDU GetBulkRequest

Une PDU GetBulkRequest est générée et transmise à la demande d'une application. L'objet de la PDU GetBulkRequest est de demander le transfert d'une quantité de données potentiellement grande, incluant, mais sans s'y limiter, la restitution efficace et rapide de grands tableaux.

À réception d'une PDU GetBulkRequest, l'entité SNMP receveuse traite chaque liaison de variable de la liste des liens de variable pour produire une PDU Response avec son champ Identifiant de demande qui a la même valeur que dans la demande.

Pour le type GetBulkRequest, la réussite du traitement de chaque liaison de variable dans la demande génère zéro, une ou plusieurs liaisons de variables dans la PDU Response. C'est-à-dire, la transposition biunivoque entre les liaisons de variables des types de PDU GetRequest, GetNextRequest, et SetRequest et les PDU Response résultantes ne s'appliquent pas pour la transposition entre les liaisons de variables d'une PDU GetBulkRequest et la PDU Response résultante.

Les valeurs des champs non-repeaters et max-repetitions dans la demande spécifient le traitement demandé. Une liaison de variable dans la PDU Response est exigée pour les N premières liaisons de variables dans la demande et M liaisons de variables sont exigées pour chacune des R liaisons de variables restantes dans la demande. Par conséquent, le nombre total de liaisons de variables demandées communiqué par la demande est donné par N + (M * R), où N est le minimum de : a) la valeur du champ non-repeaters dans la demande et, b) le nombre de liaisons de variables dans la demande ; M est la valeur du champ max-repetitions dans la demande ; et R est le maximum de : a) le nombre de liaisons de variables dans la demande - N, et b) zéro.

L'entité SNMP receveuse produit une PDU Response avec jusqu'au nombre total de liaisons de variables demandées communiqué par la demande. L'identifiant de demande devra avoir la même valeur que la PDU GetBulkRequest reçue.

Si N est supérieur à zéro, les premières des (N)èmes liaisons de variables de la PDU Response sont chacune produites comme suit :

(1) On localise la variable dont le nom est le premier successeur lexicographique du nom de la liaison de variable dans la liste ordonnée lexicographiquement des noms de toutes les variables qui sont accessibles par cette demande dans la PDU GetBulkRequest entrante. Les champs correspondants de nom et de valeur de la liaison de variable dans la PDU Response sont réglés au nom et valeur de la variable localisée.

(2) Si le nom de la liaison de variable demandée ne précède lexicographiquement le nom d'aucune variable accessible par cette demande, c'est-à-dire, si il n'y a pas de successeur lexicographique, la liaison de variable correspondante produite dans la PDU Response a son champ Valeur réglé à "endOfMibView", et son champ Nom réglé au nom de la liaison de variable dans la demande.

Si M et R ne sont pas à zéro, la (N + 1)ème liaison de variable et les suivantes de la PDU Response sont chacune produites de manière similaire. Pour chaque itération i, telle que i est supérieur à zéro et inférieur ou égal à M, et pour chaque variable r répétée, telle que r supérieur à zéro et inférieur ou égal à R, la (N + ((i-1) * R) + r)ème liaison de variable de la PDU Response est produite comme suit :

(1) La variable qui est dans la liste ordonnée lexicographiquement des noms de toutes les variables qui sont accessibles par cette demande et dont le nom est le (i)ème successeur lexicographique du nom de la (N + r)ème liaison de variable dans la PDU GetBulkRequest entrante est localisée et les champs Nom et Valeur de la liaison de variable sont réglés au nom et à la valeur de la variable localisée.

(2) Si il n'y a pas de (i)ème successeur lexicographique, alors la liaison de variable correspondante produite dans la PDU Response a son champ Valeur réglé à "endOfMibView", et son champ Nom est réglé soit au dernier successeur lexicographique, soit, si il n'y a pas de successeur lexicographique, au nom de la (N + r)ème liaison de variable dans la demande.

Bien que le nombre maximum de liaisons de variables dans la PDU Response soit limité à N + (M * R), la réponse peut être générée avec un nombre of liaisons de variables inférieur (éventuellement zéro) pour une des trois raisons suivantes.

(1) Si la taille du message qui encapsule la PDU Response contenant le nombre demandé de liaisons de variables serait supérieur à une contrainte locale ou à la taille maximum de message du générateur, la réponse est alors générée avec un nombre moindre de liaisons de variables. Ce moindre nombre est l'ensemble ordonné de liaisons de variables avec certaines des liaisons de variables retirées à la fin de l'ensemble, de telle sorte que la taille du message encapsulant la PDU Response soit approximativement égal mais pas supérieur à une contrainte locale ou à la taille maximum de message du générateur. Noter que le nombre de liaisons de variables retiré n'a pas de relation avec les valeurs de N, M, ou R.

(2) La réponse peut aussi être générée avec un moindre nombre de liaisons de variables si pour une certaine valeur d'itérations i, tel que i soit supérieur à zéro et inférieur ou égal à M, toutes les liaisons de variables générées ont le

champ Valeur réglé à "endOfMibView". Dans ce cas, les liaisons de variables peuvent être tronquées après la (N+(i*R))ème liaison de variable.

(3) Dans le cas où le traitement d'une demande avec de nombreuses répétitions exige un temps de traitement significativement supérieur à celui d'une demande normale, une application de répondeur de commandes peut terminer la demande avec moins que le nombre complet de répétitions, pourvu qu'au moins une répétition soit achevée.

Si le traitement d'une liaison de variable échoue pour une raison autre que celles données ci-dessus, la PDU Response est alors reformatée avec les mêmes valeurs dans ses champs request-id et variable-bindings que ceux de la PDU GetBulkRequest reçue, avec la valeur de son champ error-status réglée à "genErr", et la valeur de son champ error-index réglée à l'indice de la liaison de variable dans la demande originale qui correspond à la liaison de variable en échec.

Autrement, la valeur du champ error-status de la PDU Response est réglé à "noError", et la valeur de son champ error-index est à zéro.

La PDU Response générée (éventuellement avec un champ variable-bindings vide) est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à une contrainte locale et à la taille maximum de message du générateur, il est transmis au générateur de la PDU GetBulkRequest. Autrement, le compteur snmpSilentDrops [RFC3418] est incrémenté et le message résultant est éliminé.

4.2.3.1 Autre exemple de traversée de tableau

Cet exemple montre comment la PDU GetBulkRequest peut être utilisée comme solution de remplacement à la PDU GetNextRequest. La même traversée du tableau IP réseau à support que montré au paragraphe 4.2.2.1 est réalisée avec moins d'échanges.

L'entité SNMP qui prend en charge l'application de générateur de commande commence par envoyer une PDU GetBulkRequest avec la valeur modeste de max-repetitions de 2, et contenant les valeurs d'IDENTIFIANT D'OBJET indiquées comme noms de variables demandées :

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
            ( sysUpTime, ipNetToMediaPhysAddress, ipNetToMediaType )
```

L'entité SNMP qui prend en charge l'application de répondeur de commande répond par une PDU Response :

```
Response (( sysUpTime.0 =  "123456" ),
        ( ipNetToMediaPhysAddress.1.9.2.3.4 = "000010543210" ),
      ( ipNetToMediaType.1.9.2.3.4 =  "dynamic" ),
        ( ipNetToMediaPhysAddress.1.10.0.0.51 = "000010012345" ),
      ( ipNetToMediaType.1.10.0.0.51 =  "static" ))
```

L'entité SNMP qui prend en charge l'application de générateur de commande continue avec :

```
GetBulkRequest [ non-repeaters = 1, max-repetitions = 2 ]
            ( sysUpTime, ipNetToMediaPhysAddress.1.10.0.0.51, ipNetToMediaType.1.10.0.0.51 )
```

L'entité SNMP qui prend en charge l'application de répondeur de commande répond avec :

```
Response (( sysUpTime.0 =  "123466" ),
        ( ipNetToMediaPhysAddress.2.10.0.0.15 = "000010987654" ),
        ( ipNetToMediaType.2.10.0.0.15 = "dynamic" ),
        ( ipNetToMediaNetAddress.1.9.2.3.4 = "9.2.3.4" ),
      ( ipRoutingDiscards.0 =  "2" ))
```

On notera, comme dans le premier exemple, comment les liaisons de variables dans la réponse indiquent que la fin du tableau a été atteinte. La quatrième liaison de variable le fait en retournant des informations de la prochaine colonne disponible ; la cinquième liaison de variable le fait en retournant des informations du premier objet disponible qui suit lexicographiquement dans le tableau. Cette réponse signale la fin du tableau à l'application de générateur de commandes.

4.2.4 PDU de réponse

Une PDU Response n'est générée par une entité SNMP qu'à réception d'une PDU GetRequest, GetNextRequest, GetBulkRequest, SetRequest, ou InformRequest, comme décrit dans le présent document.

Si le champ error-status de la PDU Response n'est pas zéro, les champs Valeur des liaisons de variables dans la liste de liaisons de variable sont ignorés.

Si les deux champs error-status et error-index de la PDU Response ne sont pas à zéro, la valeur du champ error-index est alors l'indice de la liaison de variable (dans la liste variable-binding de la demande correspondante) pour lequel la demande a échoué. La première liaison de variable dans la liste de liens de variable d'une demande est l'indice un, la seconde a l'indice deux, etc.

Une entité SNMP conforme qui prend en charge une application de générateur de commandes doit être capable de recevoir et traiter correctement une PDU Response avec un champ error-status égal à "noSuchName", "badValue", ou "readOnly". (voir les paragraphes 1.3 et 4.3 de la [RFC2576].)

À réception d'une PDU Response, l'entité SNMP receveuse présente son contenu à l'application qui a généré la demande avec la même valeur de request-id. Pour les détails, voir la [RFC3412].

4.2.5 PDU SetRequest

Une PDU SetRequest est générée et transmise à la demande d'une application.

À réception d'une PDU SetRequest, l'entité SNMP receveuse détermine la taille d'un message encapsulant une PDU Response ayant les mêmes valeurs dans ses champs request-id et variable-bindings que la PDU SetRequest reçue, et les plus grandes tailles possibles des champs error-status et error-index. Si la taille de message déterminée est supérieure à une contrainte locale ou à la taille maximum de message du générateur, une PDU Response de remplacement est alors générée, transmise au générateur de la PDU SetRequest, et le traitement de la PDU SetRequest se termine immédiatement après. Cette PDU Response de remplacement est formatée avec les mêmes valeurs dans son champ request-id que la PDU SetRequest reçue, avec la valeur de son champ error-status réglée à "tooBig", la valeur de son champ error-index réglée à zéro, et un champ variable-bindings vide. Cette PDU Response de remplacement est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à la contrainte locale et à la taille maximum de message du générateur, elle est transmise au générateur de la PDU SetRequest. Autrement, le compteur snmpSilentDrops [RFC3418] est incrémenté et le message résultant est éliminé. De toutes façons, le traitement de la PDU SetRequest se termine.

Autrement, l'entité SNMP receveuse traite chaque liaison de variable de la liste de variable-binding pour produire une PDU Response. Tous les champs de la PDU Response ont la même valeur que les champs correspondants de la demande reçue, sauf comme indiqué ci-dessous.

Les liaisons de variables sont traitées conceptuellement comme une opération en deux phases. Dans la première phase, chaque liaison de variable est validée ; si toutes les validations sont réussies, chaque variable est alors altérée dans la seconde phase. Bien sûr, les mises en œuvre ont la liberté de traiter la première, ou la seconde, ou les deux, de ces phases conceptuelles comme plusieurs phases de mise en œuvre. Bien sûr, de telles phases de mise en œuvre multiples peuvent être nécessaires dans certains cas pour assurer la cohérence.

Les validations suivantes sont effectuées dans la première phase sur chaque liaison de variable jus qu'à ce qu'elles soient toutes réussies, ou qu'une échoue :

(1) Si le nom de la liaison de variable spécifie une variable existante ou non à laquelle cette demande est ou serait refusé l'accès parce qu'elle n'est/serait pas dans la vue de MIB appropriée, la valeur du champ error-status de la PDU Response est alors réglée à "noAccess", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(2) Autrement, si il n'y a pas de variable qui partage le même préfixe IDENTIFIANT D'OBJET que le nom de la liaison de variable, et qui est capable d'être créée ou modifiée quelle que soit la nouvelle valeur spécifiée, alors la valeur du champ error-status de la PDU Response est réglée à "notWritable", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(3) Autrement, si le champ Valeur de la liaison de variable spécifie, conformément au langage ASN.1, un type qui n'est pas cohérent avec celui requis pour toutes les variables qui partagent le même préfixe d'IDENTIFIANT D'OBJET que le nom de la liaison de variable, alors la valeur du champ error-status de la PDU Response est réglée à "wrongType", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(4) Autrement, si le champ Valeur de la liaison de variable spécifie, conformément au langage ASN.1, une longueur qui n'est pas cohérente avec celle requise pour toutes les variables qui partagent le même préfixe d'IDENTIFIANT

D'OBJET que le nom de la liaison de variable, alors la valeur du champ error-status de la PDU Response est réglée à "wrongLength", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(5) Autrement, si le champ Valeur de la liaison de variable contient un codage ASN.1 qui n'est pas cohérent avec l'étiquette ASN.1 de ce champ, alors la valeur du champ error-status de la PDU Response est réglée à "wrongEncoding", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec. (Noter que toutes les stratégies de mise en œuvre ne vont pas générer cette erreur.)

(6) Autrement, si le champ Valeur de la liaison de variable spécifie une valeur qui ne pourrait en aucune circonstance être allouée à la variable, alors la valeur du champ error-status de la PDU Response est réglée à "wrongValue", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(7) Autrement, si le nom de la liaison de variable spécifie une variable qui n'existe pas et ne pourrait pas être créée (même si certaine variables partageant le même préfixe d'IDENTIFIANT D'OBJET pourraient dans certaines circonstances être capables d'être créées) alors la valeur du champ error-status de la PDU Response est réglée à "noCreation", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(8) Autrement, si le nom de la liaison de variable spécifie une variable qui n'existe pas mais ne peut pas être créée dans les circonstances présentes (même si elle pourrait être créée dans d'autres circonstances) alors la valeur du champ error-status de la PDU Response est réglée à "inconsistentName", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(9 Autrement, si le nom de la liaison de variable spécifie une variable qui existe mais ne peut pas être modifiée quelle que soit la nouvelle valeur spécifiée, alors la valeur du champ error-status de la PDU Response est réglée à "notWritable", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(10) Autrement, si le champ Valeur de la liaison de variable spécifie une valeur qui pourrait dans d'autres circonstances être portée par la variable, mais est présentement incohérente ou par ailleurs impossible à allouer à la variable, alors la valeur du champ error-status de la PDU Response est réglée à "inconsistentValue", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(11) Lorsque, durant l'étape ci-dessus, l'allocation de la valeur spécifiée par le champ Valeur de la liaison de variable à la variable spécifiée exige l'allocation d'une ressource présentement indisponible, alors la valeur du champ error-status de la PDU Response est réglée à "resourceUnavailable", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(12) Si le traitement de la liaison de variable échoue pour une raison autre que celles énumérées ci-dessus, alors la valeur du champ error-status de la PDU Response est réglée à "genErr", et la valeur de son champ error-index est réglée à l'indice de la liaison de variable en échec.

(13) Autrement, la validation de la liaison de variable est réussie.

À la fin de la première phase, si la validation de toutes les liaisons de variables a réussi, alors la valeur du champ error-status de la PDU Response est réglée à "noError" et la valeur de son champ error-index est de zéro, et le traitement continue comme suit.

Pour chaque liaison de variable dans la demande, la variable désignée est créée si nécessaire, et la valeur spécifiée lui est allouée. Chacune de ces allocations de variable survient comme si elle était simultanée par rapport à toutes les autres allocations spécifiées dans la même demande. Cependant, si la même variable est désignée plus d'une fois dans une seule demande, avec des valeurs associées différentes, l'allocation réelle faite à cette variable est alors spécifique de la mise en œuvre.

Si une de ces allocations échoue (même après toutes les validations précédentes) alors toutes les autres allocations sont annulées, et la PDU Response est modifiée pour avoir la valeur de son champ error-status réglée à "commitFailed", et la valeur de son champ error-index réglée à l'indice de la liaison de variable en échec.

Si et seulement si il n'est pas possible d'annuler toutes les allocations, la PDU Response est alors modifiée pour avoir la valeur de son champ error-status réglée à "undoFailed", et la valeur de son champ error-index est réglée à zéro. Noter que les mises en œuvre sont fortement encouragées à prendre toutes les mesures possibles pour éviter d'utiliser "commitFailed" ou "undoFailed" – ces deux codes d'état d'erreur ne sont pas à interpréter comme une licence de faire n'importe quoi dans une mise en œuvre.

Finalement, la PDU Response générée est encapsulée dans un message, et transmise au générateur de la PDU SetRequest.

4.2.6 PDU SNMPv2-Trap

Une PDU SNMPv2-Trap est générée et transmise par une entité SNMP au nom d'une application de générateur de notification. La PDU SNMPv2-Trap est souvent utilisée pour notifier à une application de receveur de notification à l'entité SNMP logiquement distante la survenance d'un événement ou la présence d'une condition. Il n'y a pas de confirmation associée à ce mécanisme de livraison de notification.

La ou les destinations auxquelles une PDU SNMPv2-Trap est envoyée sont déterminées d'une façon qui dépend de la mise en œuvre par l'entité SNMP. Les deux premières liaisons de variables de la liste des liaisons de variable d'une PDU SNMPv2-Trap sont respectivement sysUpTime.0 [RFC3418] et snmpTrapOID.0 [RFC3418]. Si la clause OBJECTS est présente dans l'invocation de la macro TYPE DE NOTIFICATION correspondante, chaque variable correspondante, comme instanciée par cette notification, est alors copiée, dans l'ordre, dans le champ variable-bindings. Si des variables supplémentaires sont incluses (ce qui est une option de l'entité SNMP génératrice) chacune est alors copiée dans le champ variable-bindings.

4.2.7 PDU InformRequest

Une PDU InformRequest est générée et transmise par une entité SNMP au nom d'une application de générateur de notification. La PDU InformRequest est souvent utilisée pour notifier à une application de receveur de notification qu'un événement est survenu ou qu'une condition est présente. C'est un mécanisme de confirmation de livraison de notification, bien qu'il n'y ait, bien sûr, aucune garantie de livraison.

La ou les destinations auxquelles est envoyée une PDU InformRequest sont spécifiées par l'application de générateur de notification. Les deux premières liaisons de variables de la liste de liaisons de variable d'une PDU InformRequest sont respectivement sysUpTime.0 [RFC3418] et snmpTrapOID.0 [RFC3418]. Si la clause OBJECTS est présente dans l'invocation de la macro TYPE DE NOTIFICATION correspondante, chaque variable correspondante, comme instanciée par cette notification, est alors copiée, dans l'ordre, dans le champ variable-bindings. Si des variables supplémentaires sont incluses (c'est une option de l'entité SNMP génératrice) chacune est alors copiée dans le champ variable-bindings.

À réception d'une PDU InformRequest, l'entité SNMP receveuse détermine la taille d'un message encapsulant une PDU Response avec les mêmes valeurs dans ses champs request-id, error-status, error-index et variable-bindings que dans la PDU InformRequest reçue. Si la taille déterminée du message est supérieure à une contrainte locale ou à la taille maximum de message du générateur, une PDU Response de remplacement est alors générée, transmise au générateur de la PDU InformRequest, et le traitement de la PDU InformRequest se termine immédiatement après. Cette PDU Response de remplacement est formatée avec les mêmes valeurs dans son champ request-id que la PDU InformRequest reçue, avec la valeur de son champ error-status réglée à "tooBig", la valeur de son champ error-index réglée à zéro, et un champ variable-bindings vide. Cette PDU Response de remplacement est alors encapsulée dans un message. Si la taille du message résultant est inférieure ou égale à la fois à la contrainte locale et à la taille maximum de message du générateur, il est transmis au générateur de la PDU InformRequest. Autrement, le compteur snmpSilentDrops [RFC3418] est incrémenté et le message résultant est éliminé. De toutes façons, le traitement de la PDU InformRequest est terminé.

Autrement, l'entité SNMP receveuse :
(1) présente son contenu à l'application appropriée ;
(2) génère une PDU Response avec les mêmes valeurs dans ses champs request-id et variable-bindings que celles de la PDU InformRequest reçue, avec la valeur de son champ État d'erreur réglé à "noError" et la valeur de son champ Indice d'erreur réglé à zéro ;
(3) transmet la PDU Response générée à l'origine de la PDU InformRequest.

5. Notice sur la propriété intellectuelle

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.

6. Remerciements

Le présent document est le fruit des travaux du groupe SNMPv3. Des remerciements particuliers sont adressés dans l'ordre alphabétique aux membres suivants du groupe de travail : Randy Bush, Jeffrey D. Case, Mike Daniele, Rob Frye, Lauren Heintz, Keith McCloghrie, Russ Mundy, David T. Perkins, Randy Presuhn, Aleksey Romanov, Juergen Schoenwaelder, Bert Wijnen.

La présente version du document, éditée par Randy Presuhn, se fonde à l'origine sur les travaux d'une équipe de conception dont les membres étaient : Jeffrey D. Case, Keith McCloghrie, David T. Perkins, Randy Presuhn, Juergen Schoenwaelder.

Les versions précédentes de ce document, éditées par Keith McCloghrie, étaient le résultat de travaux significatifs de quatre contributeurs majeurs : Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose, Steven Waldbusser

De plus, il faut aussi remercier des contributions du groupe de travail SNMPv2 aux versions précédentes. En particulier, des remerciements sont adressés pour les contributions de : Alexander I. Alten, Dave Arneson, Uri Blumenthal, Doug Book, Kim Curran, Jim Galvin, Maria Greene, Iain Hanson, Dave Harrington, Jeff Johnson, Michael Kornegay, Deirdre Kostick, David Levi, Daniel Mahoney, Bob Natale, Brian O'Keefe, Andrew Pearson, Dave Perkins, Randy Presuhn, Aleksey Romanov, Shawn Routhier, Jon Saperia, Juergen Schoenwaelder, Bob Stewart, Kaj Tesink, Glenn Waters, Bert Wijnen.

7. Considérations pour la sécurité

Le protocole défini dans le présent document ne procure pas par lui-même un environnement sûr. Même si le réseau est lui-même sûr (par exemple en utilisant IPSec) il n'y a pas de contrôle sur les personnes qui sont autorisées sur le réseau sûr à accéder aux informations de gestion.

Il est recommandé que les développeurs prennent en considération les caractéristiques de sécurité fournies par le cadre SNMPv3. Précisément, l'utilisation du modèle de sécurité fondée sur l'utilisateur, STD 62, [RFC3414] et le modèle de contrôle d'accès fondé sur la vue , STD 62, [RFC3415] est recommandée.

Il est donc de la responsabilité du consommateur/utilisateur de s'assurer que l'entité SNMP est configurée de façon que :
- seuls les principaux (utilisateurs) qui ont des droits légitimes puissent accéder aux valeurs ou à les modifier pour tout objet de MIB pris en charge par cette entité ;
- l'occurrence d'événements particuliers sur l'entité soit communiquée de façon appropriée ;
- l'entité réponde de façon appropriée et de bonne foi aux événements et informations qui lui sont communiqués.

8. Références

8.1 Références normatives

[RFC0768] J. Postel, "Protocole de datagramme d'utilisateur (UDP)", (STD006), 28 août 1980.

[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)

[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)

[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)

[ASN1] Organiaation Internationale de Normalisation, "Systèmes de traitement de l'information – Interconnxion des systèmes ouverts - Spécification de la notation de syntaxe abstraite numéro un (ASN.1)". Norme internationale 8824, décembre 1987.

8.2 Références pour information

[FRAG] Kent, C. and J. Mogul, "Fragmentation Considered Harmful," Proceedings, ACM SIGCOMM '87, Stowe, VT, août 1987.

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

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

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

[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.

[RFC1215] M. Rose, "Convention pour la définition de filtres à utiliser avec le SNMP", mars 1991. *(Info)*

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

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

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

[RFC2914] S. Floyd, "Principes du contrôle d'encombrement", BCP 41, septembre 2000.

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

9. Changements depuis la RFC 1905

Voici les changements par rapport à la RFC 1905 :
- Correction d'une erreur d'orthographe dans la déclaration de copyright ;
- Mise à jour de la date de copyright ;
- Mise à jour des informations sur l'éditeur et le point de contact ;
- Ajout d'une notice sur la propriété intellectuelle ;
- Corrections mineures de la présentation et de la typographie ;

- Ajout d'une table des matières ;
- Modification du titre ;
- Mise à jour des en-têtes et des pieds de page du document ;
- Suppression de l'ancien paragraphe 2.3, intitulé "Accès aus informations de gestion" ;
- Changement de la façon de définir l'identifiant de demande, bien qu'avec la même syntaxe et sémantique finales, pour éviter le couplage avec le SMI. Cela n'affecte en aucune façon le protocole ;
- Remplacement du mot "exception" par le mot "erreur" dans l'ancien paragraphe 4.1. Cela n'affecte en aucune façon le protocole ;
- Suppression des deux premiers alinéas du paragraphe 4.2;
- Précision du nombre maximum de liaisons de variables qu'une mise en œuvre doit prendre en charge dans une PDU. Cela n'affecte en aucune façon le protocole ;
- Remplacement des occurrences de "application SNMPv2" par "application" ;
- Suppression de trois phrases dans le paragraphe 4.2.3 décrivant le traitement d'une situation impossible. Cela n'affecte en aucune façon le protocole ;
- Précision de l'utilisation de SNMPv2-Trap-Pdu dans le paragraphe 4.2.6. Cela n'affecte en aucune façon le protocole ;
- Alignement de la description de l'utilisation de InformRequest-Pdu dans le paragraphe 4.2.7 avec l'architecture. Cela n'affecte en aucune façon le protocole ;
- Mise à jour des références ;
- Réécriture du paragraphe d'introduction ;
- Remplacement de la terminologie d'entité gestionnaire/agent/SNMPv2 par la terminologie de la RFC 2571. Cela n'affecte en aucune façon le protocole ;
- Elimination des IMPORTS provenant du SMI, remplacés par l'équivalent en ligne ASN.1. Cela n'affecte en aucune façon le protocole ;
- Ajout de notes attirant l'attention sur deux manifestations différentes pour atteindre la fin d'un tableau dans les exemples de tableaux ;
- Ajout de contenu à la section sur les considérations pour la sécurité ;
- Mise à jour des commentaires ASN.1 sur l'utilisation de Report-PDU. Cela n'affecte en aucune façon le protocole ;
- Mise à jour de la sectioon de remerciements ;
- Inclusion d'informations sur le traitement de BITS ;
- Suppression de virgules parasites dans la définition ASN.1 des PDU ;
- Ajout du résumé ;
- Le traitement des liaisons de variables supplémentaires dans informs a été rendu cohérent avec celui de traps. Ceci est la correction d'un oubli rédactionnel, et reflète la practique des mises en œuvre ;
- Ajout de la référence à la RFC 2914.

10. Adresse de l'éditeur

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

11. Déclaration complète de droits de reproduction

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