1.7 SNMPv2

Cette nouvelle approche, basée sur les expériences pratiques avec SNMPv1, augmente considérablement les fonctionnalités de la gestion réseau. Elle propose :

- **Modèle d'accès aux objets protégé** : La sécurité individuelle et les mécanismes d'authentification peuvent être configurés par l'administrateur réseau par "**parties**". Les divers droits d'accès et les perspectives sur la MIB peuvent être paramétrés individuellement par les "parties" et plusieurs stations de gestion peuvent accéder à un agent. Certains objets de la MIB pourront être accessibles que par certaines station de gestion. La MIB standard a du être révisée pour permettre ces nouvelles fonctionnalités.

- **Extension de la SMI** : pour l'authentification et l'autorisation.
- **Communication étendue** : La distinction entre agent et station de gestion est supprimée. Une machine peut être les 2 à la fois. Des managers peuvent communiquer.
- **Simplification de requête** : Une nouvelle opération "GET-BULK-REQUEST" est proposée permettant de lire un arbre complet de la MIB en une seule requête.
- **Signalisation étendue des erreurs** : SNMPv1 proposait 13 codes d'erreurs connus. Maintenant l'information contenue dans les Traps peut intégrer des erreurs avec une référence spécifique.
- **Extension à d'autres protocole que UDP** : Novell-IPX, DPP
- **Compatibilité avec SNMPv1**

1.7.1 PDU SNMP-TRAPv2

Un TRAPv2 a le même format de trame que les PDU get, set ou get-next. Il a le même rôle qu'un TRAPv1. L'information de celui-ci est contenu dans les variables liées. Elles sont organisées par exemple de la manière suivante:

– sysUpTime :"32216671"
– snmpTrapIOD.0 : "snmpTraps.3"qui identifie le type de trap
– ifIndex.4 : "4"Variables additionnelles du trap
– ifAdminStatus.4 : "up"
– ifOperStatus.4 : "down"

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).interfaces(2).ifTable(2).ifEntry(1)
       .ifIndex(1)        R Integer32  Range: 1..2147483647
       .ifAdminStatus(7)  RW EnumVal Values: up(1),down(2),testing(3)
       .ifOperStatus(8)   R EnumVal Values:up(1),down(2),testing(3),unknown(4),dormant(5),notPresent(6),lowerLayerDown(7)
```

Il n'y a pas d'accusé de réception d'un Trap de la par de la station de gestion. C'est l'agent qui doit éventuellement retransmettre le Trap au cas ou il n'y a pas d'intervention de la station de gestion.

Syntaxe de la commande SNMPTRAPv2 de NET-SNMP à respecter :

```
snmptrap -v 2c [-Ci] [arg commun] uptime  trap-oid  [objectID type valeur]...
```

1.7.2 PDU SNMP-INFORMREQUESTv2

Lors d'un PDU-TRAP, il n'y a pas de message réponse et on n'est pas assuré de la réception (UDP). Les messages INFORM vise se problème en devenant un TRAP avec une réponse. Un agent pourra émettre ce PDU tant qu'il n'a pas reçu de réponse.

On pourra envisager de l'envoyer d'une station de gestion vers une autre station de gestion au sujet d'une application pour stipuler une information d'administration.

Cette trame est envoyée à la destination spécifiée par la table SNMPv2 Event Notify Table. Une PDU-RESPONSE est générée à sa réception.

Syntaxe de la commande SNMPINFORMv2 de NET-SNMP à respecter :

```
snmpinform -v [2c|3] [arg commun] uptime  trap-oid  [objectID type valeur]...
```

1.7.3 PDU SNMP-GET-BULKv2

Son **objectif** est de **minimiser** le **nombre d'échanges** protocolaires requis pour extraire une grande quantité d'informations en gérant la transmission d'une portion d'une table. En fait, elle supprime la répétition des PDU-GETNEXT REQUEST.

Elle utilise le même principe que le PDU-GETNEXT REQUEST à savoir la sélection de l'objet suivant et suit l'ordre de celle-ci dans l'arbre.

On spécifie en plus 2 valeurs: "n" pour le nombre de variables à obtenir et "r" le nb max de répétitions

Syntaxe de la commande SNMPBULKGET à respecter :

```
snmpbulkget [arguments communs] [objectID]...
```

1.7.4 Nouvelles indications d'erreurs en version 2

ES : peut prendre de nouvelles valeurs :

- noAccess(6) : variable non accessibles
- wrongType(7) : tentative d'affectation d'une variable avec un type incorrect.
- wrongLength(8) : tentative d'affectation d'une variable avec une longueur incorrect.
- wrongEncoding(9) : tentative d'affectation d'une variable avec un encodage ASN.1 incorrect.
- wrongValue(10) : tentative d'affectation d'une variable avec une valeur incorrect.
- noCreation(11) : tentative de modification ou création d'une variable qui n'existe pas ou ne pouvant être créée
- inconsistentValue(12) : tentative d'affectation d'une variable avec une valeur qui dans d'autres circonstances serait correcte.
- ressourceUnavailable(12) : tentative d'affectation d'une variable avec une valeur qui requière une allocaion de ressource indisponible actuellement.
- commitFailed(14) : si l'opération d'affectation échoue.
- undoFailed(15) : si l'opération d'affectation échoue et que toutes les assignations ne peuvent être annulées.
- authorizationError(16) : si une requête "get", "get-next", "get-bulk", "set" ou "inform" n'est autirisée.
- notWritable(17): tentative d'affectation d'une variable existante mais non modifiable
- inconsistentName(18) : tentative d'affectation d'une variable qui n'existe pas et ne pouvant être créés.

1.8 SNMPv3

1.8.1 PDU SNMP-REPORTv3

Ce type de PDU fut défini avec la version 2 mais jamais utilisé. Pour la version 3, il permet à un moteur SNMP d'appeler un autre moteur SNMP qu'une erreur a été détectée durant un message SNMP. Ceci (en théorie) permet à un moteur SNMP d'envoyer un message SNMP correct.

Les PDU REPORTv3 sont générés pour rapporter les types de problèmes suivants :

- Un message réponse ne peut être généré.
- Un message reçu avec "authFlag" à 0 et "privFlag" positionné.
- Une erreur se présente durant une authentification et des services privés pour un message de retour.

ReqId est soit le même que le message ayant déclenché le "report" ou 0 si celui du message n'a pu être extrait.

La variable liée contiendra un unique objet et sa valeur utilisé pour déterminer le problème.