Administrative Model for Version 2 of the Simple Network Management Protocol (SNMPv2)

Fri Sep 08 1995

draft-various-snmpv2-adminv2-syn-01.txt

Tell U. Later
snmpv2@tis.com

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet- Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

running list of open issues:

tie procedures more closely to specific mib objects

reference list
reference citations
acknowledgements
authors
author addresses
spell check

Abstract

Computer security systems can be usually be understood as being composed of two or more logically distinct components. There is often a privacy service that protects data from disclosure, an authentication service that validates the identity of the entity requesting service, and an access control service which, given an authorized entity, restricts the data and operations to which that entity has access.

The SNMPv2 Administrative Model makes these distinctions. It specifies the data and protocols that a compliant SNMPv2 entity must implement in order to provide an access control service. However, it does not say anything about how authentication, privacy, timeliness, and the like ought to be implemented. Rather, it leaves the hooks in place for these services to be implemented in a variety of ways. Henceforth, this collection of unspecified services will be collectively referred to as "the authentication and privacy services". Consequently, the SNMPv2 Administrative Model provides an architecture that realizes an access control service, and provides the means for integrating authentication and privacy services.

The access control service of the SNMPv2 Administrative Model contains the following key concepts: groups, contexts, MIB views, and access rights. The idea is that on a particular SNMPv2 entity, the identities within a group have access (within a particular context) to a set of MIB variables specified by the MIB view and can operate on those variables according to the what the access rights allow.

There are two principal provisions for integrating authentication and privacy services. First, there is a field in the SNMPv2 message header that indicates which authentication and privacy service the sender used and therefore a receiver of the message should use; and there is an opaque set of fields which contain security information specific to the indicated authentication and privacy service. Second, and more significantly, there is the concept of an "identity". The security service may use this identity to authenticate the packet, but it MUST also map the identity to a group so that the access control service can perform access control checking. In other words, the concept of an identity is used by the authentication and privacy service, while the concept of a group is used by the access control service. It is the responsibility of the authentication and privacy service to maintain the mapping between an identity and a group.

1. Introduction

A management system contains: several (potentially many) manageable nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol. The management protocol is used to convey management information between the agents and management stations; and, for manager-to-manager communications, between management stations. Operations of the protocol are carried out under an administrative framework which defines authentication, authorization, access control, and privacy policies.

Management stations execute management applications which monitor and control managed elements. Managed elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled via access to their management information.

It is the purpose of this document, the Administrative Model for SNMPv2, to define an administrative framework which realizes effective management in a variety of configurations and environments.

The model described here includes the identification of the entities on whose behalf SNMPv2 messages are exchanged. Thus, it represents a departure from the community-based administrative model of the original SNMP [@ref 1157]. This new strategy improves upon the historical community-based scheme by defining a model for authentication, authorization, access control, and privacy including definitions of an initial set of mechanisms to provide these services. A wide range of mechanisms are compatible with this model, including mechanisms based on symmetric (private key) security protocols and mechanisms based on asymmetric (public key) security protocols.

1.1. A Note on Terminology

The original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, [@ref 1155, 1157, 1212], defined an initial community-based administrative model, and an initial set of protocol operations. For the purpose of exposition, this is termed the SNMP version 1 framework (SNMPv1).

The SNMPv2 management framework consists of multiple building blocks, including SMI definitions [@ref v2smi], [@ref tc], and [@ref conf]; MIB definitions [@ref]; an administrative framework consisting of an administrative model described herein, definitions of security and privacy services such as those found in [@ref sec], and a set of MIB

definitions to support the administrative model [@ref adminmib]; definitions of protocol operations [@ref protoops]; and SNMPv2 applications.

The SNMP version 2 administrative framework (SNMPv2) includes an administrative model as described herein which couples SNMPv2 applications to one or more transport stacks and thereby provides remote, networked, access to management information via an enhanced set of protocol operations as described in [@ref protoops].

A transitional approach, using the SNMP version 1 administrative model in combination with the SNMP version 2 protocol operations is defined in [@ref v2Cadmin].

2. Overview

This section provides an overview of the remainder of the document, including an introduction to some of the pertinent vocabulary. As such, there is some overlap between this section and the subsequent ones in order to avoid forward references.

The administrative model which is a part of the SNMP version 2 administrative framework couples SNMPv2 applications to one or more transport stacks and thereby provides remote, networked, access to management information (See Figure 1).

An SNMPv2 entity is an implementation of the administrative framework described herein, coupled with one or more applications, such as agent, manager, or dual-role entity applications, which remotely access management information via the SNMPv2.

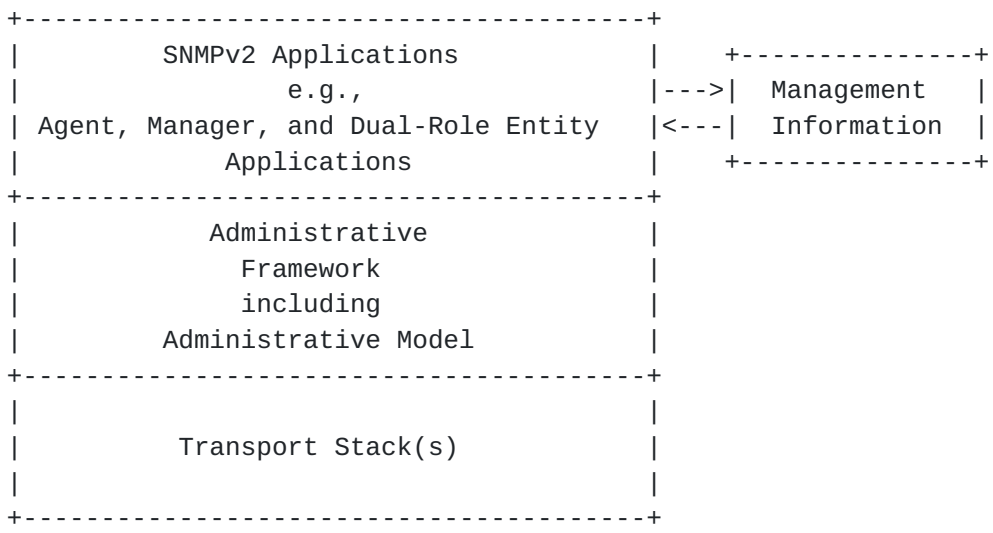

Figure 1: The SNMPv2 Administrative Framework

2.1. Management Information

A management domain typically contains a large amount of management information. Each individual item of management information is an instance of a managed object type. The definition of a related set of managed object types is contained in a Management Information Base (MIB) module. Many such MIB modules are defined. For each managed object type it describes, a MIB module defines not only the semantics and syntax of that managed object type, but also the method of identifying an individual instance so that multiple instances of the same managed object type can be distinguished.

2.2. Contexts

Typically, there are many instances of each managed object type within a management domain. For simplicity, the method for identifying instances specified by the MIB module does not allow each instance to be distinguished amongst the set of all instances within the management domain; rather, it allows each instance to be identified only within some scope or "context", where there are multiple such contexts within the management domain. Often, a context is a physical device, or perhaps, a logical device, although a context can also encompass multiple devices, or a subset of a single device, or even a subset of multiple devices. Thus, in order to identify an individual item of management information within the management domain, its context must be identified in addition to its object type and its instance. Note that

this requires each context to have a globally-unique identification within the management domain. Note also that the same item of management information can exist in multiple contexts. Contexts are identified in an unambiguous and a globally unique manner by the pairing of a contextSnmpID and a contextName; with the contextSnmpID identifying a SNMPv2 entity which is an implementation of the administrative framework including its administrative model, and the contextName identifying a context within that entity.

For example, the managed object type, ifDescr [@ref ifevolution], is defined as the description of a network interface. To identify the description of device-X's first network interface, four pieces of information are needed: the contextSnmpID which uniquely identifies device-X, the contextName within device-X, ifDescr (the managed object type), and "1" (the instance).

The contextName is expressed as a user-friendly, and often mnemonic, string-based label which is convenient for manipulation by humans within a system, such as when configuring a device.

Management information often changes over time, for example, where the value of a device parameter after the device's next restart is to be different than its current value. Thus, when naming a specific value of a managed object instance, an indication of time is needed. In most situations, it is the value at the current time which is of interest to the network manager. There are, however, situations where times other than the current time are of interest, e.g., such as after the next restart. To accommodate this, each context has an associated notion of time, called its temporal domain. This allows, for example, one context to refer to the current values of a device's parameters, and a different context to refer to the values that the same parameters for the same device will have after the device's next restart.

2.3. MIB Views

For security reasons, it is often valuable to be able to restrict the access rights of some management applications to only a subset of the management information in the management domain. To provide this capability, access to a context is via a "MIB view" which details a specific set of managed object types (and optionally, the specific instances of object types) within that context. For example, for a given context, there will typically always be one MIB view which provides access to all management information in that context, and often there will be other MIB views each of which contains some subset of the information. So, by providing access rights to a management application

in terms of the particular (subset) MIB view it can access for that context, then the management application is restricted in the desired manner.

Since managed object types (and their instances) are identified via the tree-like naming structure of ISO's OBJECT IDENTIFIERs [@ref iso8824, v2smi], it is convenient to define a MIB view as the combination of a set of "view subtrees", where each view subtree is a sub-tree within the managed object naming tree. Thus, a simple MIB view (e.g., all managed objects within the Internet Network Management Framework) can be defined as a single view sub-tree, while more complicated MIB views (e.g., all information relevant to a particular network interface) can be represented by the union of multiple view sub-trees.

While any set of managed objects can be described by the union of some number of view subtrees, situations can arise that would require a very large number of view subtrees. This could happen, for example, when specifying all columns in one conceptual row of a MIB table because they could appear in separate subtrees, one per column, each with a very similar format. Because the formats are similar, the required set of subtrees can easily be aggregated into one structure. This structure is named a family of view subtrees after the set of subtrees that it conceptually represents. A family of view subtrees can either be included or excluded from a MIB view.

2.4. Access Rights

In addition to restricting access rights by identifying (sub-)sets of management information, it is also valuable to restrict the operations allowed on the management information within a particular context. For example, one management application might not be permitted write-access to a particular context, while another might be allowed to perform any type of operation.

2.5. Authentication and Privacy

The enforcement of access rights requires the means not only to identify the entity on whose behalf a request is generated but also to authenticate such identification. Another security capability which is (optionally) provided is the ability to protect the data within an SNMPv2 operation from disclosure (i.e., to encrypt the data). This is particularly useful when sensitive data (e.g., passwords) are accessed via SNMPv2 requests.

2.6. Security Mechanisms and Algorithms

Recommendations for which algorithms are best for authentication and privacy are subject to change. Such changes may occur when new research results on the vulnerability of various algorithms are published, and/or with the prevailing status of export control and patent issues. Thus, it is valuable to allow these algorithms to be specified as parameters, so that new algorithms can be accommodated over time. In particular, one type of algorithm which may become increasingly useful in the future is the set of algorithms associated with asymmetric (public key) cryptography.

Note that not all accesses via SNMPv2 requests need to be secure. Indeed, there are purposes for which insecure access is required. One example of this is the ability of a management application to learn about devices of which it has no previous knowledge. Another example is to perform any synchronization which the security algorithms need before they can be used to communicate securely. This need for insecure access is accommodated by defining one of the algorithms for security as providing neither authentication nor protection against disclosure.

2.7. Security Protocol Identifiers and the Reportable Flag

Each SNMPv2 message header contains a field which indicates the particular authentication and privacy mechanisms and protocols in use within that message. This is called the security flags field.

The security flags field conveys two aspects: the security protocol identifier (sPI) value and the reportableFlag (reportableFlag).

The sPI value can be used to select a particular authentication and privacy service, which, among other things, can be used to derive an identity from a message. Correspondingly, the sPI value is added to a message as a part of message generation for later use in the decoding process.

In general, it is poor protocol design to generate an error message in response to an error message and the SNMPv2 administrative framework includes mechansims to prevent such behaviors. The reportableFlag portion of the security flags is used to indicate if the message contains an SNMPv2 protocol operation which could possibly result in a report operation signifying an error. That is, the reportableFlag is true if the message contains a GetRequest, GetNextRequest, GetBulkRequest, SetRequest, or InformRequest operation. This reportableFlag is never true for a message which contains a Response,

SNMPv2-Trap, or Report operation.

The security flags field is a signed integer, consisting of two parts corresponding to the reportableFlag and the sPI. The reportableFlag is communicated in the least significant bit of the security flags field.

The remaining bits convey the value of the sPI.

The values of sPI are allocated as follows. Negative and zero values for sPI are reserved. Values of sPI between 1 and 127, inclusive, are reserved for use with standards-track protocols and are managed by the Internet Assigned Numbers Authority (IANA).

Values of sPI greater than 127 are allocated to enterprise-specific protocol definitions. An enterprise-specific sPI value is defined to be the enterprise number * 128 + the protocol number within that enterprise. For example, the fourth protocol defined by the enterprise whose enterprise number is 1 would be 132.

These seven bits allow a maximum of 128 standards-based authentication and privacy services. Similarly, they allow a maximum of 128 authentication and privacy services per enterprise. It is believed that the assignment of new sPI values should be rare in practice because the larger the number of simultaneously utilized security protocols, the larger the chance that interoperability will suffer. Consequently, it is believed that such a range will be sufficient. In the unlikely event that the standards committee finds this number to be insufficient over time, an enterprise number can be allocated to obtain an additional 127 possible values.

Note that the most significant bit must be zero; hence, there are 23 bits allocated for various organizations to design and define non-standard security protocols. This limits the ability to define new proprietary implementations of security protocols to approximately the first 8 million enterprises.

It is worthwhile to note that, in its encoded form, the sPI value will normally require only a single byte since, in practice, the leftmost bits will be zero for most messages and sign extension is suppressed by the encoding rules.

As of this writing, there are several values of sPI defined for use with SNMPv2 or reserved for use with supporting MIB objects. They are as follows:

1 reserved for use in MIB objects and in association with the SNMPv1

administrative framework [@ref 1157]

2 reserved for use in MIB objects and in association with the SNMPv2C administrative framework [@ref v2Cadmin]

3 unauthenticated report messages

4 usecNoAuth messages [@ref sec]

5 usecAuth messages [@ref sec]

6 usecPriv messages [@ref sec]

2.8. Identities and Group Names

Each SNMPv2 message conveys information on behalf of an identity. To provide the necessary security capabilities, an SNMPv2 message needs to indicate that identity. The identity can be conveyed either explicitly or implicitly.

The exact manner in which an identity is conveyed in an SNMPv2 message is dependent upon the particular security mechanisms and protocols which have been applied to that SNMPv2 message and these may vary from time-to-time and from message-to-message.

In addition to the sPI value, each message header also contains a set of fields, called the authInfo sequence, which contains the necessary parameters used by the authentication and privacy service identified by the sPI.

Accordingly, the sPI field indicates how a receiver should parse and interpret the data found in the authInfo portion of the header upon receipt as well as how to generate the authInfo portion of the header prior to transmission.

While there are a nearly infinite number of possibilities as to how the information regarding an identity can be encoded in the packet header, identities within a system are always identified by user-friendly, and often mnemonic, string-based labels which are convenient for manipulation by humans within a system, such as when configuring a device.

It is the responsibility of the authentication and privacy service to transform the encoded information in a packet header into a string-based

identity upon receipt of a message and to transform a string-based identity into the properly encoded information for the sPI in effect.

An identity may operate at more than one manager location and more than one agent location or may be restricted to a single pair of locations, depending upon the particular authentication and privacy sPI in effect.

As a result, an identityName alone is insufficient to uniquely designate an identity. Consequently, in some parts of an SNMPv2 infrastructure, an identityName is paired with an authSnmpID, in order to provide a globally unique designation of an identity and its associated parameters. This allows, for example, the use of multiple passwords by a single identity. However, in some parts of an SNMPv2 infrastructure, in particular some low cost systems, the authSnmpID is always a constant value corresponding to an identifier which uniquely identifies that entity.

For some protocol operations, local agent operations in particular, it is both convenient and efficient to grant access based on various group profiles rather than per individual identity. For implementations of this model which perform those operations, each identity is also associated with a groupName.

A groupName, like an identityName, is always identified by a user-friendly, and often mnemonic, string-based label which is convenient for manipulation by humans within a system, such as when configuring a device. For example, an identity might be associated with a group of identities with a groupName of "ActiveDayShiftSupervisor" whereas other identities might be associated with a group of identities with a groupName of "FieldServiceTechnician".

2.9. Authorization: Access Control

As described above, an SNMPv2 message is associated with one context, parameters in an authInfo sequence, and security flags field, where the context determines the set of management information being accessed by the message, the authInfo parameters convey the identity on whose behalf the information is being communicated, and the security flags field conveys the reportableFlag and sPI values. The reportableFlag indicates if failures in the processing of the message are allowed to invoke report operations and the sPI value indicates the type and nature of the authentication and privacy parameters. These properties of the message are used for access control within local operations and to control the forwarding of messages within proxy operations. Specifically, access control is specified as a set of local access policies which regulate

access to data within a specified local context by a given identity using a particular sPI. The local access control specifies the set of permitted operations and the MIB views which are allowed. If the context, the identity derived from the authInfo sequence, and sPI are not a valid combination, or the operation requested is not one of the operations allowed by the triple, then the requested access is denied.

2.10. Construction of an SNMPv2 Message

The purpose of an SNMPv2 message is to contain an SNMPv2 PDU. (The PDU contains an operation-code, some additional request/response parameters and a list of names and values of specific managed object instances; for details, see [@ref protoops].)

There are seven parts to every SNMPv2 message. They are as follows:

- The first part of every SNMPv2 message conveys the version number. Every SNMPv2 message has a version number of "2" which denotes that it supports version 2 of the administrative model as described herein and version 2 of the protocol operations, as described in [@ref protoops]. (Version 1 of the Internet-standard management framework allocated "0" to this field. The value "1" is reserved for use to denote the SNMPv2C administrative model [@ref v2Cadmin].)

  By doing so, a parser of received messages can determine if a packet should be processed by the SNMPv2 engine, or should be redirected to an SNMPv1 protocol engine [@ref 1157], or an SNMPv2C protocol engine [@ref v2Cadmin].

- The mms portion of the message conveys information about the sender's maximum message size for messages received via the same transport layering used by the message.

- The security flags portion conveys the reportableFlag and the security protocol identifier (sPI). The sPI indicates the authentication and privacy service used by the message.

- The authInfo portion of the header contains the parameters used by that authentication and privacy service. Hence, the exact contents of the authInfo portion of the header varies, depending upon the value of sPI.

- The contextSnmpID is the fifth portion of the header. The contextSnmpID conveys the globally unique portion of the context

identification.

- The contextName is the sixth portion of the header. The contextName conveys the locally-significant portion of the context identification within the system denoted by the contextSnmpID.

- The PDU is the seventh, and last, portion of the message.

The concatenation of a contextSnmpID, contextName, and PDU is termed a ScopedPDU for the convenience of exposition. The format of a message is illustrated in Figure 2 (formal ASN.1 definitions follow in a later section).

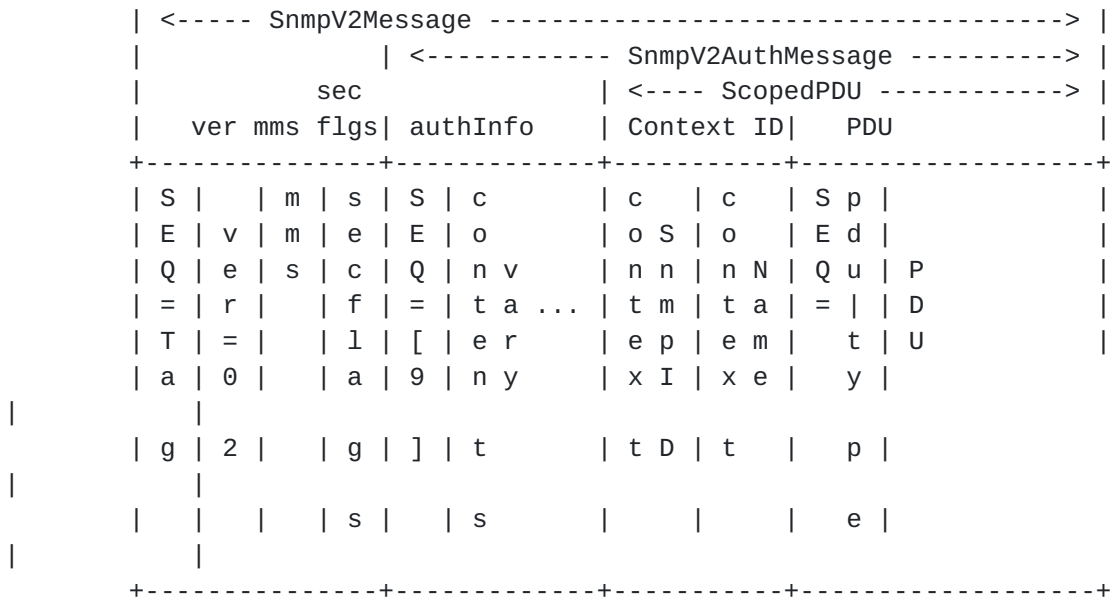

Figure 2: SNMPv2 Message Format

2.11. Authentication and Privacy Service

There can be a number of authentication and privacy services, denoted by various values of sPI, defined for use with the administrative framework defined by this document. Any particular implementation may support one or more of the possible authentication and privacy services. The definition of an initial set of these is found in accompanying documents [@ref sec] and additional values of sPI may be defined in future documents.

Any newly-defined authentication and privacy service must:

- process the authInfo portion of the message to derive an identityName, authSnmpID, and groupName;

- decrypt, if necessary, the ScopedPDU containing the context information and PDU to provide plaintext, and

- provide any other information needed, if any, to be cached for use when producing a subsequent message, e.g., a response to a query or command, or in association with proxy forwarding operations.

Any newly-defined authentication and privacy service must also be able to generate an SnmpV2AuthMessage value from:

- an identityName and authSnmpID;

- an sPI value; and

- a ScopedPDU with context information consisting of a contextName and contextSnmpID followed by a PDU.

2.12. An SNMPv2 Entity's Local Configuration Datastore

To implement the model described in the preceding sections, each SNMPv2 entity needs to retain its own set of information about contexts, identities, access rights and policies, and the like. This set of information is called the SNMPv2 entity's Local Configuration Datastore (LCD) because it is locally-stored information. Note, however, that the LCD often contains information about both local and remote systems.

In order to allow an SNMPv2 entity's LCD to be configured, and to allow the synchronization needed by the security algorithms before they can be used to communicate securely, portions of the LCD need to be accessible as managed objects. A MIB module, the SNMPv2 Administration MIB, which defines these managed object types is contained in [@ref adminmib]. Furthermore, each definition of an authentication and privacy service, denoted by an sPI, must also define any MIB and LCD support needed by that authentication and privacy service. It can do this either by describing new MIB elements, defining additional MIB objects which AUGMENT those of another authentication and privacy service, or by explaining the proper semantics and rules for use of MIB objects previously defined for use by another authentication and privacy sPI.

It is expected that, over time, each of these three approaches will be used for defining the relevant information in the LCD. As of this writing, one set of authentication service-specific MIB objects are shared by three authentication and privacy services [@ref sec]. Other objects provide shared support for the SNMPv1 [@ref 1157] and SNMPv2 [@ref v2Cadmin].

2.13. Maintenance Functions

In order to facilitate communication between SNMPv2 entities, certain "maintenance" functions are defined. A maintenance function is identified as a management communication which accesses a well-known or algorithmically derived maintenance context and makes use of a corresponding well-known or algorithmically derived maintenance identity. For example, error reporting and clock synchronization, are achieved by performing SNMP operations in this manner.

When processing a maintenance function, an SNMPv2 entity utilizes the same mechanisms defined for normal operations; however, unlike normal operations which are executed with respect to an administration's security policy (which may vary between administrations), maintenance functions always occur within a fixed, standardized security policy. This is advantageous in that it allows code re-use within an SNMPv2 entity, while also not allowing an administration's policy to impair the proper operation of essential maintenance functions. However, many of the rules applicable to normal identities and contexts specified in this document do not always apply to these maintenance functions.

The sole purpose of maintenance functions is to ensure that all SNMPv2 entities provide essential maintenance functionality within a well-known, standardized, security environment. Maintenance functions are intended for use only by the internal operations of an SNMPv2 entity. Thus, their scope is intentionally restricted to be the minimum necessary to fulfill their purpose.

The only maintenance function defined in this specification of the administrative model is that of error reporting.

Additional maintenance functions may be defined by particular security and privacy services.

2.14. Proxy

The identification of a context is (architecturally) independent of the location at which its management information can be accessed. Of course, it is an SNMPv2 agent which responds to requests for access to management information. Each such request is contained within an SNMPv2 message which provides the capability to perform a single operation on a list of items of management information. Rather than having to identify the context as well as the managed object type and instance for each item of management information, each SNMPv2 message is concerned with only a single context. Thus, an SNMPv2 agent must be able to process

each request for items of management information within one of the possibly multiple contexts it supports.

In responding to a request, an SNMPv2 agent might be acting as a proxy for some other agent. The term "proxy" has historically been used very loosely, with multiple different meanings. These different meanings include (among others):

(1) the forwarding of SNMPv2 requests to other SNMP agents without regard for what managed object types are being accessed; for example, in order to forward an SNMPv2 request from one transport domain to another, or to translate SNMPv2 requests into SNMPv1 requests;

(2) the translation of SNMPv2 requests into operations of some non-SNMP management protocol; and

(3) support for aggregated managed objects where the value of one managed object instance depends upon the values of multiple other (remote) items of management information.

Each of these scenarios can be advantageous; for example, support for aggregation of management information can significantly reduce the bandwidth requirements of large-scale management activities. However, using a single term to cover multiple different scenarios causes confusion.

To avoid such confusion, the SNMPv2 administrative framework uses the term "proxy" with a much more tightly defined meaning which covers only the first of those listed above. Specifically, the distinction between a regular SNMPv2 agent and a "proxy SNMPv2 agent" is simple:

- a proxy SNMPv2 agent is an SNMPv2 agent which forwards requests on to other agents according to the context, and irrespective of the specific managed object types being accessed;

- in contrast, an SNMPv2 agent which processes SNMPv2 requests according to the (names of the) individual managed object types and instances being accessed, is NOT a proxy SNMPv2 agent from the perspective of this administrative framework.

Thus, when an SNMPv2 agent acts as a proxy SNMPv2 agent for a particular context, not only is the information on how to forward the request specifically associated with that context, but the proxy SNMPv2 agent has no need of a detailed definition of the MIB view (since the proxy

SNMPv2 agent forwards the request irrespective of the managed object types).

In contrast, a non-proxy SNMPv2 agent must have the detailed definition of the MIB view, and even if it needs to issue requests to other agents, that need is dependent on the individual managed object instances being accessed (i.e., not only on the context).

An SNMPv2 agent need not implement proxy forwarding operations in order to be compliant with this specification.

2.15. Transparency Principle

The transparency principle that defines the behavior of an SNMPv2 entity in general, applies in particular to a proxy SNMPv2 context:

> The manner in which a receiving SNMPv2 entity processes SNMPv2 protocol messages sent by another SNMPv2 entity is entirely transparent to the sending SNMPv2 entity.

Implicit in the transparency principle is the requirement that the semantics of SNMPv2 management operations are preserved between any two SNMPv2 peers. In particular, the "as if simultaneous" semantics of a Set operation are extremely difficult to guarantee if its scope extends to management information resident at multiple network locations. For this reason, proxy configurations which support Set operations to information at multiple locations are discouraged, although such operations are not explicitly precluded by the architecture in those rare cases where they might be supported in a conformant way.

Also implicit in the transparency principle is the requirement that, throughout its interaction with a proxy SNMPv2 agent, an SNMPv2 manager is supplied with no information about the nature or progress of the proxy mechanisms used to perform its requests. That is, it should seem to the SNMPv2 manager (except for any distinction in an underlying transport address) as if it were interacting via SNMPv2 directly with the proxied device. Thus, a timeout in the communication between a proxy SNMPv2 agent and its proxied device should be represented as a timeout in the communication between the SNMPv2 manager and the proxy SNMPv2 agent. Similarly, an error response from a proxied device should - as much as possible - be represented by the corresponding error response in the interaction between the proxy SNMPv2 agent and SNMPv2 manager.

(Note, however that amongst the error conditions indicated by a Report

PDU received by a proxy SNMPv2 agent from a proxied device, some may be corrected without being reported back to the SNMPv2 manager; for example, when a clock resynchronization is needed. Even when such errors can not be corrected, they are only indirectly reported back.

3. Elements of the Model

This section provides a more formal description of the model.

3.1. SNMPv2 Applications

An SNMPv2 application is an application-layer component which effects logically remote monitoring and control functions by generating or receiving (and processing) one or more types of transactions contained within SNMPv2 messages.

SNMPv2 entities include: SNMPv2 agents, SNMPv2 managers, and SNMPv2 dual-role entities.

3.1.1. SNMPv2 Agent

An SNMPv2 agent is the operational role assumed by an SNMPv2 entity when it acts in an agent role. Specifically, an SNMPv2 agent performs SNMPv2 protocol operations in response to received SNMPv2 protocol messages (except for inform notifications) generated by an SNMPv2 manager. Some SNMPv2 agents also emit trap notifications.

For the purposes of this administrative model, when a dual-role entity is acting in an agent role, it is considered to be an SNMPv2 Agent.

3.1.2. SNMPv2 Manager

An SNMPv2 manager is the operational role assumed by an SNMPv2 entity when it acts in a manager role on behalf of management applications. Specifically, an SNMPv2 entity acts in a manager role when it initiates SNMPv2 management operations by the generation of appropriate SNMPv2 protocol messages, including when it generates inform notifications, when it receives and processes trap and inform notifications, and when it receives reports. In those cases where an SNMPv2 manager receives inform requests, it also generates reports.

For the purposes of this administrative model, when a dual-role entity is acting in a manager role, it is considered to be an SNMPv2 manager.

3.1.3. SNMPv2 Dual-Role Entity

An SNMPv2 entity which sometimes acts in an agent role and sometimes acts in a manager role, is termed an SNMPv2 dual-role entity. An SNMPv2 dual-role entity receives requests for service through acting in an agent role and performs requests through acting in a manager role.

There are two categories of SNMPv2 dual-role entities:

(1) proxy SNMPv2 agents, and

(2) (so-called) mid-level managers.

Proxy SNMPv2 agents only forward requests/responses; they do not originate requests. In contrast, mid-level managers often originate requests. A primary purpose of SNMPv2 dual-role entities is often to generate requests and aggregate responses.

3.1.4. SNMPv2 Application Layering within the Administrative Framework

The following drawing of the administrative framework illustrates the conceptual layering of SNMPv2 applications on top of the administrative model (See Figure 3).

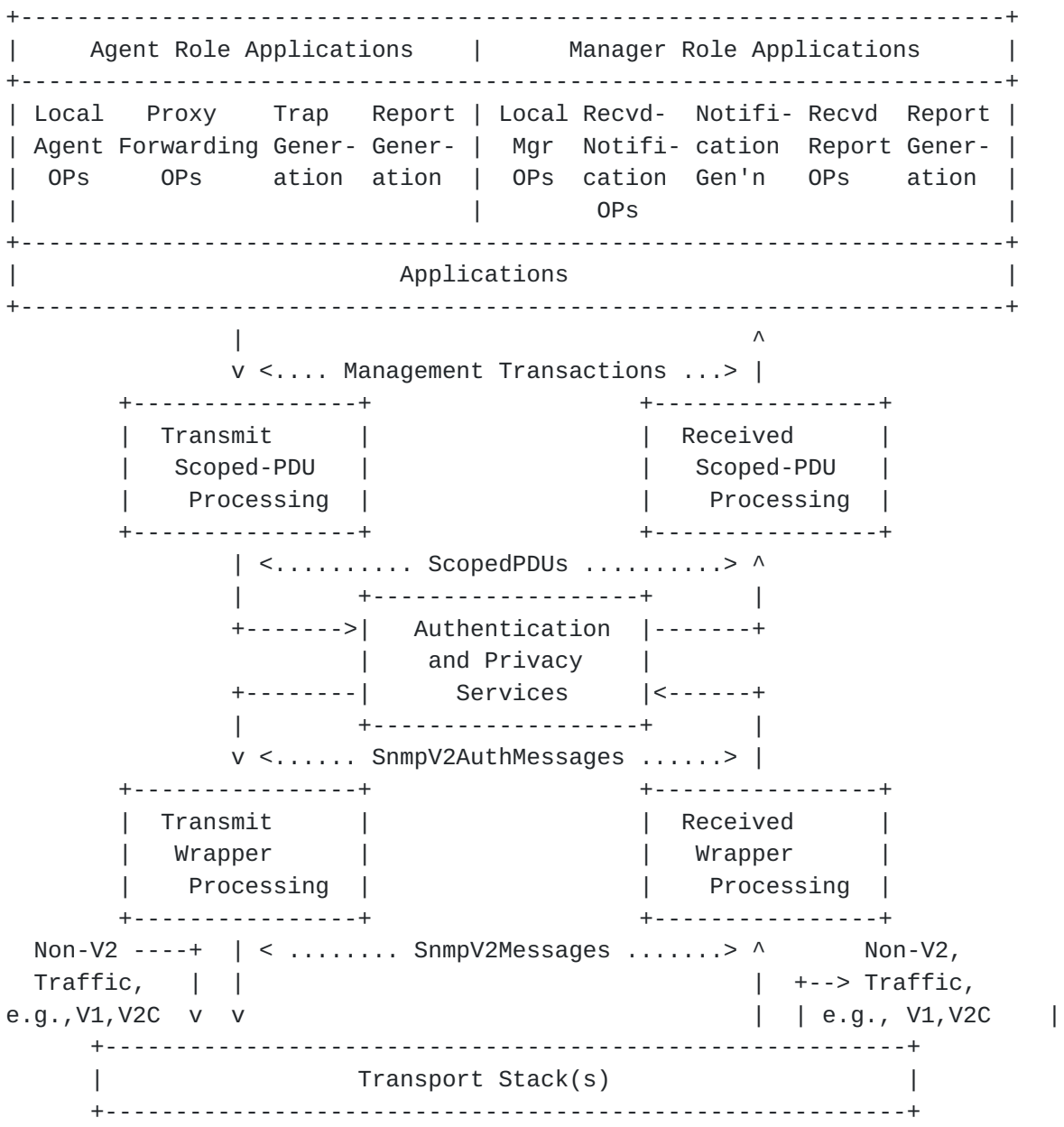

Figure 3: Conceptual Layering of SNMPv2 Applications and the Administrative Model

3.1.5. SNMPv2 Agent Role Applications

There are at least four functional services provided by SNMPv2 entities acting in an agent role. These include: Local Agent Operations, Proxy Forwarding Operations, Trap Generation, and Report Generation.

3.1.5.1. Local Agent Operations

Local Agent Operations are performed by an SNMPv2 entity acting in an agent role in response to request messages received from a logically remote SNMPv2 entity acting in a manager role. These operations manipulate data which are locally accessible by the agent. These request messages include several types of operations, including GetRequest, GetNextRequest, GetBulkRequest, and SetRequest operations.

Local Agent Operations almost always result in the production and transmission of an appropriate response, (sometimes including error or other exceptional indicators), termed the Response message.

3.1.5.2. Proxy Forwarding Operations

Proxy Forwarding Operations are performed by an SNMPv2 entity by receiving SNMPv2 messages from one or more SNMPv2 entities, analyzing them to determine a disposition, and then transmitting regenerated messages to one or more SNMPv2 entities.

Entities which perform Proxy Forwarding Operations are often bi-directional, forwarding messages both "upstream" and "downstream" but this is not necessarily the case. For example, this would not be the case for an entity which is able to perform forwarding operations of unidirectional messages, i.e., trap messages, either because of implementation restrictions or due to local policies such as security, configuration, or administration policies.

3.1.5.3. Trap Generation Operations

Trap Generation Operations are performed by an entity acting in an agent role when it generates a trap message and sends it to one or more SNMPv2 entities acting in a manager role.

3.1.5.4. Report Generation Operations

An SNMPv2 entity acting in an agent role performs Report Generation Operations when it prepares and sends error report messages to a logically remote SNMPv2 entity acting in a manager role.

These error reports are generated in response to received messages on behalf of the administrative model, i.e., as a byproduct of processing by the core of the administrative model as described herein or by one of the authentication and privacy services.

3.1.6. SNMPv2 Manager Role Applications

There are at least five functional services provided by SNMPv2 entities acting in a manager role. These include: Local Manager Operations, Received Notification Operations, Notification Generation, Received Report Operations, and Report Generation.

3.1.6.1. Local Manager Operations

Local Manager Operations are initiated by SNMPv2 entities acting in a manager role. They include the generation of requests containing queries or commands (i.e., GetRequest, GetNextRequest, GetBulkRequest, or SetRequest) to read or write management information. Local Manager Operations also include the receiving and processing of responses to these requests, i.e., messages containing a Response to a GetRequest, GetNextRequest, GetBulkRequest, or SetRequest.

3.1.6.2. Received Notification Operations

Received Notification Operations are performed by SNMPv2 entities acting in a manager role whenever they receive and process a trap message from an SNMPv2 entity acting in an agent role, i.e., SNMPv2-Trap. Received Notification Operations are also performed by SNMPv2 entities acting in a manager role whenever they receive and process a manager-to-manager message, i.e., one containing an InformRequest. The processing of an InformRequest normally includes generation of a confirmation message to indicate information was received, (e.g., a Response confirming an InformRequest.

3.1.6.3. Notification Generation by Managers

Notification Generation is performed by SNMPv2 entities acting in a manager role whenever they initiate a manager-to-manager message, i.e., one containing an InformRequest.

3.1.6.4. Received Report Operations

Received Report Operations are performed by SNMPv2 entities acting in a manager role whenever they receive, process, and [if appropriate] act upon reports received by the manager.

3.1.6.5. Report Generation by Managers

Report Generation by Managers is performed by SNMPv2 entities acting in a manager role whenever they generate and send a message containing an

error report, i.e., one containing a Report. A Report is initiated by a manager only in response to an InformRequest, and since InformRequests are initiated only by SNMPv2 entities acting in a manager role, a manager never sends a report to an agent. Consequently, an entity acting in an agent role does not receive and process reports.

3.2. SNMPv2 snmpID

SNMPv2 entities are named by an SNMPv2 snmpID [@ref adminmib]. An SNMPv2 snmpID is a 12 byte quantity which provides a unique identification of SNMPv2 entities throughout the administrative domain; preferably globally as well.

Each implementation of the administrative model described in this memo which initiates request operations or trap operations (including those which initiate InformRequest operations) is assigned an SNMPv2 snmpID. An implementation of the administrative model described in this memo which does not initiate request operations nor trap operations may be assigned an SNMPv2 snmpID, but this is not necessary.

An SNMPv2 entity which implements this administrative model typically includes a SNMPv2 protocol engine which is associated with one or more applications.

Two management applications are part of the same SNMPv2 entity if and only if they have the same value of snmpID. That is, multiple applications, e.g., an agent and a manager, or two management applications on a platform, are part of the same named entity if they have the same snmpID.

Hence, a set of coordinated management applications from a single vendor might share the same snmpID and thereby be a part of the same SNMPv2 entity whereas a set of decoupled applications, each of which having their own values for snmpID, would not be a part of the same SNMPv2 entity.

The snmpID is used within the administrative model to provide uniqueness for contexts and identities. For example, the local value of snmpID is sometimes used for authSnmpID or contextSnmpID when generating an SNMPv2 message at an SNMPv2 entity. The determination of the source of the snmpID to be used, i.e., remote, local, or that of a third party, is different for different transactions.

The snmpID value assigned to the contextSnmpID value in a message is always that snmpID value associated with the context name and the information in the variable bindings list. Accordingly, the snmpID value assigned to the contextSnmpID value in a Get, GetNext, GetBulk, or Set operation is always the snmpID value associated with the agent receiving the request. Similarly, the contextSnmpID value in the response to any of these operations is that same snmpID value. Further, the snmpID value assigned to the contextSnmpID value in a Trap notification operation is always the snmpID value associated with the agent originating the trap message.

The assignment of the contextSnmpID value for use in an InformRequest operation follows these same rules. Accordingly, the snmpID value assigned to the contextSnmpID value in an InformRequest may be that of the originating manager, the receiving manager, or that of a third party. For example, in the last case, manager 1 may tell manager 2 about what it has discovered about information contained within a context on agent a. In this case, the contextSnmpID value will be that snmpID value associated with agent a. The contextSnmpID value used in the response to an InformRequest is always the same as that contained in the request.

There are parallel sets of rules for the selection of authSnmpID values. The authSnmpID value is always set to the value of snmpID associated with the SNMPv2 entity which is considered to be "authoritative". For a Get, GetNext, GetBulk, or Set request operation, the "next hop" destination, i.e., the agent or the next intervening agent which provides proxy forwarding services is said to be authoritative, and is the source of the value of snmpID to be used for authSnmpID. This same value of authSnmpID is used for the responses to these request operations.

For trap notification operations, the snmpID value associated with the agent which originates the trap, or the agent which provides proxy forwarding services is said to be authoritative, and is the source of the value of snmpID to be used for authSnmpID.

For an inform notification request operation, the "next hop" destination, i.e., the destination manager or the next intervening agent which provides proxy forwarding services is said to be authoritative, and is the source of the value of snmpID to be used for authSnmpID. This same value of authSnmpID is used for the responses to these request operations.

3.3. SNMPv2 Identities

An identity is assumed by an SNMPv2 entity in order to restrict its operations (for security or other purposes) to an administratively defined subset of all possible SNMPv2 operations. Whenever an SNMPv2 entity processes an SNMPv2 message, it does so by operating as a SNMPv2 identity and is thereby restricted to the set of operations defined for that identity. The set of possible operations specified for an SNMPv2 identity may be overlapping or disjoint with respect to the sets of other SNMPv2 identities.

Each SNMPv2 identity is named by the pairing of an identityName and an authSnmpID. The identityName is derived from the authInfo sequence of a message and other parameters in accordance with the rules of the sPI in effect for the message. Similarly, the authInfo sequence of a message is derived from an identityName and other parameters in accordance with the rules of the sPI in effect for the message. These other parameters might include the authentication and privacy data in use on behalf of the identity, such as secret or public keys. The specifics of the mapping algorithm are dependent upon the authentication and privacy mechanisms and algorithms identified by the sPI.

An SNMPv2 identity is unique only when used in conjunction with both an sPI and an authSnmpID. For example, if managers M1, M2, M3, and M4 communicate with one another and with agents A1, A2, and A3 directly and via proxies P1, P2, and P3 as shown in the following figure, then the required knowledge about a prototypical identity "joe" for a given sPI is as depicted (See Figure 4.)

There are multiple instances of "joe" on each manager and each proxy node to allow for different security parameters for identity "joe", for example, different passwords on different nodes. It is through this capability that a change to the private security keys associated with "joe" might be updated and distributed through the administrative domain over a period of time. Configuration of agents is the easiest whereas managers and proxies require additional configuration.

In addition, the identity "joe" may have different security parameters for different values of sPI. For example, it is possible for "joe" to have one authentication key for use with one authentication and privacy service and a different key for another authentication and privacy service.

Indeed, it is even possible for there to be two different identities with the identityName of "joe" for two different values of sPI which

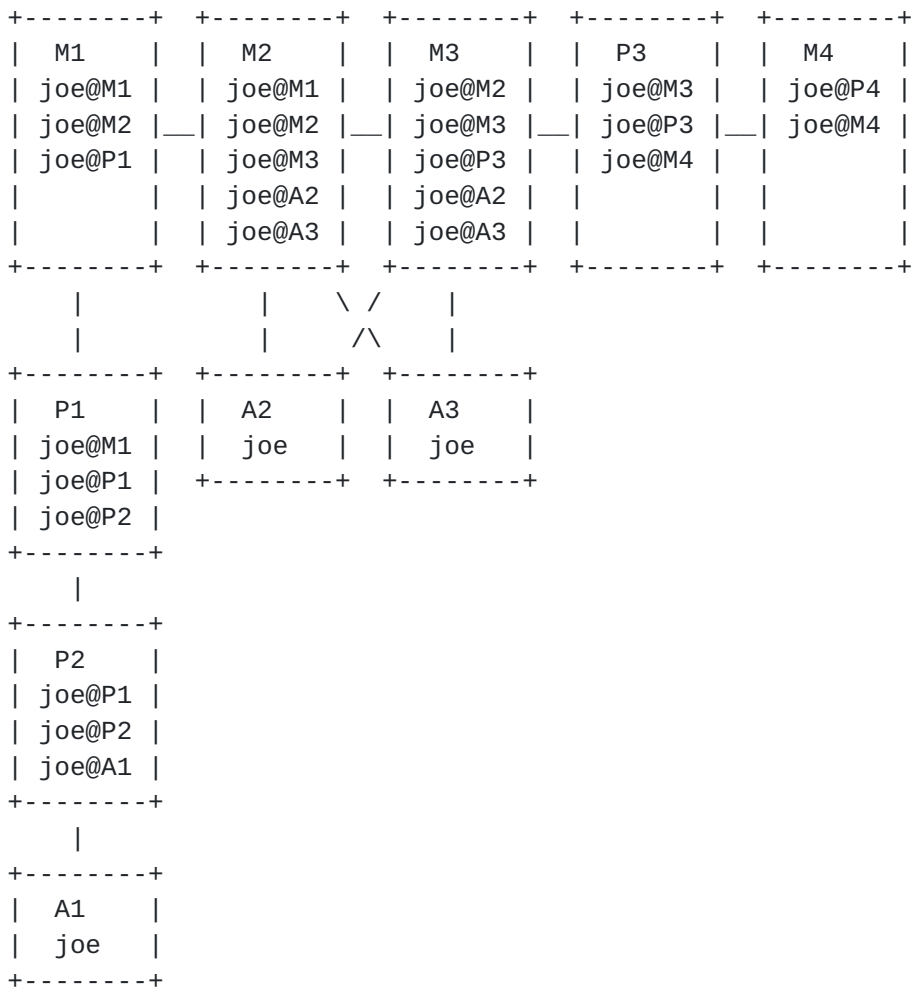

Figure 4: Inter-entity Identity Relationships

correspond to two different human users by the name of "joe". However, the prudent administrator will likely attempt to avoid such configurations as a matter of policy even though such configurations are possible via the model and underlying protocols.

It is worth mentioning that an identityName is unique within the authentication and privacy service for a given sPI on a simple agent, i.e., one which does not perform proxy forwarding operations. The authSnmpID is implicit in such systems.

3.4. SNMPv2 Entity

An SNMPv2 entity is an actual process or set of processes which performs management operations by generating and/or responding to SNMPv2 messages in the manner specified in [@ref protoops] and Section 4.3, Proxy Forwarding Operations. An SNMPv2 entity assumes the identity of a particular SNMPv2 identity when processing an SNMPv2 message.

An SNMPv2 entity is not required to process multiple protocol messages concurrently, regardless of whether such messages require it to assume the identity of the same or different SNMPv2 identities. Thus, implementation of an SNMPv2 entity to support more than one identity need not be multi-threaded. However, there may be situations where implementors may choose to use multi-threading.

Every SNMPv2 entity maintains a Local Configuration Datastore (LCD) which includes information on all identities which it uses to communicate, as well as other information (see below).

An SNMPv2 entity listens for incoming, unsolicited SNMPv2 messages on each transport service address for which it is configured to do so. It is a local matter whether an SNMPv2 entity also listens for SNMPv2 messages on any other transport service addresses. In the absence of any other information on where to listen, an SNMPv2 entity must listen on the transport service addresses corresponding to the standard transport-layer "ports" [@ref tm] on its local network-layer addresses.

3.5. View Subtree and Families

A view subtree is the set of all MIB object instances which have a common ASN.1 OBJECT IDENTIFIER prefix to their names. A view subtree is identified by the OBJECT IDENTIFIER value which is the longest OBJECT IDENTIFIER prefix common to all (potential) MIB object instances in that subtree.

For example, the subtree 1.3.6.1.2.1.1, which corresponds to the object system, identifies the objects within the system group of MIB-II [@ref v2mib4v2] as a view subtree.

A family of view subtrees is a pairing of an OBJECT IDENTIFIER value (called the family name) together with a bitstring value (called the family mask). The family mask indicates which sub-identifiers of the associated family name are significant to the family's definition.

For each possible managed object instance, that instance belongs to a particular view subtree family if both of the following conditions are true:

- the OBJECT IDENTIFIER name of the managed object instance contains at least as many sub-identifiers as does the family name, and

- each sub-identifier in the the OBJECT IDENTIFIER name of the managed object instance matches the corresponding sub-identifier of the family name whenever the corresponding bit of the associated family mask is non-zero.

When the configured value of the family mask is all ones, the view subtree family is identical to the single view subtree identified by the family name.

When the configured value of the family mask is shorter than required to perform the above test, its value is implicitly extended with ones. Consequently, a view subtree family having a family mask of zero length always corresponds to a single view subtree.

When the OBJECT IDENTIFIER prefix identifying a view subtree is longer than the OBJECT IDENTIFIER of an object type defined according to the SMI [@ref v2smi], then the use of such a view subtree for access control has granularity at the object instance level. Such granularity is considered beyond the scope of an SNMPv2 agent. As such, no implementation of an SNMPv2 agent is required to support values of viewSubtree [@ref adminmib] which have more sub-identifiers than is necessary to identify a particular leaf object type. However, it may choose to do so.

3.6. MIB View

A MIB view is a subset of the set of all instances of all object types defined according to the SMI [@ref v2smi] within an SNMPv2 context, subject to the following constraints:

- It is possible to specify a MIB view which contains the full set of all object instances within an SNMPv2 context.

- Each object instance within a MIB view is uniquely named by an ASN.1 OBJECT IDENTIFIER value.

As such, identically-named instances of a particular object type must be contained within different SNMPv2 contexts. That is, a particular

object instance name resolves within a particular SNMPv2 context to at most one object instance.

A MIB view is defined as a collection of view subtree families, where each view subtree family has a type. The type determines whether the view subtree family is included in, or excluded from, the MIB view.

A managed object instance is contained/not contained within the MIB view according to the view subtree families to which the instance belongs:

- If a managed object instance belongs to none of the relevant subtree families, then that instance is not in the MIB view.

- If a managed object instance belongs to exactly one of the relevant subtree families, then that instance is included in, or excluded from, the relevant MIB view according to the type of that subtree family.

- If a managed object instance belongs to more than one of the relevant subtree families, then that instance is included in, or excluded from, the relevant MIB view according to the type of a particular one of the subtree families to which it belongs. The particular subtree family is the one for which, first, the associated family name comprises the greatest number of sub-identifiers, and, second, the associated family name is lexicographically greatest.

3.7. SNMPv2 Context

An SNMPv2 context is a collection of management information accessible by an SNMPv2 entity. The collection of management information identified by a context is either proxy or non-proxy.

For a local SNMPv2 context which is realized by an SNMPv2 entity, that SNMPv2 entity uses locally-defined mechanisms to access the management information identified by the SNMPv2 context.

For a proxy SNMPv2 context, the SNMPv2 entity acts as a proxy SNMPv2 agent to access the management information identified by the SNMPv2 context. There is no requirement that an SNMPv2 entity acting in an agent role support any proxy contexts, i.e., support of proxy is not required for conformance with this specification.

3.7.1. Local SNMPv2 Context

A local context refers to a collection of managed (MIB) objects which logically belong to a single entity within a managed device, even if that device is constructed of multiple physical devices. When an SNMPv2 entity accesses that management information, it does so using locally-defined mechanisms.

A managed device may have multiple collections of managed objects belonging to multiple logical entities within the managed device, each local context has associated with it a "local entity" name. Further, because management information changes over time, each local context also has associated with it an associated temporal domain, termed its "local time". This allows, for example, one context to refer to the current values of a device's parameters, and a different context to refer to the values that the same parameters for the same device will have after the device's next restart.

3.7.2. Proxy SNMPv2 Context

A proxy relationship exists when a proxy SNMPv2 agent processes a received SNMPv2 message (a request or a response) by forwarding it to another entity according to the value of the context in the received message. Such a context is called a proxy SNMPv2 context. When an SNMPv2 entity processes management requests/responses for a proxy context, it is operating as a proxy SNMPv2 agent.

3.8. SNMPv2 PDUs and Operations

An SNMPv2 PDU is defined in [@ref protoops]. Each SNMPv2 PDU specifies a particular operation. The types of operation (see Table 1) are represented by the possible values of the ASN.1 tag for the appropriate PDU.

GetRequest	SetRequest	SNMPv2-Trap
GetNextRequest	Response	Inform
GetBulkRequest	Report	

Table 1: SNMPv2 Operation Types

3.9. SNMPv2 Message

A SNMPv2 message contains a single SNMPv2 PDU, is transmitted on behalf of an identity, and contains management information for an identified SNMPv2 context.

In particular, an SNMPv2 message may be

- a query (e.g., GetRequest, GetNextRequest, or GetBulkRequest),
- an indicative assertion (e.g., Response, InformRequest, or SNMPv2-Trap),
- an imperative assertion (e.g., SetRequest),
- a confirmation message to indicate information was received (e.g., a Response confirming an InformRequest), or
- an error report (e.g., Report).

An SNMPv2 message is constructed by creating an ASN.1 SEQUENCE, constructed from several sub components, some of which are also SEQUENCEs:

- The first portion is the version number, where

```
version ::= INTEGER {
                version-2(2)
            }
```

- The second portion is the mms, where

  mms ::= INTEGER (484..2147483647)

- The third portion is the security protocol identifier, where

  securityFlags ::= INTEGER (1..2147483647)

- The fourth portion is a sequence. This sequence consists of an sPI-dependent authInfo portion.

- The fifth portion contains a contextSnmpID, a contextName, and PDU. Depending upon the value of sPI, portions of some or all of these may be either plaintext or protected from disclosure (encrypted).

  where the syntax and semantics of the contents of the authInfo portion is designated by the value of sPI, and the PDU portion is an SNMPv2 PDU as defined in [@ref protoops].

More formally, the complete definition of an SNMPv2 message is:

```
SnmpV2Admin DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS
    PDUS
        FROM SNMPv2-PDU

        SnmpV2Message ::= SEQUENCE {
            version          -- version-2 for this RFC
                INTEGER {
                    version-2(2)
                },
            mms
                INTEGER (484..2147483647),
            securityFlags
                INTEGER (1..2147483647),
            authMessage
                ANY DEFINED BY securityFlags
        }

    -- Where the value contained within an authMessage is defined by the
    -- authentication and privacy service as selected by securityFlags.
    -- The auth/priv service MUST define an authMessage such that the
    -- decoded and unencrypted authMessage contains the following
elements.
    -- The auth/priv service is free to define the ASN.1 for the
authMessage,
```

```
    -- although it is suggested that the members be specified in the      |
    -- following order:                                                   |
    --                                                                    |
    --      authInfo ( AuthInfo )                                        |
    --      contextSnmpID ( OCTET STRING (SIZE(12)) )                    |
    --      contextName ( OCTET STRING )                                 |
    --      pdu ( PDUs )                                                 |
    --                                                                   |
    --  AuthInfo is defined by the authentication and privacy service.   |
END
```

3.10. SNMPv2 Access Control Policy

An SNMPv2 access control policy is a specification of a local access policy which authorizes access to an SNMPv2 context via an identity. In particular, an SNMPv2 access policy specifies the accessible MIB views within various SNMPv2 contexts.

The application of SNMPv2 access control policy is performed: on receipt of GetRequest, GetNextRequest, GetBulkRequest, and SetRequest operations.

A set of MIB definitions allows for the configuration of some SNMPv2-Trap and InformRequest operations which are also based upon MIB views within various SNMPv2 contexts, but this is independent of the access control policy expressed in the acTable [@ref adminmib]. These MIB definitions are not the exclusive mechanisms by which SNMPv2-Trap and InformRequest operations can be configured.

Note that application of SNMPv2 access control policy is not performed for messages containing Response nor Report operations.

4. Elements of Procedure

This section describes the procedures followed by an SNMPv2 entity in processing SNMPv2 messages. These procedures are independent of the particular authentication and privacy protocols that may be in use.

The procedure and data flow are depicted in the following diagrams which are provided for expository purposes and are not meant to dictate any particular implementation strategy.

A portion of an implementation of the model processes received messages, steers them to an appropriate authentication and privacy service, and, if they are deemed authentic, continues processing them until they are delivered to an SNMPv2 application for further disposition. Another portion of an implementation of the model generates messages from parameters provided by the application and found in the LCD, preparing them for transmission over the network, including applying an appropriate authentication and privacy service.

4.1. Generating a Message

This section describes the procedure followed by an SNMPv2 entity whenever an SNMPv2 message is to be transmitted by an SNMPv2 application on behalf of an SNMPv2 identity.

4.1.1. Data Flow

The following diagram illustrates the data flow when a message is generated (See Figure 5). It should be noted that, especially in the case of management station applications, the "applications" referenced in the procedures which follow may be isolated from the actual management station application by additional implementation-specific layers which add additional value.

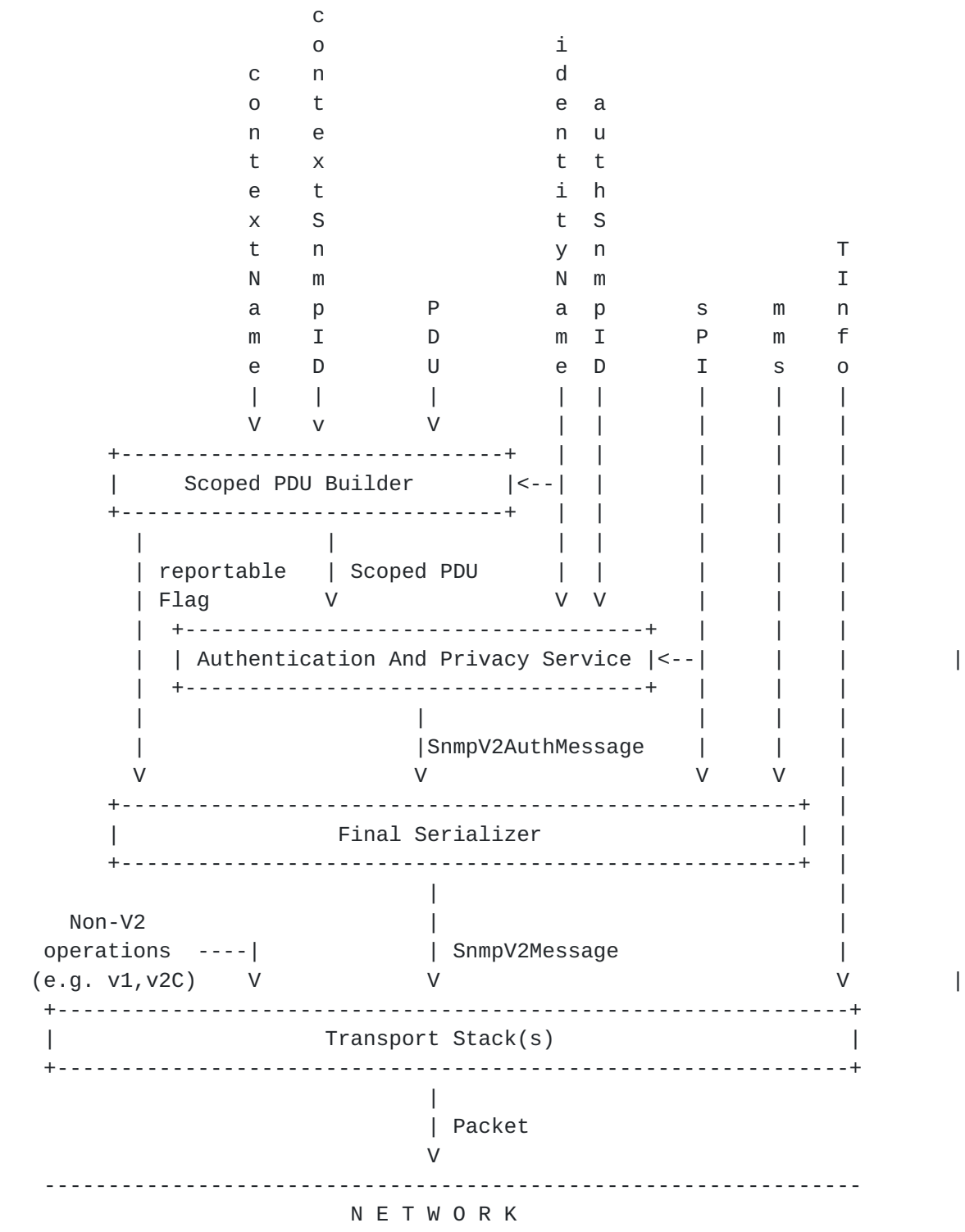

Figure 5: Data Flow for Generating a Message

The sending application provides several parameters, including the relevant values for: the destination transport domain and address; its maximum message size of received messages; the required sPI; the identityName and authSnmpID; a globally unique identification of a context, including its contextSnmpID and contextName; and a PDU indicating the desired operations and objects.

4.1.2. Procedure

The procedure is as follows.

(1) The contextName, contextSnmpID, and PDU are assembled into a ScopedPDU value.

(2) The PDU value is examined to determine the SNMPv2 operation type. If the SNMPv2 operation type indicates a GetRequest, GetNextRequest, GetBulkRequest, SetRequest, or InformRequest operation, then the reportableFlag is set to be true; and it is set to false for all other operation types, i.e., Response, SNMPv2-Trap, or Report operations.

(3) The required sPI, authSnmpID, identityName, and ScopedPDU are passed to the authentication and privacy service designated by the sPI.

(4) The selected authentication and privacy service follows its procedures using the provided parameters and those in the LCD to construct an appropriate authInfo portion which is prepended to the ScopedPDU, possibly encrypting all or part of the ScopedPDU and authInfo portion in accordance with the syntax and semantics defined in the specification associated with the value of sPI provided by the application; thereby producing an SnmpV2AuthMessage value. If there is an error or exception encountered in the authentication and privacy service, then the message cannot be sent and the requesting application is suitably advised.

(5) The transport information provided by the management application is examined to determine the transport stack to be used to reach the destination and the value of mms is reduced (never increased) to the smallest of those for the application, SNMPv2 protocol engine, and the transport stack. The serialization of the mms parameter is deferred as long as possible so that this minimum can be determined with a late binding.

(6) The provided value for sPI and the value of reportableFlag determined from the SNMPv2 operation type are combined to form a seurityFlags value.

(7) The resultant values of mms, securityFlags, and the SnmpV2AuthMessage are serialized (i.e., encoded) according to the conventions of [@ref tm], thereby producing an SnmpV2Message value.

(8) The serialized SNMPv2 message is transmitted using the transport address and transport domain provided by the management application.

Note that the above procedure does not include any application of any SNMPv2 access control policy (see Section 3.11).

4.2. Processing a Received Communication

This section describes the procedure followed by an SNMPv2 entity whenever a SNMPv2 message is received.

4.2.1. Data Flow

The following diagram illustrates the data flow when processing a received communication (See Figure 6).

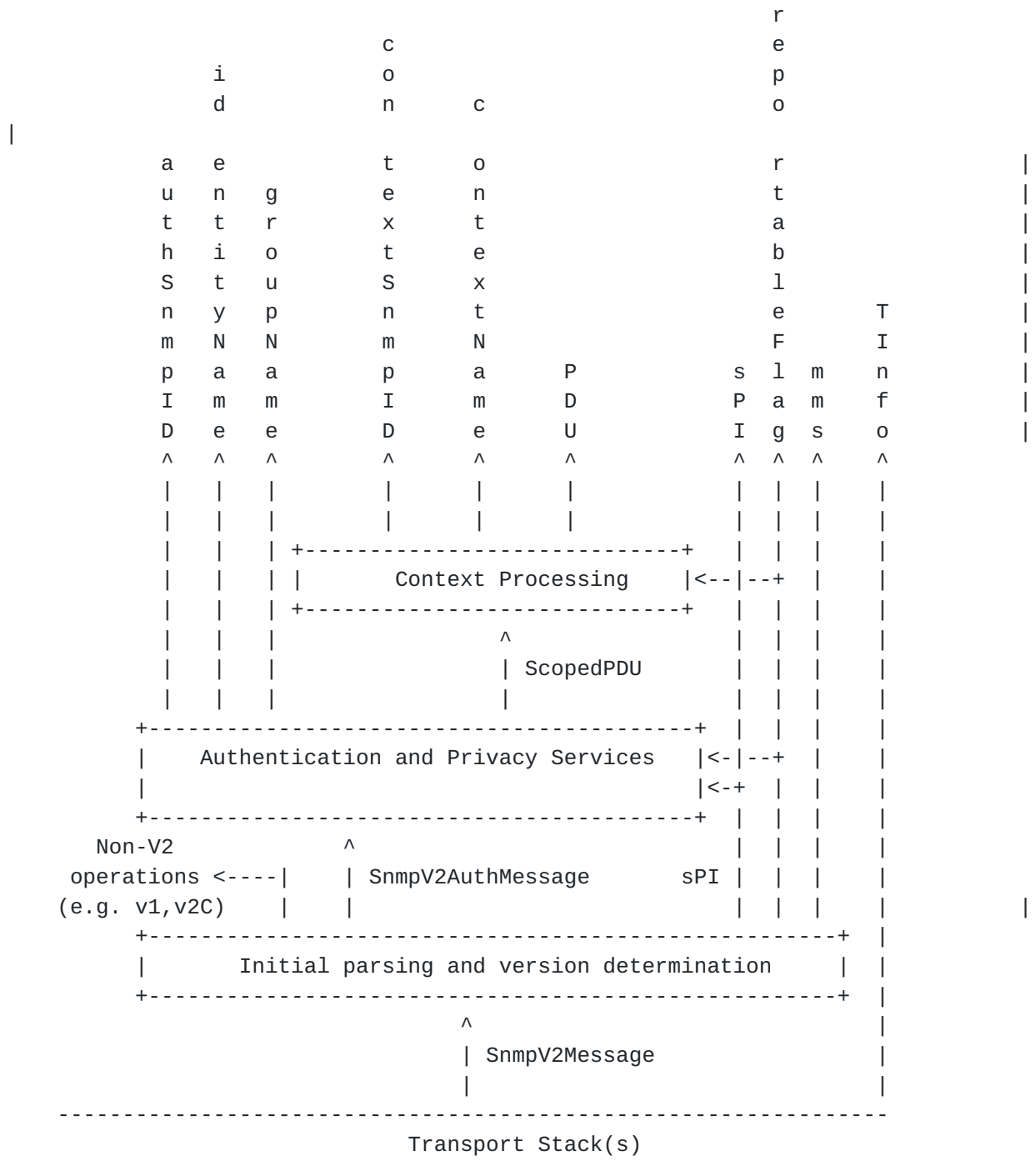

Figure 6: Data Flow for Received Message Processing

4.2.2. Procedure

The steps of the procedure are as follows.

(1) The snmpInPkts counter [@ref v2mib4v2] is incremented and an initial parse of the packet is performed. If the received message is not the serialization (according to the conventions of [@ref tm]) of an SnmpV2Message value, then that message is passed to another appropriate SNMP application running on the node, e.g., an SNMPv1 [@ref 1157] or SNMPv2C [@ref v2Cadmin] entity, if any, for further processing. Otherwise:

- If the first octet of the packet has the value hexadecimal 30, then the snmpInBadVersions counter [@ref v2mib4v2] is incremented, and the message is discarded without further processing.

- If the first octet of the packet is not the value hexadecimal 30, then the snmpInASNParseErrs counter [@ref v2mib4v2] is incremented, a report PDU is generated, and the message is discarded without further processing.

(2) The value of mms is extracted and saved.

(3) The values of sPI and reportableFlag are extracted and saved. If the sPI value is not implemented by this entity, then the value of v2AdminStatsUnknownSPI counter [@ref v2mib4v2] is incremented, a report PDU is generated, and the received message is discarded without further processing.

(4) The SnmpV2AuthMessage value is forwarded to the authentication and privacy service designated by the sPI. Processing within the designated authentication and privacy service may result in one or more error conditions, in which case a report PDU may be generated by the authentication and privacy service, and the received message is discarded without further processing.

However, if the snmpV2EnableAuthenTraps object [@ref v2mib4v2] is enabled, then the SNMPv2 entity sends authorizationFailure traps [@ref v2mib4v2] according to its configuration.

If the message is deemed authentic, the selected authentication and privacy service translates the SnmpV2AuthMessage value into an identityName, authSnmpID, groupName, and a plaintext (i.e., unencrypted) version of the serialized ScopedPDU.

(5) The LCD is consulted for information about the identity as named by the combination of the authSnmpID and identityName values. If information about the SNMPv2 identity is absent from the LCD, then the snmpStatsUnknownIdentities counter [@ref v2mib4v2] is incremented, a report PDU is generated, and the received message is discarded without further processing.

(6) The portion of the ScopedPDU containing context information is processed to extract the values for contextSnmpID and a contextName. If the serialized ScopedPDU value is not the serialization (according to the conventions of [@ref tm]) of a ScopedPDU value, then the received message is discarded without further processing, after the snmpInASNParseErrs counter [@ref v2mib4v2] is incremented and a report PDU is generated.

(7) The PDU portion of the ScopedPDU is processed. If the serialized PDU value is not the serialization (according to the conventions of [@ref tm]) of a PDU value, then the received message is discarded without further processing, after the snmpInASNParseErrs counter [@ref v2mib4v2] is incremented and a report PDU is generated.

(8) If the SNMPv2 operation type is either a Get, GetNext, GetBulk, or Set operation, then:

   a) If the receiving entity performs proxy forwarding operations and the context is such that it indicates a possible proxy operation, then the processing continues as described in Section 4.3, Proxy Forwarding Operations.

   b) If the identified context is such that it indicates a non-local operation, then the v2AdminStatsUnknownContexts counter [@ref v2mib4v2] is incremented, a report PDU is generated, and the received message is discarded without further processing.

   c) Otherwise, if the identified context indicates a configured local operation but the identified context is presently unavailable, perhaps because of the removal of a component from a hot-swappable chassis, then the received message is discarded without further processing, after the v2AdminStatsUnavailableContexts counter [@ref v2mib4v2] is incremented and a report PDU is generated.

d) The LCD is consulted for access rights authorized for communications on behalf of the identity concerning management information in the indicated SNMPv2 context for the particular SNMPv2 operation type.

e) If the SNMPv2 operation type is not among the authorized access rights, then the received message is discarded without further processing after generation and transmission of a response message. This response message is sent on behalf of the same identity. Its contextName, contextSnmpID, var-bind-list, and request-id values are identical to those of the received request. Its error-index portion is zero and its error-status portion is authorizationError [@ref protoops].

f) The information concerning the identityName, authSnmpID, contextSnmpID, contextName, and sending transport domain and address are cached for later use in generating a response message.

g) The management operation represented by the SNMPv2 operation type is performed by the SNMPv2 entity with respect to the relevant MIB view within the SNMPv2 context according to the procedures set forth in [@ref protoops], where the relevant MIB view is determined by the groupName, authSnmpID, sPI, contextSnmpID, contextName, and type of operation requested; where the relevant MIB view is:

SNMPv2 operation type	MIB View given by
read(get/getNext/getBulk)	acReadViewName
write(set)	acWriteViewName

(9) If the SNMPv2 operation type is a Response operation (to a Get, GetNext, GetBulk, Set, or Inform), then:

a) The request-id is extracted from the PDUs portion of the ScopedPDU received from the authentication and privacy service. The value of request-id is used to locate a corresponding entry in the cache of outstanding requests (both those generated locally and those which were the result of proxy forwarding operations, if any). The values of authSnmpID, identityName, and sPI received from the authentication and privacy service are compared with the cached values for the forwarded request. The values of

contextSnmpID and contextName found in the ScopedPDU are compared with the cached values for the forwarded request. If any of these values do not match, then the v2AdminStatsCacheMisses counter [@ref v2mib4v2] is incremented, the received message is discarded without further processing, and the cached entry is deleted.

b) If the located cached entry indicates this message is a response to a proxy forwarding operation, then processing continues as described in Section 4.3, Proxy Forwarding Operations, and if the located cached entry indicates this message is a response to a locally generated request, then processing continues as described by [@ref protoops].

(10) If the SNMPv2 operation type is a Trap notification operation, then

a) If the receiving entity performs proxy forwarding operations and the context is such that it indicates a possible proxy operation, then the processing continues as described in Section 4.3, Proxy Forwarding Operations.

b) Otherwise, processing of the message continues as described by [@ref protoops].

(11) If the SNMPv2 operation type is an Inform notification operation, then

a) If the receiving entity performs proxy forwarding operations and the context is such that it indicates a possible proxy operation, then the processing continues as described in Section 4.3, Proxy Forwarding Operations.

b) Otherwise, the receiving entity acknowledges receipt by sending a response as described in Section 4.4, Generating A Response, and [@ref protoops].

c) Processing of the message continues as described at [@ref protoops].

(12) If the SNMPv2 operation type is a Report operation, then

a) The request-id is extracted from the PDUs portion of the ScopedPDU received from the authentication and privacy service. The value of request-id is used to locate a corresponding entry in the cache of outstanding requests (both

those generated locally and those which were the result proxy forwarding operations, if any). (If no such entry is found, but the received request-id is 2147483647, then all entries are searched to attempt to locate an appropriate entry such that the values of authSnmpID, identityName, sPI, contextSnmpID, and contextName match.)
The values of authSnmpID, identityName, and sPI received from the authentication and privacy service are compared with the cached values for the forwarded request. The values of contextSnmpID and contextName found in the ScopedPDU are compared with the cached values for the forwarded request. If any of these values do not match, (or no suitable matching entry can be located in the special case when the request-id is 2147483647) then the v2AdminStatsCacheMisses counter [@ref v2mib4v2] is incremented,
the received message is discarded without further processing,
and the cached entry (if one was found) is deleted.

b) If the located cached entry indicates this message is a response to a proxy forwarding operation, then processing continues as described in Section 4.3, Proxy Forwarding Operations, and if the located cached entry indicates this message is a response to a locally generated request, then processing continues as described at [@ref protoops].

4.2.3. Common Constructs

For the sake of clarity and to prevent the above procedure from being even longer, the following details were omitted from the above procedure.

- Some situations in which an ASN.1 parsing error can occur were omitted (e.g., the possibility that a snmpV2AuthMessage portion is not a correct serialization of a SnmpV2AuthMessage value). The snmpInASNParseErrs
counter [@ref v2mib4v2] is incremented and a report PDU is generated whenever such an ASN.1 parsing error is discovered.

Some steps specify that the received message is discarded without further processing whenever a report PDU is generated. However, a report PDU must not be generated unless the reportableFlag is set, which ensures that a reportPDU is not generated due to the receipt of a report PDU. In addition, a generated report PDU must whenever possible contain the same request-id value as in the PDU contained in the received message. Meeting this constraint normally requires

the message to be further processed just enough so as to extract its request-id. Even in the case where the identity cannot be authenticated, an attempt must be made to extract the request-id by assuming that no encryption is in use by the identity. With this assumption, the only situation in which the request-id cannot be extracted is when an ASN.1 parsing error occurs.

For a possible procedure to invoke on receipt of a message with an SNMPv2 operation type of SNMPv1 trap, see [@ref coex].

4.3. Proxy Forwarding Operations

These procedures are implemented if and only if the entity also supports proxy forwarding operations.

The procedures are as follows.

(1) If the SNMPv2 operation type is either a Get, GetNext, GetBulk, or Set operation, then:

a) The LCD information is inspected to locate the entry in the proxyForwardingTable such that proxyDirection equals gnsb(1) for which there is a favorable comparison between the values of proxySPIIn, proxyAuthSnmpIDIn, proxyIdentityNameIn, proxyContextSnmpIDIn, and proxyContextName with the corresponding values received in the ScopedPDU or determined from the authentication and privacy service. The values of proxySPIOut, proxyAuthSnmpIDOut, proxyIdentityNameOut, proxyTransportLabelOut, and proxyPrivs are extracted from the located entry. If there is no such entry, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, and the received message is discarded without further processing.

b) The value of the proxyPrivs is compared with the SNMPv2 operation type. If the requested operation is not among the permitted operations, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, and the received message is discarded without further processing.

c) A new ScopedPDU is constructed.

Its PDUs portion is the same as the PDUs portion of the ScopedPDU request received from the authentication and privacy service except that the contained request-id is replaced by a unique value (this value will enable a subsequent response message to be correlated with this request). However, the new unique value should not equal 2147483647; this value is reserved.

Its contextSnmpID and contextName are the same as the corresponding values in the ScopedPDU received from the authentication and privacy service.

d) The appropriate authentication and privacy service, as selected by the extracted value of proxySPIOut is used to prepare an appropriate SnmpV2AuthMessage, using the extracted values of proxyAuthSnmpIDOut and proxyIdentityNameOut. If the value of proxySPIOut equals "1", which indicates that the next hop utilizes SNMPv1, then [@ref coex] defines an appropriate set of transformations to be applied. If a suitable SnmpV2AuthMessage cannot be constructed, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, and the received message is discarded without further processing.

e) Several values associated with the original request are cached for later use in generating a response message. These values include: the request-id of the original request and the request-id of the newly generated request, the sPI used to forward the message, transport domain and address of the source of the original request, the contextSnmpID and contextName found in the ScopedPDU of the original request, proxyAuthSnmpIDIn, identityNameIn, and sPI values received from the authentication and privacy service when processing the original request.

f) The extracted value of proxyTransportLabelOut is used to determine the transportDomain, transportAddress, and transportMMS for each destination.

For each destination, the value of mms associated with the request is reduced, to the smallest of the mms of the protocol engine of the proxy agent and the mms of the transport to be used to forward the request, if either is smaller than the received value for mms.

g) For each destination, the values for mms, sPI, and the SnmpV2AuthMessage are converted into a SnmpV2Message which is then sent via the transport layer. If the message cannot be constructed or is determined to be unforwardable, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, the received message is discarded without further processing, and the cached entry is deleted.

(2) If the SNMPv2 operation type is a Response operation (to a Get, GetNext, GetBulk, Set, or Inform operation), then:

a) The request-id is extracted from the PDUs portion of the ScopedPDU received from the authentication and privacy service. The value of request-id is used to locate a corresponding entry in the cache of outstanding requests. The values of authSnmpID, identityName, and sPI received from the authentication and privacy service are compared with the cached values for the forwarded request. The values of contextSnmpID and contextName found in the ScopedPDU are compared with the cached values for the forwarded request. If any of these values do not match, then the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, the received message is discarded without further processing, and the cached entry is deleted.

b) A new ScopedPDU is constructed.

Its PDUs portion is the same as the PDUs portion of the ScopedPDU response received from the authentication and privacy service except that the contained request-id is replaced by the cached value of the original request.

Its contextSnmpID and contextName are the same as the corresponding values in the ScopedPDU received from the authentication and privacy service.

c) The appropriate authentication and privacy service, as selected by the cached value of sPI for the original request is used to prepare an appropriate SnmpV2AuthMessage, using the cached values authSnmpID and identityName derived from the original request message. If the received value of sPI indicates a SNMPv1message, then [@ref coex] defines a suitable

set of transformations. If a suitable SnmpV2AuthMessage cannot be constructed, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, the received message is discarded without further processing, and the cached entry is deleted.

d) The value of mms associated with the response is reduced, to the smallest of the mms of the protocol engine of the proxy agent and the mms of the transport to be used to forward the request, if either are smaller than the received value for mms.

This step is unnecessary for the purpose of processing the response message currently in transit. However, this step is useful as means to learn the effective path-mms for the purpose of optimizing subsequent request-message size.

e) The values for mms, sPI, and the SnmpV2AuthMessage are converted into a SnmpV2Message and sent to the transport layer. The message is prepared and sent using the cached values of the original request sender's transport domain and address. If the message cannot be constructed or is determined to be unforwardable, the snmpProxyDrops [@ref v2mib4v2] is incremented, a report PDU is generated and sent to the source of the original request, the received response message is discarded without further processing, and the cached entry is deleted.

(3) If the SNMPv2 operation type is either SNMPv2-Trap or Inform, then the steps are identical to those given above for Get, GetNext, GetBulk, or Set request operations, except:

a) In step (1), the LCD entries which are consulted to locate the entry in the proxyForwardingTable are those such that proxyDirection equals trap(2) or inform(3), respecitively, instead of gnsb(1).

b) If the operation type is SNMPv2-Trap, a report PDU is never generated, the request-id need not be remapped, and no values are cached.

(4) If the SNMPv2 operation type is Report, then the request-id is extracted from the PDUs portion of the ScopedPDU received from the

authentication and privacy service. The value of request-id is used to locate a corresponding entry in the cache of outstanding requests. If the correlation is successful, the appropriate maintenance function (e.g., time synchronization, proxy error propagation, etc.) is invoked and the cached entry is deleted. Otherwise, the v2AdminStatsCacheMisses counter [@ref v2mib4v2] is incremented, and the received message is discarded without further processing.

If the result of such maintenance procedures determines that a proxy-forwarded request cannot be delivered to the proxied agent, then the snmpProxyDrops counter [14] is incremented and a report PDU is generated and transmitted to the transport address from which the original request was received. (Note that the receipt of a report PDU containing snmpProxyDrops as a varbind, is included among the reasons why a proxy-forwarded request cannot be delivered.)

4.4. Generating a Response

The procedure for generating a response to a SNMPv2 management request is identical to the procedure for generating a message (see Section 4.1), with these exceptions:

(1) The response is sent on behalf of the same values for sPI, identityName, authSnmpID, contextSnmpID, and contextName as were communicated by the original request.

(2) The PDUs value of the responding SnmpV2Message value is the response which results from performing the operation specified in the original PDUs value. The other relevant information is obtained, not from the LCD, but rather from information cached (in Step e) when processing the original message.

(3) The serialized Message value is transmitted using the transport address and transport domain from which its corresponding request originated - even if that is different from any transport information obtained from the LCD.

4.5. Report Generation Processing

While processing a received communication, the procedures may result in a determination that the received message is unacceptable, that an appropriate counter in the snmp object group of [@ref v2mib4v2] or the v2AdminStats object group of [adminmib] be incremented,

and that a message containing an error report should be generated.

All messages containing an error report sent as a result of the procedures defined in this memo shall have an sPI value of maint(3).

If possible, the request-id shall have the same value as the request-id field of the PDU in the message whose processing caused the error report. Otherwise, the value 2147483647 is used.

The error-status and error-index fields of a report PDU are always set to zero.

The variable-bindings field contains a single variable: the identity of the statistics counter which was incremented and its new value.

A report PDU is never sent in response to a report PDU. A protocol entity makes this determination by examining the reportableFlag of the received message and only sending report PDUs when the reportableFlag is set.

Several additional parameters must be provided to the protocol engine to send the reports.

The identityName is "report".

The authSnmpID and contextSnmpID are set to the value of snmpID for the local (generating) SNMPv2 entity.

The contextName is set to "default".

The values of contextSnmpID, contextName, and the report PDU are serialized to produce a ScopedPDU value.

The values of identityName, and authSnmpID are used to build an AuthInfo structure defined as follows. (The similarity to the structure used by the usecNoAuth(4) authentication and privacy service [@ref sec] is intentional to allow shared code paths in typical implementations).

```
AuthInfo ::= [9] IMPLICIT SEQUENCE {

    authSnmpID
        OCTET STRING (SIZE(12)),

    identityName
        OCTET STRING SIZE(6)),  -- equal to "report"
```

```
            pad1
                Integer32 (0),

            pad2
                Integer32 (0)

            pad3
                OCTET STRING (SIZE(0))
        }
```

The AuthInfo and ScopedPDU values are serialized to form a SnmpV2AuthMessage value.

A securityFlags object is built from an sPI value of maint(3) and a reportableFlag value of false.

The local value of maximum message size for the transport used by the original message is determined.

An SnmpV2Message is constructed from the maximum message size, securityFlags, and SnmpV2AuthMessage values.

The resulting message is forwarded to the source of the original received message.

5. Security Considerations

In order to participate in the administrative model set forth in this memo, SNMPv2 implementations must support local, non-volatile storage of the LCD. Accordingly, every attempt has been made to minimize the amount of non-volatile storage required.

6. Acknowledgements

To be provided here.

7. References

To be provided here.

8. Authors' Addresses

Tell U. Later
snmpv2@tis.com

Table of Contents

1 Introduction 4
1.1 A Note on Terminology 4
2 Overview 5
2.1 Management Information 6
2.2 Contexts 6
2.3 MIB Views 7
2.4 Access Rights 8
2.5 Authentication and Privacy 8
2.6 Security Mechanisms and Algorithms 9
2.7 Security Protocol Identifiers and the Reportable Flag 9
2.8 Identities and Group Names 11
2.9 Authorization: Access Control 12
2.10 Construction of an SNMPv2 Message 13
2.11 Authentication and Privacy Service 14
2.12 An SNMPv2 Entity's Local Configuration Datastore 15
2.13 Maintenance Functions 16
2.14 Proxy 16
2.15 Transparency Principle 18
3 Elements of the Model 20
3.1 SNMPv2 Applications 20
3.1.1 SNMPv2 Agent 20
3.1.2 SNMPv2 Manager 20
3.1.3 SNMPv2 Dual-Role Entity 20
3.1.4 SNMPv2 Application Layering within the Administrative Framework 22
3.1.5 SNMPv2 Agent Role Applications 23
3.1.5.1 Local Agent Operations 24
3.1.5.2 Proxy Forwarding Operations 24
3.1.5.3 Trap Generation Operations 24
3.1.5.4 Report Generation Operations 24
3.1.6 SNMPv2 Manager Role Applications 25
3.1.6.1 Local Manager Operations 25
3.1.6.2 Received Notification Operations 25
3.1.6.3 Notification Generation by Managers 25
3.1.6.4 Received Report Operations 25
3.1.6.5 Report Generation by Managers 25
3.2 SNMPv2 snmpID 26
3.3 SNMPv2 Identities 28
3.4 SNMPv2 Entity 30
3.5 View Subtree and Families 30
3.6 MIB View 31
3.7 SNMPv2 Context 32

3.7.1 Local SNMPv2 Context 33
3.7.2 Proxy SNMPv2 Context 33
3.8 SNMPv2 PDUs and Operations 34
3.9 SNMPv2 Message 34
3.10 SNMPv2 Access Control Policy 36
4 Elements of Procedure 37
4.1 Generating a Message 37
4.1.1 Data Flow 37
4.1.2 Procedure 39
4.2 Processing a Received Communication 40
4.2.1 Data Flow 40
4.2.2 Procedure 42
4.2.3 Common Constructs 46
4.3 Proxy Forwarding Operations 47
4.4 Generating a Response 51
4.5 Report Generation Processing 51
5 Security Considerations 54
6 Acknowledgements 54
7 References 54
8 Authors' Addresses 55