

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

19 March 1995

[draft-ietf-snmpv2-adminv2-ds-01.txt](#)

Jeffrey D. Case
SNMP Research, Inc.
case@snmp.com

James Galvin
Trusted Information Systems
galvin@tis.com

Keith McCloghrie
Cisco Systems, Inc.
kzm@cisco.com

Marshall T. Rose
Dover Beach Consulting, Inc.
mrose@dbc.mtview.ca.us

Steven Waldbusser
Carnegie Mellon University
waldbusser@cmu.edu

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.''

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

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).

1. Introduction

A management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and 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 Infrastructure for SNMPv2, to define how the administrative framework is applied to realize effective management in a variety of configurations and environments.

The model described here entails the use of distinct identities for peers that exchange SNMPv2 messages. Thus, it represents a departure from the community-based administrative model of the original SNMP [1]. By unambiguously identifying the source and intended recipient of each SNMPv2 message, this new strategy improves upon the historical community scheme both by supporting a more convenient access control model and allowing for effective use of asymmetric (public key) security protocols in the future.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, is

termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).

1.2. Change Log

For the 19 March version:

- The many changes adopted by the SNMPv2 Working Group.

Expires September 1995

[Page 3]

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

For the 1 November version:

- recast [RFC 1445](#) into an Internet-Draft,
- added Overview section,
- rewrote Elements of the Model section,
- fixed typos in the Elements of Procedure section,
- rewrote parts of the Application of the Model section, and retitled it as Usage Examples.
- tightened the definition of a proxy SNMPv2 agent, in order to remove confusion, and thereby deleted discussion of foreign proxies.
- changed "local party database" to "local party datastore", and introduced LPD as an acronym for it.

[2.](#) Overview

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.

[2.1.](#) 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.

For example, the managed object type, `ifDescr` [7], is defined as the description of a network interface. To identify the description of device-X's first network interface, three pieces of information are needed: device-X (the context), `ifDescr` (the managed object type), and "1" (the instance).

Management information often changes over time. 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. For example, where the value of a device parameter after the device's next reboot is to be different to its current value. 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.

[2.2.](#) Authorization: Access Rights and 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 [8, 3], 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 would 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.

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 be prohibited from write-

access to a particular context, while another might be allowed to perform any type of operation.

[2.3.](#) 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 requests for all items of management information within the one or more 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 on to other SNMP agents without regard for what managed object types are being accessed; for example, in order to forward 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;
- (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 for 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 model.

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).

2.4. Security

One aspect of security was discussed above: the ability to assign different access rights to different management applications. The enforcement of these access rights requires the means not only to identify the source of a request but also to authenticate such identification. Another security capability which SNMPv2 (optionally) provides is the ability to protect the data within an SNMPv2 message from disclosure (i.e., to encrypt the data). This is particularly useful when sensitive data (e.g., passwords, or security keys) are accessed via SNMPv2 requests.

Recommendations for which algorithms are best for authentication and privacy are subject to change. Such changes may occur as and 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 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 authentication as providing no authentication, and similarly, one of the algorithms for privacy as providing no protection against disclosure.

[2.5.](#) Parties

To provide these security capabilities, an SNMPv2 message needs to include the identity of its source and destination. Each such identity is specified as an SNMPv2 "party". Different parties have different security parameters, including the authentication and privacy algorithms and the security keys (if any) used. For any SNMPv2 message, the algorithm and parameters by which it is authenticated (or not) are those of its source party. Similarly, for any SNMPv2 message, the algorithm and parameters by which it is protected from disclosure (or not) are those of its destination party.

When an SNMPv2 message is generated, the appropriate parties must be chosen so that the message will have the desired level of authentication and privacy. The parties' algorithms and security keys are used to add authentication information to the message, and (if necessary) to encrypt it. When an SNMPv2 message is received, the destination party's privacy algorithm is used to decrypt it (if necessary), and the source party's authentication information is used to authenticate it.

Typically, an SNMPv2 party operates at one and only one location, although it is possible under some circumstances for an SNMPv2 party to operate from a different location. In any case, in order to authenticate the source party without having to rely on underlying transport mechanisms to provide authentication of transport addresses, it is necessary for a party to be globally unique, and have a globally unique identifier.

[2.6.](#) Authorization: Access Control

As described above, an SNMPv2 message is associated with one context and two parties, where the context determines the set of management information being accessed by the message, and the parties are the identities of the source and destination. These properties of the message are used for access control. Specifically, access control is specified as a set of local access policies, where each such policy is a valid party/party/context triple against which to compare a received

message. In addition, each such triple specifies the set of operations and the two MIB views (one for read access, the other for write access) which are allowed for that triple. Thus, if the context and the two parties specified by the received message are not a valid triple or the operation requested is not one of the operations allowed by the triple, then the requested access is denied.

Note that a local access policy (a party/party/context triple) is also called an ACL (for historical reasons).

[2.7.](#) 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 [\[2\]](#).) To construct an SNMPv2 message, a number of headers are prepended in front of the PDU (specifically, each "header" is added as part of a new ASN.1 SEQUENCE). The header which is first prepended to the PDU contains the context and the source and destination parties. The second prepended header contains whatever authentication information needs to be transmitted as part of the message. The third identifies the destination party which determines if and how the remainder of the message (the PDU and the first two prepended headers) are encrypted.

[2.8.](#) An SNMPv2 Entity's Local Party Datastore

An SNMPv2 entity is an SNMPv2 protocol implementation used by an SNMPv2 agent and/or by one or more management applications. The local parties at an SNMPv2 entity are those which operate locally, and the local contexts are those for which the SNMPv2 entity acts as a (proxy or non-proxy) SNMPv2 agent in responding to requests.

To implement the model described in the preceding sections, each SNMPv2 entity needs to retain its own set of information about contexts, parties, ACLs, and (in agents) views. This set of information is called the SNMPv2 entity's Local Party Datastore (LPD) because it is locally-stored information. Note, however, that the LPD contains information on both local and remote parties, local and/or remote contexts, local ACLs, and sometimes on remote ACLs as well.

In order to allow an SNMPv2 entity's LPD to be configured, and to allow the synchronization needed by the security algorithms before they can be used to communicate securely, the LPD needs to be accessible as managed

objects. A MIB module, the SNMPv2 Party MIB, to define these managed object types is contained in [\[4\]](#).

Expires September 1995

[Page 10]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

[2.9](#). 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 maintenance context and makes use of corresponding well-known maintenance parties. For example, error reporting ([Section 4.1](#)) and clock synchronization and secret update ([Sections 5.3](#) and [5.4](#) of [\[6\]](#)) are achieved by performing SNMP operations which access a context known as "internalContext".

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 parties and contexts specified in this document do not always apply to these maintenance functions (e.g., maintenance contexts are not uniquely named).

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.

[3.](#) Elements of the Model

This section provides a more formal description of the model.

[3.1.](#) SNMPv2 Party

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

Each SNMPv2 party has a set of attributes which includes the following:

partyIdentity -
the value which uniquely identifies the party.

partyTDomain -
the kind of transport service by which the party receives
management traffic. |
An example of a transport domain is snmpUDPDomain (SNMPv2 over UDP, |
using SNMPv2 parties).

partyTAddress -
the transport service address at which the party normally receives |
management traffic. This address is used in sending requests or |
notifications to the party |
(but not in responding to requests from the party). Note that

there is no requirement that the party transmits management traffic from this transport address.

partyMaxMessageSize -

the length in octets of the largest SNMPv2 message this party is prepared to accept.

partyAuthProtocol -

the authentication protocol by which all messages generated by the party are authenticated. In particular, the value 'noAuth' signifies that messages generated by the party are not authenticated.

Expires September 1995

[Page 12]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

partyAuthClock -

the party's authentication clock which represents a notion of the current time that is specific to the party. The significance of this component is specific to the authentication protocol.

partyAuthPrivate -

the party's private authentication key which is any secret value needed to support the authentication protocol. The significance of this component is specific to the authentication protocol.

partyAuthPublic -

the party's public authentication key which is any publically-disclosable value that may be needed to support the authentication protocol. The significance of this component is specific to the authentication protocol.

partyAuthLifetime -

the administrative upper bound on acceptable delivery delay for protocol messages generated by the party. The significance of this component is specific to the authentication protocol.

partyPrivProtocol -

the privacy protocol by which all protocol messages received by the party are protected from disclosure. In particular, the value 'noPriv' signifies that messages received by the party are not

protected from disclosure.

partyPrivPrivate -

the party's private privacy key which is any secret value needed to support the privacy protocol. The significance of this component is specific to the privacy protocol.

partyPrivPublic -

the party's public privacy key which is any publically-disclosable value that may be needed to support the privacy protocol. The significance of this component is specific to the privacy protocol.

An SNMPv2 entity, which supports only SNMPv2 parties for which the authentication protocol is noAuth and the privacy protocol is noPriv, is called non-secure.

[3.2.](#) SNMPv2 Entity

An SNMPv2 entity is an actual process which performs management operations by generating and/or responding to SNMPv2 protocol messages in the manner specified in [\[2\]](#). An SNMPv2 entity assumes the identity of a particular local SNMPv2 party 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 parties. Thus, implementation of an SNMPv2 entity to support more than one party need not be multi-threaded. However, there may be situations where implementors may choose to use multi-threading.

Every SNMPv2 entity maintains a Local Party Datastore (LPD) which includes information on all local SNMPv2 parties, and on those remote SNMPv2 parties which are known locally, as well as other information (see below).

An SNMPv2 entity listens for incoming, unsolicited SNMPv2 messages on each transport service address configured for a local SNMPv2 party. 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" [5] on its local network-layer addresses.

[3.3.](#) 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 manager initiates SNMPv2 management operations by the generation of appropriate SNMPv2 protocol messages or when it receives and processes trap and inform notifications.

[3.4.](#) 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 management operations in response to received SNMPv2 protocol messages (except for inform notifications) generated by an SNMPv2 manager.

Expires September 1995

[Page 14]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

[3.5.](#) 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.
- (2) (so-called) mid-level managers.

Proxy SNMPv2 agents only forward requests; they do not originate

requests. In contrast, mid-level managers often originate requests. +
(Note that the term proxy SNMPv2 agent does not include an SNMPv2 agent +
which translates SNMPv2 requests into the requests of some other +
management protocol; see [section 2.3.](#)) +

[3.6.](#) View Subtree

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.

When the OBJECT IDENTIFIER prefix identifying a view subtree is longer than the OBJECT IDENTIFIER of an object type defined according to the SMI [\[3\]](#), 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 [\[4\]](#) which have more sub-identifiers than is necessary to identify a particular leaf object type. However, access control information is also used in determining which SNMPv2 entities operating on behalf of management applications should receive trap notifications (Section 4.2.6 of [\[2\]](#)). As such, agent implementors might wish to provide instance-level granularity in order to allow SNMPv2 entity operating on behalf of management applications to use fine-grain configuration of trap notifications.

[3.7.](#) View Subtree Families

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:

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

[3.8.](#) MIB View

A MIB view is a subset of the set of all instances of all object types defined according to the SMI [\[3\]](#) within an SNMPv2 context, subject to the following constraints:

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

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:

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

An SNMPv2 agent's LPD includes information on the definitions of all MIB views specified for use with local non-proxy SNMPv2 contexts.

[3.9.](#) Proxy Context

A proxy relationship exists when a proxy SNMPv2 agent processes a received management request by forwarding it to another entity, solely according to the SNMPv2 context of the received message. Such a context is called a proxy SNMPv2 context. When an SNMPv2 entity processes management requests for a proxy context, it is operating as a proxy SNMPv2 agent.

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.

3.10. 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 non-proxy 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. Each non-proxy SNMPv2 context has a set of attributes which include the following:

contextIdentity -

the value which uniquely identifies the SNMPv2 context.

contextType -

a value which indicates that this is a non-proxy SNMPv2 context, and also indicates that it is of type local which is realized by the local SNMPv2 entity, or of type remote which is realized by some other SNMPv2 entity.

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

contextLocalEntity -

the value which identifies the local entity (e.g., a logical device local to the SNMPv2 entity which realizes the context) whose management information is contained in the SNMPv2 context.

contextLocalTime -

the temporal context of the management information contained in 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. Each proxy SNMPv2 context has a set of attributes which include the following:

contextIdentity -

the value which uniquely identifies the SNMPv2 context.

contextType -

a value which indicates that this is a proxy SNMPv2 context.

contextProxyDstParty -

the SNMPv2 party to which retrieval and modification SNMPv2 requests received for this context are forwarded.

contextProxySrcParty -

the SNMPv2 party to be used as the source SNMPv2 party when retrieval and modification SNMPv2 requests received for this context are forwarded.

contextProxyContext -

the SNMPv2 context to be used in forwarding retrieval and modification SNMPv2 requests received for this context.

The applicability of contextProxySrcParty and contextProxyContext are dependent upon the partyTDomain attribute of the contextProxyDstParty party.

An SNMPv2 entity's LPD includes information on all local contexts and on any remote contexts which are known locally.

[3.11.](#) SNMPv2 PDU

An SNMPv2 PDU is defined in [2]. 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

Note that a report PDU is only used as part of maintenance functions (see [section 4.1](#)).

[3.12.](#) SNMPv2 Message

A SNMPv2 message contains a single SNMPv2 PDU, is transmitted from a source SNMPv2 party to a destination SNMPv2 party, and contains management information for an SNMPv2 context which is accessible by the source/destination SNMPv2 party.

In particular, an SNMPv2 message may be

- o a query by the source party about information accessible by the destination party (e.g., GetRequest, GetNextRequest, or GetBulkRequest),
- o an indicative assertion to the destination party about information accessible by the source party (e.g., Response, InformRequest, or SNMPv2-Trap),

- o an imperative assertion by the source party about information accessible by the destination party (e.g., SetRequest), or
- o a confirmation to the destination party about information received by the source party (e.g., a Response confirming an InformRequest).

Starting from the PDU, an SNMPv2 message is constructed by creating three successive ASN.1 SEQUENCES, where:

Expires September 1995

[Page 20]

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

- the PDU is a component of the first SEQUENCE, which is called an SNMPv2 management communication,
- the first SEQUENCE is a component of the second SEQUENCE, which is called an SNMPv2 authenticated management communication,
- the second SEQUENCE is a component of the third SEQUENCE, which is called an SNMPv2 private management communication, and
- the third SEQUENCE corresponds exactly to an SNMPv2 message.

[3.13.](#) SNMPv2 Management Communication

An SNMPv2 management communication is an ASN.1 value with the following syntax:

```

SnpMgmtCom ::= [2] IMPLICIT SEQUENCE {
    dstParty
        OBJECT IDENTIFIER,
    srcParty
        OBJECT IDENTIFIER,
    context
        OBJECT IDENTIFIER,
    pdu
        PDUs
}

```

where:

- o The dstParty component identifies the destination SNMPv2 party of the SNMPv2 message. |
- o The srcParty component identifies the source SNMPv2 party of the SNMPv2 message. |
- o The context component identifies the SNMPv2 context containing the management information referenced by the SNMPv2 message. |
- o The pdu component is an SNMPv2 PDU as defined in [2]. |

Expires September 1995

[Page 21]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

[3.14.](#) SNMPv2 Authenticated Management Communication

An SNMPv2 authenticated management communication is an ASN.1 value with the following syntax:

```

SmpAuthMsg ::= [1] IMPLICIT SEQUENCE {
    authInfo
        ANY, -- defined by authentication protocol
    authData
        SmpMgmtCom
}

```

where:

- o The authInfo component contains the authentication information required in support of the authentication protocol used by the source SNMPv2 party. The detailed significance of the authentication information is specific to the authentication protocol in use, and its only use is in determining whether the communication is authentic or not. |
- o The authData component is an SNMPv2 management communication. |

[3.15.](#) SNMPv2 Private Management Communication

An SNMPv2 private management communication is an ASN.1 value with the following syntax:

```
SnmpPrivMsg ::= [1] IMPLICIT SEQUENCE {  
    privDst  
        OBJECT IDENTIFIER,  
    privData  
        [1] IMPLICIT OCTET STRING  
}
```

where:

- o The privDst component identifies the destination SNMPv2 party of the SNMPv2 message.
- o The privData component is the (possibly encrypted) serialization (i.e., encoding according to the conventions of [\[5\]](#)) of an SNMPv2 authenticated management communication.

Expires September 1995

[Page 22]

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

[3.16.](#) 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 a pair of SNMPv2 parties. In particular, an SNMPv2 access policy specifies the authorized types of SNMPv2 operations and the accessible MIB views.

Each such local access policy, called an ACL (for historical reasons), has six attributes:

acContext -

the context - an SNMPv2 context which identifies the management information on which requested management operations are to be performed;

acTarget -

the target - a SNMPv2 party for which the SNMPv2 context is a local

or proxy context;

acSubject -

the subject - a SNMPv2 party for which the SNMPv2 context is a remote context;

acPrivileges -

the access privileges - the types of SNMPv2 operations on the particular context that are authorized for SNMPv2 messages between the subject and the target;

acReadViewIndex -

the read MIB view - the (sub-)set of management information within the context to which the subject is authorized to have read-access via the target;
and

acWriteViewIndex -

the write MIB view - the (sub-)set of management information within the context to which the subject is authorized to have write-access via the target.

An SNMPv2 entity's LPD includes information on all ACLs containing local access control policies. An SNMPv2 manager may also include remote ACLs in its LPD in order to determine which operations are authorized by particular parties for a particular remote context.

The application of SNMPv2 access control policy is performed:

- o on receipt of GetRequest, GetNextRequest, GetBulkRequest, and SetRequest operations; and
- o prior to transmission of SNMPv2-Trap and Inform operations (the procedures by which this access-control is performed are specified in [2]).

Note that application of SNMPv2 access control policy is never performed for Response nor Report operations.

[4.](#) Maintenance Functions

Each SNMPv2 entity MUST provide an internalParty and internalContext for
use with, and only with, maintenance functions, as defined in the this
document.

+

+

+

The internalParty has these characteristics: +

partyIdentity	0.0	+
partyMaxMessageSize	484	+
partyAuthProtocol	noAuth	+
partyPrivProtocol	noPriv	+

The internalContext is local to any SNMPv2 entity acting in an agent role, and has these characteristics: +

contextIdentity	0.0	+
contextType	local (to agent)	+

Usage of internalParty and internalContext are effectively governed by a pseudo-ACL: +

acTarget	internalParty	+
acSubject	internalParty	+
acContext	internalContext	+
acReadViewIndex	<pseudo-view>	+
acWriteViewIndex	0	+
acPrivileges	35 (Get, GetNext & GetBulk)	+

where <pseudo-view> is statically defined to be all instances of all objects in exactly two subtrees: snmpStats [7] and snmpParties [4]. +

Note that internalParty, internalContext, the pseudo-ACL and the pseudo-view all have a StorageType of "readOnly" [4], in that they always exist and cannot be modified. Further, internalParty and internalContext MUST NOT be accessible to entities outside of an SNMPv2 entity itself. Consequently, even though they are conceptually represented in the LPD, neither internalParty nor internalContext are visible through the SNMPv2 Party MIB [4], neither are the pseudo-ACL and pseudo-view that they use. +

It should be noted that the use of a fixed context-identity (internalContext) is contrary to the architectural principles of the administrative framework: in SNMPv2, management information is always uniquely identified by a combination of context local-entity, context +

local-time, object type, and object instance. However, in the interests of both simplicity and the reuse of infrastructure, maintenance functions are provided "outside" of the administrative infrastructure available to applications which make use of an SNMPv2 entity. It must be emphasized that a management communication using internalContext belongs to a logically different protocol than SNMP, just as the ICMP is logically a different protocol from the IP. Thus, use of internalParty and internalContext, except for the purpose of performing maintenance functions, is prohibited.

[4.1.](#) Error Reporting

While processing a received communication, an SNMPv2 entity may determine that the message is unacceptable (see [Section 5.2](#)). In this case, the appropriate counter from the snmpStats group [\[7\]](#) is incremented and the received message is discarded without further processing or reply.

If an SNMPv2 entity acting in the agent role makes such a determination, then after incrementing the appropriate counter, it is required to generate a report PDU and to send it to the transport address which originated the received message. (Note, however, that the transmission of report PDUs may be disabled through the use of the managed object, snmpV2EnableReports [\[7\]](#).)

Report-PDU ::=

[\[8\]](#)

IMPLICIT PDU

The report PDU is always transmitted within an SNMPv2 message for which the components of the SnmpMgmtCom have these values:

srcParty = internalParty

dstParty = internalParty

context = internalContext

If the agent is able to determine the request-id field of the received PDU, then it uses that value for the request-id field of the report PDU. Otherwise, the value 2147483647 is used.

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

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

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

There are two cases in which a report PDU must not be sent by an SNMPv2 entity acting in the agent role: if the snmpStats30Something counter is incremented; or, if the received message was tagged as a report PDU. +

[5.](#) 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.

[5.1.](#) Generating a Request

This section describes the procedure followed by an SNMPv2 entity whenever either a management request or notification is to be transmitted by an SNMPv2 party. (Note: in this procedure, the values retrieved from the LPD may be overridden at the SNMPv2 entity's discretion by values obtained through other local knowledge.)

- (1) A `SnmpMgmtCom` value is constructed for which the `srcParty` component identifies the originating party, for which the `dstParty` component identifies the receiving party, for which the `context` component identifies the desired SNMPv2 context, and for which the `pdu` component represents the desired management operation.
- (2) The LPD is consulted to determine the authentication protocol and other relevant information for the originating and receiving SNMPv2 parties.
- (3) A `SnmpAuthMsg` value is constructed with the following properties:

Its `authInfo` component is constructed according to the authentication protocol specified for the originating party.

In particular, if the authentication protocol for the originating SNMPv2 party is identified as `noAuth`, then this component corresponds to the OCTET STRING value of zero length.

Its `authData` component is the constructed `SnmpMgmtCom` value.

- (4) The LPD is consulted to determine the privacy protocol and other relevant information for the receiving SNMPv2 party.
- (5) A SnmpPrivMsg value is constructed with the following properties:
 - Its privDst component identifies the receiving SNMPv2 party.
 - Its privData component is the (possibly encrypted)

Expires September 1995

[Page 28]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

serialization of the SnmpAuthMsg value according to the conventions of [\[5\]](#).

In particular, if the privacy protocol for the receiving SNMPv2 party is identified as noPriv, then the privData component is unencrypted. Otherwise, the privData component is processed according to the privacy protocol.

- (6) The constructed SnmpPrivMsg value is serialized (i.e., encoded) according to the conventions of [\[5\]](#).
- (7) The serialized SnmpPrivMsg value is transmitted using the transport address and transport domain (as specified in the LPD) for the receiving SNMPv2 party.

Note that the above procedure does not include any application of any SNMPv2 access control policy (but see [Section 3.15](#)).

[5.2](#). Processing a Received Communication

This section describes the procedure followed by an SNMPv2 entity whenever a SNMPv2 message is received. This procedure is independent of the transport service address at which the message was received.

- (1) The snmpStatsPackets counter [\[7\]](#) is incremented. If the received message is not the serialization (according to the conventions of [\[5\]](#)) of an SnmpPrivMsg value, then that message is discarded without further processing. (If the first octet of the packet has the value hexadecimal 30, then the snmpStats30Something counter [\[7\]](#) is incremented prior to discarding the message; otherwise the

snmpStatsEncodingErrors counter [7] is incremented and a report PDU is generated.)

- (2) The LPD is consulted for information about the receiving SNMPv2 party identified by the privDst component of the SnmpPrivMsg value.
- (3) If information about the receiving SNMPv2 party is absent from the LPD, or indicates that the receiving party's operations are not performed by the local SNMPv2 entity, then the snmpStatsUnknownDstParties counter [7] is incremented, a report PDU is generated, and the received message is discarded without further processing.

Expires September 1995

[Page 29]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

- (4) An ASN.1 OCTET STRING value is constructed (possibly by decryption, according to the privacy protocol in use) from the privData component of said SnmpPrivMsg value.

In particular, if the privacy protocol recorded for the party is noPriv, then the OCTET STRING value corresponds exactly to the privData component of the SnmpPrivMsg value.

- (5) If the OCTET STRING value is not the serialization (according to the conventions of [5]) of an SnmpAuthMsg value, then the received message is discarded without further processing, after the snmpStatsEncodingErrors counter [7] is incremented and a report PDU is generated.
- (6) If the dstParty component of the authData component of the obtained SnmpAuthMsg value is not the same as the privDst component of the SnmpPrivMsg value, then the received message is discarded without further processing, after the snmpStatsDstPartyMismatches counter [7] is incremented and a report PDU is generated.
- (7) The LPD is consulted for information about the originating SNMPv2 party identified by the srcParty component of the authData component of the SnmpAuthMsg value.
- (8) If information about the originating SNMPv2 party is absent from

the LPD, then the received message is discarded without further processing, after the snmpStatsUnknownSrcParties counter [7] is incremented and a report PDU is generated.

- (9) The obtained SnmpAuthMsg value is evaluated according to the authentication protocol and other relevant information associated with the originating and receiving SNMPv2 parties in the LPD.

In particular, if the authentication protocol is identified as noAuth, then the SnmpAuthMsg value is always evaluated as authentic.

- (10) If the SnmpAuthMsg value is evaluated as unauthentic, then the relevant counter (e.g., snmpStatsWrongDigestValues [7]) is incremented, a report PDU is generated, the received message is discarded without further processing, and if the snmpV2EnableAuthenTraps object [7] is enabled, then the SNMPv2 entity sends authorizationFailure traps [7] according to its configuration (Section 4.2.6 of [2]).

Expires September 1995

[Page 30]

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

- (11) The SnmpMgmtCom value is extracted from the authData component of the SnmpAuthMsg value.
- (12) The LPD is consulted for information about the SNMPv2 context identified by the context component of the SnmpMgmtCom value.
- (13) If information about the SNMPv2 context is absent from the LPD, then the received message is discarded without further processing, after the snmpStatsUnknownContexts counter [7] is incremented and a report PDU is generated.
- (14) The SNMPv2 operation type is determined from the ASN.1 tag value associated with the PDUs component of the SnmpMgmtCom value.
- (15) If the SNMPv2 operation type indicates that the message contains a report PDU, then appropriate maintenance procedures are invoked.

In particular, when a proxy SNMPv2 agent receives a report PDU from a proxy destination party, and the result of the invoked

maintenance procedures determines that a proxy-forwarded request cannot be delivered to the proxy destination party, then snmpStatsProxyDrops [7] 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 snmpStatsProxyDrops [7] as a varbind, is included among the reasons why a proxy-forwarded request cannot be delivered.]

(16) If the SNMPv2 operation type is either 32, 8, 2, or 1 (i.e., GetBulk, Set, GetNext or Get), then:

- a) if the LPD information indicates that the SNMPv2 context is of type remote, the received message is discarded without further processing, after the snmpStatsUnknownContexts counter [7] is incremented and a report PDU is generated.
- b) the LPD is consulted for the access privileges authorized for communications concerning management information in the indicated SNMPv2 context from the originating SNMPv2 party to the receiving SNMPv2 party, and for the associated MIB view for the particular SNMPv2 operation type.
- c) if the SNMPv2 operation type is not among the authorized access privileges, then the received message is discarded without further processing after generation and transmission

of a response message. This response message is directed to the originating SNMPv2 party on behalf of the receiving SNMPv2 party. Its context, var-bind-list and request-id components are identical to those of the received request. Its error-index component is zero and its error-status component is authorizationError [2].

- d) The information extracted from the LPD concerning the originating SNMPv2 party, the destination SNMPv2 party, and the SNMPv2 context, together with the sending transport address of the received message is cached for later use in generating a response message.

- e) if the LPD information indicates the SNMPv2 context is of type local, then the management operation represented by the SnmpMgmtCom value is performed by the receiving SNMPv2 entity with respect to the relevant MIB view within the SNMPv2 context according to the procedures set forth in [2], where the relevant MIB view is:

SNMPv2 operation type	MIB View given by
-----	-----
32, 2, or 1 (get/getNext/getBulk)	acReadViewIndex
8 (set)	acWriteViewIndex

- f) if the LPD information indicates the SNMPv2 context is of type proxy, then:

- i. the values of contextProxyContext, contextProxyDstParty and contextProxySrcParty associated with the SNMPv2 context of the received message are extracted from the LPD. If any of these extracted values have the value { 0 0 } or if any of the indicated parties and context are not present with an active status in the LPD, then snmpStatsProxyDrops [7] is incremented, a report PDU is generated, and the received message is discarded.
- ii. a new SNMPv2 message is constructed: the pdu component of the SnmpMgmtCom is copied from that in the received message 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); and the dstParty, srcParty and context components are set to the extracted values of contextProxyDstParty,

contextProxySrcParty and contextProxyContext, respectively.

- iii. the information cached in step 16d above is augmented with the request-id of the received message as well as the request-id and context of the constructed message.

iv. the constructed message is forwarded to the transport address associated with the party given by the extracted value of contextProxyDstParty.

(17) If the SNMPv2 operation type is either 128, 64 or 4 (i.e., SNMPv2-Trap, Inform, or Response), then:

- a) if the LPD information indicates the SNMPv2 context is of type local, then the received message is discarded without further processing, after the snmpStatsUnknownContexts counter [7] is incremented and a report PDU is generated.
- b) if the LPD information indicates the SNMPv2 context is of type remote, then the management operation represented by the SnmpMgmtCom value is performed by the receiving SNMPv2 entity according to the procedures set forth in [2].
- c) if the LPD information indicates the SNMPv2 context is of type proxy and the SNMPv2 operation type is 4 (i.e., a Response), then:
 - i. the request-id is extracted from the pdu component of the received SnmpMgmtCom. The SNMPv2 context and extracted request-id are used to correlate this response message to the corresponding values for a previously forwarded request by inspecting the cache of information as augmented in substep iii of step 16f above. If no such correlated information is found, then the received message is discarded without further processing.
 - ii. a new SNMPv2 message is constructed: the pdu component of the SnmpMgmtCom is copied from that in the received message except that the contained request-id is replaced by the value saved in the correlated information from the original request; and the dstParty, srcParty and context components are set to the saved values of the original request's originating party, destination party and context, respectively.

iii. the constructed message is forwarded to the transport

- address saved in the correlated information as the sending transport address of the original request.
- iv. the correlated information is deleted from the cache of information.
- d) if the LPD information indicates the SNMPv2 context is of type proxy, and the SNMPv2 operation type is 64 (i.e., Inform), then:
- i. a unique request-id is selected for use by all forwarded copies of this request. This value will enable a subsequent response message to be correlated with this request.
 - ii. all contexts in the LPD are located for which contextProxyContext has the same value as the context of the received message, and for which contextProxyDstParty has the same value as the originating party of the received message.
 - iii. for each located context, all ACLs in the LPD are located for which the value of acContext references the located context, and for which the access privileges allow Inform operations.
 - iv. for each located ACL, a new SNMPv2 message is constructed: the pdu component of the SnmpMgmtCom is copied from that in the received message except that the contained request-id is replaced by the value selected in step i) above; the dstParty, srcParty, and context components are set to the values of acSubject, acTarget and acContext respectively from the located ACL.
 - v. for each such constructed SNMPv2 message, the information cached in step d) above is augmented with the request-id of the received message as well as the request-id and context of the constructed message, and the constructed message is forwarded to the transport address associated with acSubject from the located ACL.

- e) if the LPD information indicates the SNMPv2 context is of type proxy, and the SNMPv2 operation type is 128 (i.e., SNMPv2-Trap), then:
- i. all contexts in the LPD are located for which contextProxyContext has the same value as the context of the received message, and for which contextProxyDstParty has the same value as the originating party of the received message.
 - ii. for each located context, all ACLs in the LPD are located for which the value of acContext references the located context, and for which the access privileges allow SNMPv2-Trap operations.
 - iii. for each located ACL, a new SNMPv2 message is constructed: the pdu component of the SnmpMgmtCom is copied from that in the received message; the dstParty, srcParty, and context components are set to the values of acSubject, acTarget and acContext respectively from the located ACL.
 - iv. each such constructed SNMPv2 message is forwarded to the transport address associated with acSubject from the located ACL, and the instance of snmpTrapNumbers [7] which corresponds to the acSubject party is incremented.

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 an authData component is not a correct serialization of a SnmpMgmtCom value). The snmpStatsEncodingErrors counter [7] 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 and until the SNMPv2 operation type can be determined, so as to ensure that a report PDU 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 these constraints normally requires the message to be further processed just enough so as to extract its SNMPv2 operation type and request-id. Even in the case where the receiving party is unknown, an attempt must be made to extract the SNMPv2 operation type and request-id by assuming that the privacy protocol of the unknown party is noPriv. With this assumption, the only situation in which the SNMPv2 operation type and request-id cannot be extracted is when an ASN.1 parsing error occurs.

- All generation of report PDUs is suppressed if disabled by the managed object, snmpV2EnableReports [7].

For a possible procedure to invoke on receipt of a message with an SNMPv2 operation type of 16, see [section 3.1.2](#) (step 2) of [9].

[5.3.](#) Generating a Response

The procedure for generating a response to an SNMPv2 management request is identical to the procedure for transmitting a request (see Section 5.1),
with these exceptions:

- (1) In Step 1, the dstParty component of the responding SnmpMgmtCom value is taken from the srcParty component of the original SnmpMgmtCom value; the srcParty component of the responding SnmpMgmtCom value is taken from the dstParty component of the original SnmpMgmtCom value; the context component of the responding SnmpMgmtCom value is taken from the context component of the original SnmpMgmtCom value; and, the pdu component of the responding SnmpMgmtCom value is the response which results from performing the operation specified in the original SnmpMgmtCom value.
- (2) In Step 2, the authentication protocol and other relevant information for the originating and receiving SNMPv2 parties is obtained, not from the LPD, but rather from information cached (in Step 16d) when processing the original message.
- (3) In Step 7, the serialized SnmpPrivMsg value is transmitted using

the transport address and transport domain from which its corresponding request originated - even if that is different from the transport information recorded in the LPD.

6. Usage Examples

6.1. Non-Secure Minimal Agent Configuration

This section presents an example configuration for a minimal, non-secure SNMPv2 agent that interacts with an SNMPv2 manager. Table 2 presents information about the local access policy known by the minimal agent; Table 3 presents the same information as known by the manager. Table 4 presents information on the SNMPv2 parties known to both the minimal agent and the manager.

Target:	gracie
Subject:	george
Context:	device5
Privileges:	Get/GetNext/GetBulk/SNMPv2-Trap
ReadView:	10
WriteView:	0

Table 2: Minimal Agent's Access Information

Target:	gracie
Subject:	george
Context:	device5
Privileges:	Get/GetNext/GetBulk/SNMPv2-Trap

Table 3: Manager's Access Information for Minimal Agent

Identity	gracie (agent)	george (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
Auth Prot	noAuth	noAuth
Auth Priv Key	""	""
Auth Pub Key	""	""
Auth Clock	0	0
Auth Lifetime	0	0
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 4: Party Information for Minimal Agent

As shown in Table 4, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity gracie; the example manager operates at UDP port 2001 at IP address 1.2.3.5 using the identity george. At minimum, a non-secure SNMPv2 agent implementation must provide for administrative configuration (and non-volatile storage) of the identities and transport addresses of two SNMPv2 parties: the local and remote peers.

Suppose that the SNMPv2 manager at 1.2.3.5 wishes to interrogate management information about the SNMPv2 context named "device5", by issuing an SNMPv2 GetNext request message. The manager consults its LPD

(Table 3) to discover that management information for context "device5" is held by the party named gracie, and that the managing party george can be used to access that context. The manager further consults its LPD (Table 4) to obtain the parameters of the two parties: george and gracie. Because the authentication protocol for the party george is recorded as noAuth, the GetNext request message generated by the manager is not authenticated. Because the privacy protocol for the party gracie is noPriv, the GetNext request message is not protected from disclosure. Rather, it is simply assembled, serialized, and transmitted to the transport address (IP address 1.2.3.4, UDP port 161) associated in the manager's LPD with the party gracie.

When the GetNext request message is received at the agent, the identity of the destination party (gracie) is extracted from the message, and the receiving entity consults its LPD. Because the privacy protocol for the

party gracie is recorded as noPriv, the received message is assumed not to be protected from disclosure. Similarly, the identity of the source party (george) is extracted, and the LPD is consulted. Because the authentication protocol for the party george is recorded as noAuth, the received message is immediately accepted as authentic.

The received message is processed only if the agent's LPD authorizes GetNext request operations by the remote party george to the local party gracie with respect to the SNMPv2 context "device5". The LPD information presented as Table 2 does authorize such operations (as well as Get and GetBulk operations).

During the processing of the received request, each specified item of management information is accessed only as authorized by the relevant MIB view. The LPD information in Table 2 authorizes access for GetNext operations (as well as to Get and GetBulk operations), and authorizes retrievals operations to the read MIB view identified by view-index 10.

To complete the processing of the request, the agent generates a Response message which references the SNMPv2 context "device5" and identifies the source party as gracie (since gracie was the request's destination

party), and the destination party as george (since george was the request's source party). The agent now consults its LPD. Because the authentication protocol for gracie is noAuth, the generated Response message is not authenticated. Because the privacy protocol for the party george is noPriv, the Response message is not protected from disclosure. The Response message is transmitted to the transport address from which the corresponding request originated - without regard for the transport address associated with george in the LPD.

When the generated Response is received by the manager, the identity of the destination party (george) is extracted from the message, and the manager consults its LPD. Because the privacy protocol for the party george is recorded as noPriv, the received Response is assumed not to be protected from disclosure. Similarly, the identity of the source party (gracie) is extracted, and the LPD is consulted. Because the authentication protocol for the party gracie is recorded as noAuth, the received Response is immediately accepted as authentic. Finally, since the received message is a Response operation, this message is not subject to access control.

[6.2.](#) Secure Minimal Agent Configuration

This section presents an example configuration for a secure, minimal SNMPv2 agent that interacts with a single SNMPv2 manager. Table 5 presents information about the local access policy known by the minimal agent; Table 6 presents the same information as known by the manager. Table 7 presents information about the SNMPv2 parties known to both the minimal agent and the manager.

The interaction of manager and agent in this configuration is very similar to that sketched above for the non-secure minimal agent - except that all protocol messages are authenticated and protected from disclosure. This example requires encryption of all SNMPv2 messages. Another example having an additional pair of SNMPv2 parties could support the exchange of non-secret information in authenticated messages without incurring the cost of encryption.

Target:	ollie
Subject:	stan
Context:	device6
Privileges:	Get/GetNext/GetBulk/Set/SNMPv2-Trap
ReadView:	16
WriteView:	17

Table 5: Secure Minimal Agent's Access Information

Target:	ollie
Subject:	stan
Context:	device6
Privileges:	Get/GetNext/GetBulk/Set/SNMPv2-Trap

Table 6: Manager's Access Information for Secure Minimal Agent

Identity	ollie (agent)	stan (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"0123456789ABCDEF"	"GHIJKL0123456789"
Auth Pub Key	"config-file"	"querty"
Auth Clock	0	0
Auth Lifetime	300	300
Priv Prot	desPrivProtocol	desPrivProtocol
Priv Priv Key	"MNOPQR0123456789"	"STUVWX0123456789"

Table 7: Party Information for Secure Minimal Agent

As shown in Table 7, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity ollie; the example manager operates at UDP port 2001 at IP address 1.2.3.5 using the identity stan. At minimum, a secure SNMPv2 agent implementation must provide for administrative configuration (and non-volatile storage) of relevant information about two SNMPv2 parties: one local and the other a remote peer.

Both ollie and stan authenticate all messages that they generate by using the SNMPv2 authentication protocol v2md5AuthProtocol and their distinct, private authentication keys. Although these private authentication key values ("0123456789ABCDEF" and "GHIJKL0123456789") are presented here and in other examples in this section for expository purposes, knowledge of private authentication keys is normally confined to those portions of the protocol implementation that require it, and not made known to human beings.

When using the v2md5AuthProtocol, the public authentication key for each SNMPv2 party is not used in authentication and verification of SNMPv2 exchanges. Also, because the v2md5AuthProtocol is symmetric in character, the private authentication key for each party must be known to each SNMPv2 entity with which authenticated communication is desired. In contrast, asymmetric (public key) authentication protocols would not depend upon sharing of a private key for their operation.

All protocol messages generated for transmission to the party stan are encrypted using the desPrivProtocol privacy protocol and the private key "STUVWX0123456789"; they are decrypted upon reception according to the same protocol and key. Similarly, all messages generated for

transmission to the party ollie are encrypted using the desPrivProtocol protocol and private privacy key "MNOPQR0123456789"; they are correspondingly decrypted on reception. As with authentication keys, knowledge of private privacy keys is normally confined to those portions of the protocol implementation that require it, and not made known to

human beings.

6.3. MIB View Configurations

This section provides example configurations of MIB views illustrating both simple view subtrees, and more complicated uses of included and excluded families of view subtrees.

View Index	Type	Family Name	Family Mask
5	included	internet	'H

Table 8: View Definition for Minimal Agent

Table 8 illustrates the definition of a MIB view required for a minimal SNMPv2 entity that has a single MIB view containing all instances of all MIB objects defined within the SNMPv2 Network Management Framework. The first column identifies the View Index by which this view is referenced by local access policies. Note that the value of View Index has only local significance. The type (included) signifies that any MIB object instance belonging to the view subtree family is contained within the MIB view. The family name is internet, and the zero-length family mask value signifies that the relevant subtree family corresponds to the single view subtree rooted at that node.

Another example of MIB view definition (see Table 9) is that of a SNMPv2 entity having multiple distinct MIB views. The MIB view associated with View Index 7 comprises all instances of all MIB objects defined within the SNMPv2 Network Management Framework, except those pertaining to the administration of SNMPv2 parties. In contrast, the MIB view associated with View Index 8 contains only MIB object instances defined in the system group of the Internet-standard MIB together with those object instances by which SNMPv2 parties are administered.

View Index	Type	Family Name	Family Mask	
7	included	internet	'H	
7	excluded	snmpParties	'H	
8	included	system	'H	
8	included	snmpParties	'H	

Table 9: Definition of Multiple Views

A more complicated example of MIB view configuration illustrates the use of view subtree families (see Table 10).

In this example, the MIB view associated with the View Index 42 includes all object instances in the system group of the Internet-standard MIB together with information related to the second network interface attached to the managed device. However, this interface-related information does not include the speed of the interface. The family mask value 'FFA0'H in the second table entry signifies that a MIB object instance belongs to the relevant subtree family if the initial prefix of its name places it within the ifEntry portion of the registration hierarchy and if the eleventh sub-identifier of its name is 2. The MIB object instance representing the speed of the second network interface belongs to the subtree families represented by both the second and third entries of the table, but that particular instance is excluded from the MIB view for View Index 42 because the lexicographically greater of the relevant family names appears in the table entry with type excluded.

The MIB view for View Index 49 is also defined in this example, to include all object instances in the icmp group of the Internet-standard MIB, together with all information relevant to the fifth network interface attached to the managed device. In addition, the MIB view for View Index 49 includes the number of octets received on the fourth attached network interface.

View Index	Type	Family Name	Family Mask	
42	included	system	'H	
42	included	{ ifEntry 0 2 }	'FFA0'H	
42	excluded	{ ifSpeed 2 }	'H	
49	included	icmp	'H	
49	included	{ ifEntry 0 5 }	'FFA0'H	
49	included	{ ifInOctets 4 }	'H	

Table 10: More Elaborate View Definitions

While, as suggested by the examples above, a wide range of MIB view configurations are efficiently supported by the use of view subtree families, prudent MIB design can sometimes further reduce the size and complexity of the most likely MIB view definitions. On one hand, it is critical that mechanisms for MIB view configuration impose no absolute constraints either upon the access policies of local administrations or upon the structure of MIB namespaces; on the other hand, where the most common access policies are known, the configuration costs of realizing those policies may be slightly reduced by assigning to distinct portions of the registration hierarchy those MIB objects for which local policies most frequently require distinct treatment.

Notwithstanding the above, instance level granularity is optional (see [section 3.6](#)).

6.4. Proxy Configuration

This section presents an example configuration that supports SNMPv2 proxy operations - indirect interactions between an SNMPv2 agent and an SNMPv2 manager that are mediated by a second SNMPv2 agent which performs the role of a proxy SNMPv2 agent.

Tables 11 and 12 present information the SNMPv2 manager's LPD: Table 11 for access control policies, and Table 12 for SNMPv2 parties. Tables 13, 14 and 15 present information in the proxy SNMPv2 agent's LPD: Table 13 for SNMPv2 parties, Table 14 for proxy contexts, and Table 15 for access control policies.

(These configurations are simplified for clarity: actual configurations may require additional parties in order to support other operations and other managers.)

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

Target	Subject	Context	Privileges
chico	groucho	ducksoup	Get/GetNext/GetBulk/SNMPv2-Trap

Table 11: Access Information for Management Station

Identity	groucho (manager)	chico (proxy agent)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 2002	1.2.3.5, 161
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"0123456789ABCDEF"	"GHIJKL0123456789"
Auth Pub Key	"this"	"that"
Auth Clock	0	0
Auth Lifetime	300	300
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 12: Party Information for Management Station

Internet Draft

SNMPv2 Administrative Infrastructure

March 1995

Identity	groucho (manager)	chico (proxy agent)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 2002	1.2.3.5, 161
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"0123456789ABCDEF"	"GHIJKL0123456789"
Auth Pub Key	"this"	"the-other"
Auth Clock	0	0
Auth Lifetime	300	300
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""
Identity	harpo (proxy dst)	zeppo (proxy src)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.6, 161	1.2.3.5, 161
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"MNOPQR0123456789"	"STUVWX0123456789"
Auth Pub Key	""	""
Auth Clock	148448	5764309
Auth Lifetime	300	300
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 13: Party Information for Proxy Agent

Context	Type	Proxy Dest.	Proxy Source	Proxy Context	
ducksoup	proxy	harpo	zeppo	bigstore	
bigstore	proxy	0.0	0.0	0.0	

Table 14: Contexts known to Proxy Agent

Expires September 1995

[Page 46]

Internet Draft SNMPv2 Administrative Infrastructure March 1995

Target	Subject	Context	Privileges	
chico	groucho	ducksoup	Get/GetNext/GetBulk/SNMPv2-Trap	
zeppo	harpo	bigstore	0	

Table 15: Access Information for Proxy Agent

As shown in Table 13,
the proxy SNMPv2 agent party operates at UDP port 161 at IP address [1.2.3.5](#) using the party identity chico; the example manager operates at UDP port 2002 at IP address 1.2.3.4 using the identity groucho; the proxy source party operates at UDP port 161 at IP address 1.2.3.5 using the party identity zeppo, and
the proxy destination party operates at UDP port 161 at IP address [1.2.3.6](#) using the party identity harpo.

Messages generated by all four SNMPv2 parties are authenticated by using the authentication protocol v2md5AuthProtocol and their individual private authentication keys.

Table 14 shows the SNMPv2 contexts known to the proxy SNMPv2 agent. In particular, the SNMPv2 context ducksoup is a proxy context that is satisfied when the SNMPv2 party zeppo communicates with the SNMPv2 party harpo and references the SNMPv2 context bigstore.

In order to interrogate the proxied device associated with the context ducksoup (see Table 11), the SNMPv2 manager constructs an SNMPv2 message from party groucho containing a GetNext request for the SNMPv2 context ducksoup, and transmits it to the party chicho operating (see Table 12) at UDP port 161 and IP address 1.2.3.5. This request is authenticated using the private authentication key "0123456789ABCDEF".

When that request is received by the party chicho, the originator of the message is verified as being the party groucho by using local knowledge (see Table 13) of the private authentication key "0123456789ABCDEF". Because party groucho is authorized to issue GetNext (as well as Get and GetBulk) requests to party chicho for the SNMPv2 context ducksoup by the relevant access control policy (Table 15), the request is accepted. Because the LPD (Table 14) indicates that the SNMPv2 context ducksoup is a proxy context, the request is satisfied by its translation into a corresponding SNMPv2 GetNext request with a unique request-id, source party zeppo, destination party harpo and with SNMPv2 context bigstore. This new communication is authenticated using

the private authentication key "STUVWX0123456789" and transmitted to party harpo at the IP address 1.2.3.6.

When this new request is received by the party harpo, authentication and access control is performed in the normal way, and an SNMPv2 Response message representing the results of the query is generated from the party harpo to party zeppo referencing SNMPv2 context bigstore. This response communication is authenticated using the private authentication key "MNOPQR0123456789" and transmitted to party zeppo at IP address [1.2.3.5](#) (the source address for the corresponding request).

When this response is received by party zeppo, the source of the message is verified as being the party harpo by using local knowledge (see Table 13) of the private authentication key "MNOPQR0123456789". Because this message contains a Response operation, it is accepted without application of access control. Its request-id is used to correlate this request to the previously forwarded GetNext such that the appropriate response can be constructed. This response is constructed to have the request-id of the original GetNext request, an SNMPv2 context of

ducksoup, the source party chico and the destination party groucho; it is authenticated using the chico's private authentication key "GHIJKL0123456789" and is transmitted to IP address 1.2.3.4 (the source address for the original request).

When this response is received by the party groucho, the source of the message is verified as being the party chico by using local knowledge (see Table 13) of the private authentication key "GHIJKL0123456789". Because this message contains a Response operation, it is accepted without application of access control, and the interrogation is complete.

6.5. Public Key Configuration

This section presents an example configuration predicated upon a hypothetical security protocol. This hypothetical protocol would be based on asymmetric (public key) cryptography as a means for providing data origin authentication (but not protection against disclosure). This example illustrates the consistency of the administrative model with public key technology, and the extension of the example to support protection against disclosure should be apparent.

Identity	ollie (agent)	stan (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2004
Auth Prot	pkAuthProtocol	pkAuthProtocol
Auth Priv Key	"0123456789ABCDEF"	""
Auth Pub Key	"abcdefghijklmnop"	"0123456789012345"
Auth Clock	0	0
Auth Lifetime	300	300
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 16: Party Information for Public Key Agent

The example configuration comprises a single SNMPv2 agent that interacts with a single SNMPv2 manager. Tables 16 and 17 present information about SNMPv2 parties as known to the agent and manager, respectively. Table 5 presents information about a local access policy known by the agent, and Table 6 presents information about a local access policy known by the manager.

Identity	ollie (agent)	stan (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2004
Auth Prot	pkAuthProtocol	pkAuthProtocol
Auth Priv Key	""	"GHIJKL0123456789"
Auth Pub Key	"abcdefghijklmnop"	"0123456789012345"
Auth Clock	0	0
Auth Lifetime	300	300
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 17: Party Information for Public Key Management Station

As shown in Table 16, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity ollie; the example manager operates at UDP port 2004 at IP address 1.2.3.5 using

the identity stan. Both ollie and stan authenticate all messages that they generate by using the hypothetical SNMPv2 authentication protocol pkAuthProtocol and their individual private authentication keys.

In most respects, the interaction between manager and agent in this configuration is almost identical to that in the example of the minimal, secure SNMPv2 agent described above. The most significant difference is

that neither SNMPv2 party in the public key configuration has knowledge of the private key by which the other party authenticates its transmissions. Instead, for each received authenticated SNMPv2 communication, the identity of the originator is verified by applying an asymmetric cryptographic algorithm to the received message together with the public authentication key for the originating party. Thus, in this configuration, the agent knows the manager's public key ("0123456789012345") but not its private key ("GHIJKL0123456789"); similarly, the manager knows the agent's public key ("abcdefghijklmnop") but not its private key ("0123456789ABCDEF").

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

8. Acknowledgements

The authors wish to acknowledge the contributions of the SNMPv2 Working Group in general. In particular, the following individuals

Dave Arneson (Cabletron),
Uri Blumenthal (IBM),
Doug Book (Chipcom),
Maria Greene (Ascom Timeplex),
Deirdre Kostik (Bellcore),
Dave Harrington (Cabletron),
Jeff Johnson (Cisco Systems),
Brian O'Keefe (Hewlett Packard),
Dave Perkins (Bay Networks),
Randy Presuhn (Peer Networks),
Shawn Routhier (Epilogue),
Bob Stewart (Cisco Systems),
Kaj Tesink (Bellcore).

deserve special thanks for their contributions.

9. References

- [1] Case, J., Fedor, M., Schoffstall, M., Davin, J., "Simple Network Management Protocol", STD 15, [RFC 1157](#), SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.
- [2] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.
- [3] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Structure of Management Information for Version 2 of the Simple Network

- Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995. |
- [4] Case, J., Galvin, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Party MIB for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, March 1995. SNMP Research, Inc., Trusted Information Systems, Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, |
- [5] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995. |
- [6] Case, J., Galvin, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Security Protocols for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Trusted Information Systems, Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995. |
- [7] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995. |
- [8] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).
- [9] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995. +

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

10. Authors' Addresses

Jeffrey D. Case
SNMP Research, Inc.
3001 Kimberlin Heights Rd.
Knoxville, TN 37920-9716
US

+
+
+
+
+

Phone: +1 615 573 1434
Email: case@snmp.com

+
+

James M. Galvin
Trusted Information Systems, Inc.
3060 Washington Road, Route 97
Glenwood, MD 21738

Phone: +1 301 854-6889
EMail: galvin@tis.com

Keith McCloghrie
Cisco Systems, Inc.
170 West Tasman Drive,
San Jose CA 95134-1706.

Phone: +1 408 526 5260
Email: kzm@cisco.com

Marshall T. Rose
Dover Beach Consulting, Inc.
420 Whisman Court
Mountain View, CA 94043-2186
US

+
+
+
+
+

Phone: +1 415 968 1052
Email: mrose@dbc.mtview.ca.us

+
+

Steven Waldbusser

+

Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
US

+

+

+

+

Expires September 1995

[Page 53]

Internet Draft SNMPv2 Administrative Infrastructure

March 1995

Phone: +1 412 268 6628
Email: waldbusser@cmu.edu

+

+

Table of Contents

1	Introduction	3
1.1	A Note on Terminology	3
1.2	Change Log	3
2	Overview	5
2.1	Contexts	5
2.2	Authorization: Access Rights and MIB Views	6
2.3	Proxy	7
2.4	Security	8
2.5	Parties	9
2.6	Authorization: Access Control	9
2.7	Construction of an SNMPv2 Message	10
2.8	An SNMPv2 Entity's Local Party Datastore	10
2.9	Maintenance Functions	11
3	Elements of the Model	12
3.1	SNMPv2 Party	12
3.2	SNMPv2 Entity	14
3.3	SNMPv2 Manager	14
3.4	SNMPv2 Agent	14
3.5	SNMPv2 Dual-Role Entity	15
3.6	View Subtree	15
3.7	View Subtree Families	16
3.8	MIB View	16
3.9	Proxy Context	17
3.10	SNMPv2 Context	18
3.11	SNMPv2 PDU	20
3.12	SNMPv2 Message	20

3.13	SNMPv2 Management Communication	21
3.14	SNMPv2 Authenticated Management Communication	22
3.15	SNMPv2 Private Management Communication	22
3.16	SNMPv2 Access Control Policy	23
4	Maintenance Functions	25
4.1	Error Reporting	26
5	Elements of Procedure	28
5.1	Generating a Request	28
5.2	Processing a Received Communication	29
5.3	Generating a Response	36
6	Usage Examples	37
6.1	Non-Secure Minimal Agent Configuration	37
6.2	Secure Minimal Agent Configuration	40
6.3	MIB View Configurations	42
6.4	Proxy Configuration	44
6.5	Public Key Configuration	48

Expires September 1995

[Page 55]

Internet Draft	SNMPv2 Administrative Infrastructure	March 1995
----------------	--------------------------------------	------------

7	Security Considerations	51
8	Acknowledgements	51
9	References	51
10	Authors' Addresses	53

