Network Working Group
Request for Comments: 1909
Category: Experimental

K. McCloghrie, Editor
Cisco Systems, Inc.

February 1996

An Administrative Infrastructure for SNMPv2

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Table of Contents

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, An Administrative Infrastructure for SNMPv2, to define an administrative framework which realizes effective management in a variety of configurations and environments. The SNMPv2 framework is fully described in [1-6]. This framework is derived from the original Internet-standard Network Management Framework (SNMPv1), which consists of these three documents:

STD 16, RFC 1155 [7] which defines the Structure of Management Information (SMI), the mechanisms used for describing and naming objects for the purpose of management.

STD 16, RFC 1212 [8] which defines a more concise description mechanism, which is wholly consistent with the SMI.

STD 15, RFC 1157 [9] which defines the Simple Network Management Protocol (SNMP), the protocol used for network access to managed objects.

For information on coexistence between SNMPv1 and SNMPv2, consult [10].

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.

For example, the managed object type, ifDescr [11], 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, e.g., device-X (the context), ifDescr (the managed object type), and "1" (the instance).

Note that each context has (at least) one globally-unique identification within the management domain. Note also that the same item of management information can exist in multiple contexts. So, an item of management information can have multiple globally-unique identifications, either because it exists in multiple contexts, and/or because each such context has multiple globally-unique identifications.

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 [12, 1],

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 requests 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. 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, 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. (The combination of these two insecure algorithms is sometimes referred to as "noAuth/noPriv".)

2.4. Access Control

An access control policy specifies the types of SNMPv2 requests and associated MIB views which are authorized for a particular identity (on whose behalf a request is generated) when using a particular level of security to access a particular context.

2.5. Security Models

A security model defines the mechanisms used to achieve an administratively-defined level of security for protocol interactions:

(1) by defining the security parameters associated with a communication, including the authentication and privacy algorithms and the security keys (if any) used.

(2) by defining how entities on whose behalf requests are generated are identified.

(3) by defining how contexts are identified.

(4) by defining the mechanisms by which an access control policy is derived whenever management information is to be accessed.

2.6. Proxy

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, this 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, although information on how to forward the request is specifically associated with that context, 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 SNMPv2 agent operating without proxy 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).

3. Elements of the Model

This section provides a more formal description of the model.

3.1. 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 [4]. An SNMPv2 entity assumes the identity of a particular administrative entity 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 administrative entity. Thus, an implementation of an SNMPv2 entity which supports more than one administrative entity need not be multi-threaded. However, there may be situations where implementors may choose to use multi-threading.

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" [5] on its local network-layer addresses.

3.2. 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).

In order to be manageable, all network components need to be instrumented. SNMPv2 access to the instrumented information is via the managed objects supported by an SNMPv2 agent in one or more contexts.

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.

It is interesting to consider the case of managing an SNMPv2 manager. It is highly desirable that an SNMPv2 manager, just like any other networking application, be instrumented for the purposes of being managed. Such instrumentation of an SNMPv2 manager (just like for any other networking application) is accessible via the managed objects supported by an SNMPv2 agent. As such, an SNMPv2 manager is no different from any other network application in that it has instrumentation, but does not itself have managed objects.

That is, an SNMPv2 manager does not itself have managed objects. Rather, it is an associated SNMPv2 agent supporting managed objects which provides access to the SNMPv2 manager's instrumentation.

3.4. 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 initiates requests by acting in a manager role, and processes requests regarding management information accessible to it (locally or via proxy) through acting in an agent role. In the case of sending inform notifications, an SNMPv2 dual-role entity acts in a manager role in initiating an inform notification containing management information which is accessible to it when acting in an agent role.

An SNMPv2 entity which can act only in an SNMPv2 manager role is not SNMP-manageable, since there is no way to access its management instrumentation. In order to be SNMP-manageable, an SNMPv2 entity must be able to act in an SNMPv2 agent role in order to allow its instrumentation to be accessed. Thus, it is highly desirable that all SNMPv2 entities be either SNMPv2 agents or SNMPv2 dual-role entities.

There are two categories of SNMPv2 dual-role entities: proxy SNMPv2 agents and (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. (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 2.6.)

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.

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 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.

3.6. MIB View

A MIB view is a subset of the set of all instances of all object types defined according to the SMI [1] 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 local or 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.

The term remote SNMPv2 context is used at an SNMPv2 manager to indicate a SNMPv2 context (either local or proxy) which is not realized by the local SNMPv2 entity (i.e., the local SNMPv2 entity uses neither locally-defined mechanisms, nor acts as a proxy SNMPv2 agent, to access the management information identified by the SNMPv2 context).

3.7.1. Local SNMPv2 Context

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

Because a device may contain several such local entities, 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, 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/responses 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. Note however, that agents which support the forwarding of Set operations concerning information at multiple locations are not considered to be proxy SNMPv2 agents (see section 2.6 above).

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.

3.8. SNMPv2 PDUs and Operations

An SNMPv2 PDU is defined in [4]. Each SNMPv2 PDU specifies a particular operation, one of:

```
GetBulkRequest
GetNextRequest
GetRequest
Inform
Report
Response
SNMPv2-Trap
SetRequest
```

3.8.1. The Report-PDU

[4] requires that an administrative framework which makes use of the Report-PDU must define its usage and semantics. With this administrative framework, the Report-PDU differs from the other PDU types described in [4] in that it is not a protocol operation between SNMPv2 managers and agents, but rather is an aspect of error-reporting between SNMPv2 entities. Specifically, it is an interaction between two protocol engines.

A communication between SNMPv2 entities is in the form of an SNMPv2 message. Such a message is formatted as a "wrapper" encapsulating a PDU according to the "Elements of Procedure" for the security model used for transmission of the message.

While processing a received communication, an SNMPv2 entity may determine that the received message is unacceptable due to a problem associated with the contents of the message "wrapper". In this case, an appropriate counter is incremented and the received message is discarded without further processing (and without transmission of a Response-PDU).

However, if an SNMPv2 entity acting in the agent role makes such a determination, then after incrementing the appropriate counter, it may be required to generate a Report-PDU and to send it to the

transport address which originated the received message.

If the agent is able to determine the value of the request-id field of the received PDU [4], then it must use 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. The variable-bindings field contains a single variable: the identity of the counter which was incremented and its new value.

There is at least one case in which a Report-PDU must not be sent by an SNMPv2 entity acting in the agent role: if the received message was tagged as a Report-PDU. Particular security models may identify other such cases.

3.9. SNMPv2 Access Control Policy

An SNMPv2 access policy specifies the types of SNMPv2 operations authorized for a particular identity using a particular level of security, and if the operation is to be performed on a local SNMPv2 context, two accessible MIB views. The two MIB views are a read-view and a write-view. A read-view is a set of object instances authorized for the identity when reading objects. Reading objects occurs when processing a retrieval (get, get-next, get-bulk) operation and when sending a notification. A write-view is the set of object instances authorized for the identity when writing objects. Writing objects occurs when processing a set operation. An identity's access rights may be different at different agents.

A security model defines how an SNMPv2 access policy is derived; however, the application of an SNMPv2 access control policy is performed only:

- on receipt of GetRequest, GetNextRequest, GetBulkRequest, and SetRequest operations; and
- prior to transmission of SNMPv2-Trap and Inform operations.

Note that application of an SNMPv2 access control policy is never performed for Response or Report operations.

4. Security Considerations

Security issues are not directly discussed in this memo.

5. Editor's Address

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

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

6. Acknowledgements

This document is the result of significant work by three major contributors:

Keith McCloghrie (Cisco Systems, kzm@cisco.com)
Marshall T. Rose (Dover Beach Consulting, mrose@dbc.mtview.ca.us)
Glenn W. Waters (Bell-Northern Research Ltd., gwaters@bnr.ca)

The authors wish to acknowledge James M. Galvin of Trusted Information Systems who contributed significantly to earlier work on which this memo is based, and the general contributions of members of the SNMPv2 Working Group, and, in particular, Aleksey Y. Romanov and Steven L. Waldbusser.

A special thanks is extended for the contributions of:

Uri Blumenthal (IBM)
Shawn Routhier (Epilogue)
Barry Sheehan (IBM)
Bert Wijnen (IBM)

7. References

[1] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[2] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

[3] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S., Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.

[4] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[6] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907, January 1996.

[7] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", STD 16, RFC 1155, May 1990.

[8] Rose, M., and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.

[9] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[10] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", RFC 1908, January 1996.

[11] McCloghrie, K., and F. Kastenholz, "Evolution of the Interfaces Group of MIB-II", RFC 1573, Cisco Systems, FTP Software, January 1994.

[12] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824, (December, 1987).

APPENDIX A - Disambiguating the SNMPv2 Protocol Definition

The descriptions in [4] of the role in which an SNMPv2 entity acts when sending an Inform-Request PDU are ambiguous. The following updates serve to remove those ambiguities.

(1) Add the following sentence to section 2.1:

> Further, when an SNMPv2 entity sends an inform notification, it acts in a manager role in respect to initiating the operation, but the management information contained in the inform notification is associated with that entity acting in an agent role. By convention, the inform is sent from the same transport address as the associated agent role is listening on.

(2) Modify the last sentence of the second paragraph in section 2.3:

> This type is used by one SNMPv2 entity, acting in a manager role, to notify another SNMPv2 entity, also acting in a manager role, of management information associated with the sending SNMPv2 entity acting in an agent role.

(3) Modify the second paragraph of section 4.2 (concerning the generation of Inform-Request PDUs):

> It is mandatory that all SNMPv2 entities acting in a manager role be able to generate the following PDU types: GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, and Response-PDU; further, all such implementations must be able to receive the following PDU types: Response-PDU, SNMPv2-Trap-PDU, InformRequest-PDU. It is mandatory that all dual-role SNMPv2 entities must be able to generate an Inform-Request PDU.

(4) Modify the first paragraph of section 4.2.7:

> An InformRequest-PDU is generated and transmitted at the request of an application in a SNMPv2 entity acting in a manager role, that wishes to notify another application (via an SNMPv2 entity also acting in a manager role) of information in a MIB view which is accessible to the sending SNMPv2 entity when acting in an agent role.

APPENDIX B - Who Sends Inform-Requests?

B.1. Management Philosophy

Ever since its beginnings as SGMP, through its definition as SNMPv1, and continuing with the definition of SNMPv2, SNMP has embodied more than just a management protocol and the definitions of MIB objects. Specifically, SNMP has also had a fundamental philosophy of management, consisting of a number of design strategies. These strategies have always included the following:

(1) The impact of incorporating an SNMP agent into a system should be minimal, so that both: a) it is feasible to do so even in the smallest/cheapest of systems, and b) the operational role and performance of a system is not compromised by the inclusion of an SNMP agent. This promotes widespread development, which allows ubiquitous deployment of manageable systems.

(2) Every system (potentially) incorporates an SNMP agent. In contrast, the number of SNMP managers is limited. Thus, there is a significantly larger number of SNMP agents than SNMP managers. Therefore, overall system development/complexity/cost is optimized if the SNMP agent is allowed to be simple and any required complexity is performed by an SNMP manager.

(3) The number of unsolicited messages generated by SNMP agents is minimized. This enables the amount of network management traffic to be controlled by the small number of SNMP managers which are (more) directly controlled by network operators. In fact, this control is considered of greater importance than any additional protocol overhead which might be incurred. Monitoring of network state at any significant level of detail is accomplished primarily by SNMP managers polling for the appropriate information, with the use of unsolicited messages confined to those situations where it is necessary to properly guide an SNMP manager regarding the timing and focus of its polling. This strategy is sometimes referred to as "trap-directed polling".

B.2. The Danger of Trap Storms

The need for such control over the amount of network management traffic is due to the potential that the SNMP manager receiving an unsolicited message does not want, no longer wants, or already knows of the information contained in the message. This potential is significantly reduced by having the majority of messages be specific requests for information by SNMP managers and responses (to those requests) from SNMP agents.

The danger of not having the amount of network management be controlled in this manner is the potential for a "storm" of useless traps. As a simple example of "useless", consider that after a building power outage, every device in the network sends a coldStart trap, even though every SNMP manager and every network operator already knows what happened. For a simple example of "storm", consider the result if each transmitted trap caused the sending of another. The greater the number of problems in the state of the network, the greater the risk of such a storm occurring, especially in the unstructured, heterogeneous environment typical of today's internets.

While SNMP philosophy considers the above to be more important than any lack of reliability in unsolicited messages, some users/developers have been wary of using traps because of the use (typically) of an unreliable transport protocol and because traps are not acknowledged. However, following this logic would imply that having acknowledged-traps would make them reliable; of course, this is false since no amount of re- transmission will succeed if the receiver and/or the connectivity to the receiver is down. A SNMP manager cannot just sit and wait and assume the network is fine just because it is not receiving any unsolicited messages.

B.3. Inform-Requests

One of the new features of SNMPv2 is the Inform-request PDU. The Inform-Request contains management information specified in terms of MIB objects for a context supported by the sender. Since by definition, an SNMPv2 manager does not itself have managed objects (see sections 3.3), the managed objects contained in the Inform-request belong to a context of an SNMPv2 agent, just like the managed objects contained in an SNMPv2-Trap.

However, it is not the purpose of an Inform-request to change the above described philosophy, i.e., it would be wrong to consider it as an "acknowledged trap". To do so, would make the likelihood and effect of trap storms worse. Recall the building power outage example: with regular traps, the SNMP manager's buffer just overflows when it receives messages faster than it can cope with; in contrast, if every device in the network were to send a coldStart Inform-request, then after a power outage, all will re-transmit their Inform-request several times unless the receiving SNMP managers send responses. In the best case when no messages are lost or re-transmitted, there are twice as many useless messages; in the worst case, the SNMP manager is unable to respond at all and every message is re-transmitted its maximum number of times.

The above serves to explain the rationale behind the definition (see Appendix A's update to section 4.2.7 of [4]) that:

> An InformRequest-PDU is generated and transmitted at the request of an application in a SNMPv2 entity acting in a manager role, that wishes to notify another application (via an SNMPv2 entity also acting in a manager role) of information in a MIB view which is accessible to the sending SNMPv2 entity when acting in an agent role.

This definition says that SNMPv2 agents do not send Inform-Requests, which has three implications (ordered in terms of importance):

(1) the number of devices which send Inform-requests is required to be a small subset of all devices in the network;

(2) while some devices traditionally considered to be SNMP agents are perfectly capable of sending Inform-requests, the overall system development/complexity/cost is not increased as it would be by having to configure/re-configure every SNMPv2 agent as to which Inform-requests to send where and how often; and

(3) the cost of implementing an SNMPv2 agent in the smallest/cheapest system is not increased.