ACE-SNMP

Web Based SNMP Network Management System

An Introductory Overview of SNMP

---

ddri

diversified data resources, inc.

114 Professional Center Drive, Novato, CA 94947

AN OVERVIEW OF SNMP

Appendices

AN OVERVIEW OF SNMP

This document provides an overview of SNMP (Simple Network Management Protocol) as an introduction to the basic capabilities provided by the DDRI ACE-SNMP Network Management system. This information will be of interest to new users of SNMP, as well as existing SNMP users interested in the alternate network management perspectives offered by DDRI.

1. Introduction

SNMP is a communication protocol that has gained widespread acceptance since 1993 as a method of managing TCP/IP networks, including individual network devices, and devices in aggregate. SNMP was developed by the IETF (Internet Engineering Task Force), and is applicable to any TCP/IP network, as well as other types of networks.

A variety of reference materials exist on SNMP. The protocol has been in existence for some time, and has been written about extensively. One of the first books to describe SNMP in detail was “The Simple Book” by Marshall T. Rose, published in the early 1990's, which set the tone for SNMP usage, and which is still a highly applicable reference today.

1.1 SNMP Description

SNMP defines a client/server relationship. The client program (called the network manager) makes virtual connections to a server program (called the SNMP agent) which executes on a remote network device, and serves information to the manager regarding the device’s status. The database, controlled by the SNMP agent, is referred to as the SNMP Management Information Base (MIB), and is a standard set of statistical and control values. SNMP additionally allows the extension of these standard values with values specific to a particular agent through the use of private MIBs.

Directives, issued by the network manager client to an SNMP agent, consist of the identifiers of SNMP variables (referred to as MIB object identifiers or MIB variables) along with instructions to either get the value for the identifier, or set the identifier to a new value.

Through the use of private MIB variables, SNMP agents can be tailored for a myriad of specific devices, such as network bridges, gateways, and routers. The definitions of MIB variables supported by a particular agent are incorporated in descriptor files, written in Abstract Syntax Notation (ASN.1) format, made available to network management client programs so that they can become aware of MIB variables and their usage.

1.2 Strengths and Shortcomings of SNMP

SNMP has several strengths. Its biggest strength is arguably its widespread popularity. SNMP agents are available for network devices ranging from computers, to bridges, to modems, to printers. The fact that SNMP exists with such support gives considerable credence to its reason for existence; SNMP has become interoperable.

Additionally, SNMP is a flexible and extensible management protocol. Because SNMP agents can be extended to cover device specific data, and because a clear mechanism exists for upgrading network management client programs to interface with special agent capabilities (through the use of ASN.1 files), SNMP can take on numerous jobs specific to device classes such as printers, routers, and bridges, thereby providing a standard mechanism of network control and monitoring.

Several weaknesses of SNMP can be identified. Despite its name, (i.e. “Simple” Network Management Protocol), SNMP is a highly complicated protocol to implement. By the admission of its designers, a more apt name might be “Moderate Network Management Protocol”, although even this seems generous in light of SNMP's highly complicated encoding rules. Also, SNMP is not a particularly efficient protocol. Bandwidth is wasted with needless information, such as the SNMP version (transmitted in every SNMP message) and multiple length and data descriptors scattered throughout each message. The way that SNMP variables are identified (as byte strings, where each byte corresponds to a particular node in the MIB database) leads to needlessly large data handles that consume substantial parts of each SNMP message.

The above complaints, while fair, cannot reasonably be used to detract from the aforementioned strengths of SNMP. While complicated encoding rules frustrate programmers who must deal with them, it is easily argued that end users of the protocol aren't concerned with the complexity of the algorithms that handle their data. As for complaints about SNMP efficiency, it has been shown to work well enough, given the nature of network administration activities. In short, SNMP has been shown to work as a management protocol, making criticisms (however well deserved) irrelevant.

1.3 The Greatest Criticism of SNMP

One criticism of SNMP cannot be dismissed so easily: SNMP has generally lacked focus on why it is useful. Neither the governing RFC's nor other explanatory references describe why SNMP is better than more traditional network management tools such as “ping”, “rsh”, and “netstat”. Consider: the design of SNMP agents difficult. Equally difficult is the administration of these many agents and their managers, which typically entails individual configuration and possible maintenance. Why is SNMP all that much better than, say, telnet protocol? Does the elaborateness of SNMP justify its usefulness? Or can simple telnet driver programs, which periodically log into network devices to gather statistics and apply control, achieve the same end?

In part, this criticism has been aggravated by SNMP management program vendors, who create systems where SNMP is merely an alternative to remote shells rather than a new way of analyzing and managing networks. Many SNMP vendors seem to see network management activities in terms of GUI's with point and click controls instead of automated systems for configuring devices, gathering data, and conduct corrective procedures on large networks.

1.4 The Argument For SNMP

Faced with the above criticism of SNMP, one tendency is to admit that there are alternatives to SNMP, but these alternatives are supplanted by the general popularity and interoperability of SNMP. This tendency should be rejected in favor of a much better argument for SNMP; the fact is that there ARE NO current alternatives to SNMP, at least for certain types of network administrative functions. While the protocol itself may be less than perfect, SNMP provides the ONLY way to efficaciously manage large networks.

The lack of understanding about SNMP's usefulness is partly attributable to a basic flaw in the philosophy of many SNMP management programs as follows: Most vendors implement network managers with the idea that a user's primary interested is in the data associated with particular network devices. But the truth is, this data is easily acquired by other means (such as with “netstat” and “rsh” Unix programs). The truly pertinent information about the network is the DIFFERENCES between devices irrespective of their current state. SNMP affords the only mechanism currently available to process these differences rapidly on networks of scale, since SNMP avoids the processing burden of remote login and execution.

2. The SNMP Standard

SNMP can be viewed many different ways. One perspective is to regard SNMP as three distinct standards.

1. **A Standard Message Format.** SNMP is a standard communication protocol that defines a UDP message format. This part of the standard is highly involved, and is of little consequence to users (but of great interest to SNMP programmers.)

2. **A Standard Set Of Managed Objects.** SNMP is a standard set of values (referred to as SNMP "objects") that can be queried from a device. Specifically, the standard includes values for monitoring TCP, IP, UDP, and device interfaces. Each manageable object is identified with an official name, and also with a numeric identifier expressed in dot-notation.

3. **A Standard Way Of Adding Objects.** Certainly, one reason that SNMP has become popular and the industry standard is that a method was originally defined so that the standard set of managed objects (above) could be augmented by network device vendors with new objects specific for a particular network

Each of these different standards are described in more detail within the paragraphs below.

2.1 SNMP Message Standard

Four types of SNMP messages are defined: a "get" request exists that returns the value of a named object; a "get-next" request exists that returns the next name (and value) of the "next" object supported by a network device given a valid SNMP name; a "set" request is defined which sets a named object to a specific value; a "trap" message is defined, generated asynchronously by network devices, which can notify a network manager of a problem apart from any polling of the device. Each of these message types fulfills a particular requirement of network managers:

1. **Get Request.** Specific values can be fetched via the "get" request to determine the performance and state of the device. Typically, many different values and parameters can be determined via SNMP without the overhead associated with logging into the device, or establishing a TCP connection with the device.

2. **Get Next Request.** The SNMP standard network managers to "walk" through all SNMP values of a device (via the "get-next" request) to determine all names and values that an operant device supports. This is accomplished by beginning with the first SNMP object to be fetched, fetching the next name with a "get-next", and repeating this operation until an error is encountered (indicating that all MIB object names have been "walked".)

3. **Set Request.** The SNMP standard provides a method of effecting an action associated with a device (via the "set" request) to accomplish activities such as disabling interfaces, disconnecting users, clearing registers, etc. This provides a way of configuring and controlling network devices via SNMP.

4. **Trap Message**. The SNMP standard furnishes a mechanism by which devices can "reach out" to a network manager on their own (via the "trap" message) to notify the manager of a problem with the device. This typically requires each device on the network to be configured to issue SNMP traps to one or more network devices that are awaiting these traps.

The above message types are all encoded into messages referred to as "Protocol Data Units" (PDUs) which are interchanged with SNMP devices. The actual format of this message is not-at-all simple or convenient; fortunately this complexity is generally hidden by the network management software.

2.2 Standard Managed Objects

The list of values that an object supports is often referred to as the SNMP "Management Information Base" (MIB). This term is often used in a cavalier way to describe any SNMP object or portion of an SNMP hierarchy. Strictly speaking, "MIB" is simply an abstraction like "Database" which can be applied to mean all data, or any portion thereof, associated with the network. (This casual nomenclature often confuses new users.)

The various SNMP values in the standard MIB are defined in RFC-1213, (one of the governing specifications for SNMP). The standard MIB includes various objects to measure and monitor IP activity, TCP activity, UDP activity, IP routes, TCP connections, interfaces, and general system description. These each of these values is associated both an official name (such as "sysUpTime", which is the elapsed time since the managed device was booted) and a numeric value expressed in dot-notation (such as "1.3.6.1.2.1.1.3.0", which is the "object identifier" for "sysUpTime").

Naturally, the tendency is to avoid the SNMP “dot-notation” name of a MIB object, in favor of the official object name, much the same way that IP addresses are usually supplanted by official host names.

Section 3 of this document is dedicated to a comprehensive description of the characteristics and properties of MIB objects.

2.3 SNMP Extensibility Feature

It can easily be argued that SNMP has become prominent mainly from its ability to augment the standard set of MIB objects with new values specific for certain applications and devices. Hence, new functionality can continuously be added to SNMP, since a standard method has been defined to incorporate that functionality into SNMP devices and network managers. This is accomplished through a process usually referred to as “compiling” a new MIB, which allows the user to add new MIB definitions to the system. These definitions are usually supplied by network equipment vendors in specially formatted text files using the ASN.1 standard syntax. (ASN.1 refers to “Abstract Syntax Notation One”, which is an obscure type declaration language, adopted by SNMP, and used a few other places, including encryption and CMIP protocols.)

Note that the MIB of an SNMP device is usually fixed; it is constructed by the network equipment vendor (i.e. router manufacturer, computer hardware vendor, etc.) and cannot be added to or modified. This extensibility of SNMP refers strictly to SNMP management software, which can become aware of the MIB values supported by the device by compiling a description of the device into the network management program.

3. SNMP MIB Objects

To use SNMP effectively, users need to become acquainted with the SNMP Management Information Base, which defines all the values that SNMP is capable of reading or setting. To become an expert in SNMP, an administrator must become familiar with the SNMP MIB.

The SNMP MIB is arranged in a tree-structured fashion, similar in many ways to a disk directory structure of files. The top level SNMP branch begins with the ISO “internet” directory, which contains four main branches: The “mgmt” SNMP branch contains the standard SNMP objects usually supported (at least in part) by all network devices; the “private” SNMP branch contains those “extended” SNMP objects defined by network equipment vendors; the “experimental” and “directory” SNMP branches, also defined within the “internet” root directory, are usually devoid of any meaningful data or objects.

The “tree” structure described above is an integral part of the SNMP standard, however the most pertinent parts of the tree are the “leaf” objects of the tree that provide actual management data egarding the device. Generally, SNMP leaf objects can be partitioned into two similar but slightly different types that reflect the organization of the tree structure:

1. **Discrete MIB Objects.** Discrete SNMP objects contain one precise piece of management data. These objects are often distinguished from “Table” items (below) by adding a “.0” (dot-zero) extension to their names. (If the “.0” extension is omitted from a leaf SNMP object name, it is always implied.)

2. **Table MIB Objects.** Table SNMP objects contain multiple pieces of management data. These objects are distinguished from “Discrete” items (above) by requiring a “.” (dot) extension to their names that uniquely distinguishes the particular value being referenced.

The “.” (dot) extension is referred to in some literature as the “instance” number of an SNMP object. In the case of “Discrete” objects, this instance number will be zero. In the case of “Table” objects, this instance number will be the index into the SNMP table. More detail is provided on these two categories of SNMP objects below.

3.1 Discrete MIB Objects

Many SNMP objects are discrete. That is, the operator merely has to know the name of the object and no other information. Discrete objects often represent summary values for a device, particularly useful for scanning information from the network for the purposes of comparing network device performance. If the extension (instance number) of the object is not specified, it can be assumed as “.0” (dot-zero).

3.2 Table MIB Objects

SNMP tables are special types of SNMP objects, which allow parallel arrays or information to be supported. Tables are distinguished from scalar objects, in that tables can grow without bounds. For example, SNMP defines the “ifDescr” object (as a standard SNMP object) which indicates the text description of each interface supported by a particular device. Since network devices can be configured with more than one interface, this object could only be represented as an array.

By convention, SNMP objects are always grouped in an “Entry” directory, within an object with a “Table” suffix. (The “ifDescr” object described above resides in the “ifEntry” directory contained in the “ifTable” directory.) Several constraints are placed on SNMP objects as follows:

1. Each object in the “Entry” directory of a table must contain the same number of elements as other objects in the same “Entry” directory, where a instance numbers of all entries are the same. Table objects are always regarded as parallel arrays of data.

2. When creating a new “Entry” object, SNMP requires that a value be associated with each table entry in a single SNMP message (single PDU). This means that, to create a row in a table (via an SNMP “set” command), a value must be specified for each element in the row.

3. If a table row can be deleted, SNMP requires that at least one object in the entry has a control element that is documented to perform the table deletion. (This applies only if a row can be deleted, which is not necessarily required of an SNMP table.)

3.3 MIB Object Types

MIB objects have specific “type” values. SNMP defines several primitive types as follows:

1. **Text Type MIB Objects.** SNMP defines a “DisplayString” type that can contain arbitrary textual information (usually to a maximum of 256 characters). The text must contain only printable characters. Examples of this type of object include “sysDescr”, and “ifDescr” values. This type of object is quite common in the SNMP MIB.

2. **Counter Type MIB Objects**. SNMP defines a “Counter” type, which is a numeric value that can only increase. This is the most common type of SNMP object in the standard MIB, and includes objects such as “ipInReceives”, “ipOutRequests”, and “snmpInPkts”. Counters roll over at their maximum value, and can never be less than zero.

3. **Gauge Type MIB Objects**. SNMP defines a “Gauge” type, which is a numeric value that can increase or decrease. This type is found in only a few locations within the standard MIB (although is often expressed in the “private” MIBs of vendor equipment). Examples of this type of object include the “tcpCurrEstab” object.

4. **Integer Type MIB Objects.** SNMP defines a basic “Integer” type that can contain positive or negative values. This value is usually supplanted by “Counter” or “Gauge” type values, but is sometimes expressed in “private” MIBs of vendor equipment.

5. **EnumVal Type MIB Objects.** SNMP defines an “Enumerated Value” type that associates a textual label with a numeric value. This type is quite common in the standard MIB, and includes objects such as “ifAdminStatus”, whose enumerated values are “up(1)”, “down(2)”, and “testing(3)”. In general, enumerated values are usually formatted for display using the convention “name(number)”.

6. **Time Type MIB Objects**. SNMP defines a “TimeTicks” type that represents an elapsed time. This time always has a resolution of one hundredth of a second, even if this resolution is not used. Network managers frequently format this time value as “HH:MM:SS:hh” for display. Note that the “Time” value is always an elapsed time. For example, the “sysUpTime” value indicates the elapsed time since a device was rebooted.

7. **Object Type MIB Objects.** SNMP defines an “Object” type that can contain the identifier for another SNMP object. If the named object is compiled into the MIB, the name is usually displayed as the name of the SNMP object. An example of this type of object is the “sysObjectID” MIB variable.

8. **IPAddr Type MIB Objects.** SNMP defines an “IP address” type that contains the IP address of a network device. This type of object is often displayed in type as an IP address in conventional dot-notation. Examples of this type of object include “ipRouteDest”, “ipRouteNextHop”, and “ipRouteMask”.

9. **PhysAd Type MIB Objects.** SNMP defines a “Physical address” type that contains the MAC layer address of a network device. Managers often display this type of object as a sequence of hexadecimal values, prefixed by the “hex:” keyword, and separated by colons. Examples of this type of object include the “ifPhysAddress” object

10. **String Type MIB Objects.** SNMP defines a “String” type that contains arbitrary byte strings. If the byte string contains only ASCII characters, managers display this value as a text string. Otherwise, managers display this type as a sequence of hexadecimal values, prefixed by the “hex:” keyword, and separated by colons. This type of value is uncommon, but occasionally found in the “private” MIBs of vendor equipment.

11. **Table Type MIB** Objects. The SNMP “Table” type is a branch object that contains table entries. This object type is always an intermediate name that contains an “Entry” directory, which in turn contains various table objects.

12. **Branch Type MIB Objects.** The SNMP “Branch” type is a branch object that contains additional branches, tables, or any of the discrete objects types listed above.

3.4 MIB Object Access Values

Each SNMP object is defined to have a particular access, either “read-only”, “read-write”, or “write-only” that determines whether the user can read the object value, read and write the object (with a “set” command) or only write the object.

Before any object can be read or written, the SNMP community name must be known. These community names are configured into the system by the administrator, and can be viewed as passwords needed to gather SNMP data. Strictly speaking, community names exist to allow portions of the SNMP MIB, and object subsets, to be referenced. As the term “Community” implies, the true purpose of these values is intended to identify commonality between SNMP object sets. However, it is common practice to make these community names obscure so as to limit access to SNMP capability by outside users.

3.5 Compiling MIB Objects

One of the principal components of any respectable SNMP manager is a “MIB Compiler” which allows new MIB objects to be added to the management system. This concept can be confusing to new users, possibly because of the strange nomenclature associated with this term.

When a MIB is compiled into an SNMP manager, the manager is made “aware” of new objects that are supported by agents on the network. The concept is similar to adding a new schema to a database. The agent is not affected by the MIB compilation (since it is already “aware” of its own objects). The act of compiling the MIB allows the manager to know about the special objects supported by the agent and access these objects as part of the standard object set.

Typically, when a MIB is compiled into the system, the manager creates new folders or directories that correspond to the objects. These folders or directories can typically be viewed with a “MIB Browser”, which is a traditional SNMP management tool incorporated into virtually all network management systems. These new objects can often be alarmed or possibly modified to affect the performance of the remote agent.

MIB objects are documented in ASN.1 syntax, which is an obscure type declaration language, adopted by SNMP, and used a few other places. The user will obtain ASN.1 definitions for a new piece of network equipment or new SNMP agent, transfer this file to the network management system, and run the management system “MIB Compiler” to incorporate these definitions into the system. Virtually all agents support the RFC1213 MIB definitions, and most agents support other definitions as well.

4. The SNMP Management Paradigm

Previous sections of this document describe the basic characteristics of SNMP. This begs the question as to why SNMP is preferable to other types of management protocols. How is SNMP useful in the management of networks?

The simple answer to this question is that SNMP is designed as a uniform way of accessing a device. SNMP is an Application Program Interface (API) to the network, so that general-purpose network management programs can be easily written to work with a variety of different devices. Without SNMP, there would certainly be a myriad of special, custom-written applications, operable only with a specific vendor's equipment. Additionally, SNMP is a low-overhead way of determining device status without the need for elaborate remote login or authentication, thereby allowing large numbers of status values to be obtained quickly for a large-scale network.

4.1 Essential Management Components

All substantial SNMP managers provide several basic components as follows:

1. **Alarm Polling Functions**. All substantial SNMP managers provide some ability to set thresholds on SNMP MIB objects, and respond with some type of notification when these thresholds are violated. This provides a means of constantly testing a network's integrity against a baseline. Inherent with this function is the ability to determine which devices are responding (i.e. are online) and which devices are not responding (i.e. are faulted).

2. **Trend Monitoring Functions.** All SNMP managers provide some ability to continuously watch an SNMP value over time, sampling the value at periodic intervals so as to view the trend of a network. This type of function can be used to determine the load of a network over time (by watching bandwidth). A typical output of a management system is a plot of network utilization versus time.

3. **Trap Reception Functions.** All SNMP managers provide some ability to receive (and filter) SNMP traps issued by network devices. SNMP traps are an important part of the SNMP standard, and allow devices to self-report problems. These traps are typically logged by the network manager, and drive event notifications. Because traps are asynchronous and outside of the control of the network manager, all SNMP managers requires some type of trap filtering to eliminate traps from being report that are not pertinent or annoying.

4. **Management Tool Set.** All SNMP managers provide a tools set. The most traditional type of SNMP management tool is a MIB browser, which allows a user to inspect the raw MIB objects of a particular device. This is often the principal interface for setting SNMP values of an agent and actually applying changes to the network via SNMP.

5. **MIB Compiler.** For an SNMP manager to be useful, it must support the ability to add in new MIB objects provided by specialized network equipment. This implies the absolute need for a MIB compiler so that the special extensions provided by network equipment vendors can be browsed and alarmed.

Note that the above features of an SNMP manager are heavily loaded for the “Monitor” aspect of network management. The vast majority of SNMP objects are “read-only”. SNMP does well in its ability to obtain network status information and determine network health. It is less powerful when it comes to making modifications to the network, although the SNMP “Set” command (often achieved fully by a MIB Browser) allows an administrator to take some forms of corrective action via SNMP.

4.2 ACE-SNMP Network Manager

By way of example, consider DDRI's ACE-SNMP network management system. This is an SNMP management system providing various functions illustrative of network managers in general. It incorporates a full set of management screens, tools, and functions suitable for use with small, medium, or large networks.

One or more copies of ACE-SNMP execute on network platforms. Each program copy continuously polls network devices and tests SNMP values against user defined thresholds. When a threshold is violated, or a network device is found to be inaccessible, ACE-SNMP reports this condition via a syslog message, e-mail message, or both. Up to 4000 different alarms can be configured for each executing copy of ACE-SNMP. Each alarm tests a single MIB object, or multiple objects.

Additionally, ACE-SNMP watches for SNMP traps, and allows the operator to associate textual messages and alert severities for enterprise traps. ACE-SNMP provides a MIB compiler that creates trap definitions automatically from ASN.1 files, and allows alert severities and alert messages to be associated with SNMP traps.

The particular tools and functions available to ACE-SNMP users is site dependent. ACE-SNMP implements a plug-in facility, which allows the user to add new functions and screens to the ACE-SNMP system. These plug-in components include a check URL function (ChkUrl), an alpha-numeric pager notification function (SendPg), and other special functions that allow the user to scale up or down depending upon site requirements.

The major application divisions of the ACE-SNMP system are as follows:

1. **Events Application.** The ACE-SNMP "Events" application provides the main interface to the ACE-SNMP event manager function, and allows logged events to be inspected, including historical data regarding these events.

2. **Maps Application.** The ACE-SNMP "Maps" application allows the user to construct displays that indicate the aggregate status of multiple devices, representing physical or logical partitioning of the network.

3. **Trends Application.** The ACE-SNMP "Trends" application allows the user to specify SNMP devices that are trend monitored. The trend monitor continuously checks the interface bandwidth, IP activity, and TCP activity of these devices and depicts network usage in graphical format.

4. **Alarms Application.** The ACE-SNMP "Alarms" application provides the main interface to ACE-SNMP alarms, allowing the user to inspect the current state of alarms and devices on the network, and to make changes to alarm definitions.

5. **Tools Application.** The ACE-SNMP "Tools" application provides different tools to diagnose and trouble shoot the system, including access to a MIB Browser, MIB Table Viewer, Network Scanner, MIB Compiler, and Auto-Discovery tool.

6. **Find Application.** The ACE-SNMP "Find" application provides a search engine that can be used to search for device information, event data, help files, and other managed data items.

Each of these applications consist of a substantial number of screens discussed in the various system help files. Each application may be regarded as stand-alone; the operator can switch context from one application to another at any time, from any screen.

4.2.1 Events Application

The ACE-SNMP “Events” application is typical of many SNMP managers. This application records and diagnoses network events as they occur. The application consists of an event log, a correlation function, and a notification function. When events occur, the user can be contacted via e-mail or pager. Events can also trigger reactions (via the execution of scripts for example) that can take corrective action or perform other types of notifications.

Network events are classified into high level messages that indicate a network problem. The “Events” application allows the operator to inspect both recent events, and long term historical event trends.

ACE-SNMP provides three main types of network events:

1. **Network Events.** ACE-SNMP indicates when network devices are online, offline, or have been rebooted. These are built in events, which are correlated to form discrete or “Network” events. For example, if many devices go offline, and then return online after a short period, ACE-SNMP reports an “intermittent network” event rather than a series of offline and online events.

2. **Alarm Events.** ACE-SNMP reports when an alarmed MIB object has violated a user define threshold. For example, the user may set an alarm on the interface traffic of a router. If this traffic exceeds a certain threshold (or is less than an expected value) an event occurs. ACE-SNMP allows any MIB object to be alarmed, including text strings, numeric values, or time values. Each object can be compared to an absolute value or a “delta” value, where a delta value is the previous value of a sample subtracted from the current value, and scaled by time.

3. **SNMP Trap Events.** ACE-SNMP reports when SNMP traps occur. The program collects traps, and allows the user to define a severity and text message to be used for each trap. ACE-SNMP additionally allows traps to be filtered by a variety of techniques.

Associated with each event is an alert severity value, which is incorporated into alert messages, and which can be used to filter alerts. This alert severity level works with the event notification facility, which allows messages above a specific severity to be relayed to a syslog host or transmitted by e-mail. Specific elements of this notification system are configurable via screens accessed via the ACE-SNMP “Setup” screens.

4.2.2 Maps Application

The ACE-SNMP “Maps” application is typical of many SNMP managers, and allows the user to setup “Process” models, where each map object icon contains the status of multiple managed devices. When an aspect of the process fails, or is out of tolerance, the top-level icon reflects this. The user can see an immediate indication of what the failure is by clicking on the object icon, which decomposes into the series of devices, each of which decompose further into a series of alarms.

The ACE-SNMP “Maps” application can create topology maps and detailed subnet maps. However, the main power of this facility is to define “process maps” and logical diagrams that summarize network-status information. For example, the user can define a hierarchical display, where the top-level process includes all managed devices, and lower level processes (connected via lines on the screen) represent subsets of the main process. Therefore, when an error condition occurs, the user sees which particular branches of the hierarchy are affected. These top level branches can be named things like “Building 4”, and “Accounting Department” and “Workstations” so that the user sees a logical representation of network health (as opposed to a physical topology map.)

4.2.3 Trends Application

The ACE-SNMP “Network Trend Monitor” is typical of many SNMP managers, and allows the user to inspect historical data and trends regarding network device usage for the current day, and for a period of 30 days. This data is depicted in graphical or tabular form, and provides application in baselining network usage, or detecting changes pertaining to network usage.

The ACE-SNMP Trend Monitor agent runs as a continuous background process and gathers data from each managed device at 15-minute intervals. This data is recorded and saved for 30 days. The “Network Trend Monitor” allows data to be inspected for the current day, or for past days. In particular, statistical evaluation of this data is performed by the “Network Trend Monitor” screens to allow easy assessment of minimum, maximum, average, and standard deviation for each monitored parameter.

4.2.4 Alarms Application

The ACE-SNMP “Alarms” application is typical of many SNMP managers, and allows the user to add thresholds on MIB objects. This process is often referred to as “instrumenting” a device, because the process is similar to putting test instruments on a hardware component. The “Alarms” application allows the user to inspect alarm, add new alarms, or modify existing alarms.

The ACE-SNMP Alarm facility consists of various screens, and a background process that continuously gathers SNMP information on the system and compares this data against user defined thresholds. When an alarm or network fault is detected, this information is logged in the “Events” application (discussed above), and is relayed to the user via an e-mail message or other notification.

Each alarm tests a single MIB object, or multiple objects through a sophisticated math expression capability that empowers ACE-SNMP to make aggregate tests of system performance. For example, the user can poll a single value such as “snmpInPkts.0”, or multiple values using syntax such as “expression: (%ifOutOctets.1 * 100) / %ipInReceives.0”

4.2.5 Tools Application

ACE-SNMP provides a suite of tools that are both typical and unique to traditional network managers. This tool set includes a MIB Browser and Table Viewer, Network Scanner, MIB Compiler, Performance Scanner, and various other tools to assess network performance. ACE-SNMP also includes a “auto-discovery” tool to assist in the setup and configuration of the ACE-SNMP system, as well as tools needed to setup and administrate the system (such as adding logins to restrict access to the management software)

Administrators can set permissions to execute these tools via the top level “Setup” application. Detailed information on each of these tools is provided in the ACE-SNMP online help facility.

4.2.6 Find Application

ACE-SNMP provides a “Find” application, which provides a central location for performing search operations. The “Find” application allows the user to search for device names, event data, MIB objects, trap data, and help files using wildcards and rule based search queries. These searches are performed using heuristically derived rules, which are intended to present most likely matches first in the list of matched data items.

The ACE-SNMP “Find” application may be of special importance when first learning about the ACE-SNMP system, since it affords an easy way for users to familiarize themselves with SNMP, and the capabilities of the ACE-SNMP program. Additionally, the search application provides numerous features to facilitate the real-time display of device states qualified by name or SNMP system identity.

The importance of the “Find” application grows with the number of managed devices and alarms. As more devices are added to the system, the amount management data scan become extraordinary. For example, if the user is managing 500 devices, and each device provides access to 500 MIB object, this constitutes 250,000 separate data items. The “Find” utility makes this extraordinary amount of data tenable to the user.

4.3 SNMP Paradigm Weaknesses

The SNMP Network Management paradigm suffers from several weaknesses, which are endemic to the protocol:

1. **Security Problems.** SNMP has no encryption standard. Attempts to codify a standard (in the form of SNMP-V2) failed in the late 1990's. The attempt to codify such a standard is ongoing. This is particular egregious because the read and write community names (which serve to limit access to a managed agent) are embedded in each SNMP message, and are easily sniffed off of the network. Notwithstanding the argument that, if a malicious sniffer exists on the network, loss of management information is the least of an administrator’s problems, the lack of security has served to reduce the usefulness of SNMP in the management of remote networks.

2. **Latency Problems.** SNMP is a request/response type protocol. This means that there is a delay between the type that a request is made and the response is received. The delay is typically small if the transaction is made on a local network, but can become a problem if the request is made across many hops. This causes SNMP to be less than useful in the management of remote networks, especially if large data gathering efforts from different remote devices is attempted.

Both of these limitations affect the use of SNMP as a remote management protocol. They are of less consequence when managing secure local networks.

ACE-SNMP addresses both of the above problems using HTTP to manage remote devices via server SNMP CGI programs. Specifically, HTTP incorporates a rock-solid encryption mechanism, which allows data to be transported across a public network with no security risk. Additionally, because SNMP requests are made at a server on a local network (in a remote location) and then shipped in aggregate back to the user via HTTP, latency is greatly reduced.

5. Conclusion

SNMP has shown to be a useful protocol for network management. Network administrators are well served by learning about SNMP, and using this protocol to monitor and manage their networks. Although the SNMP standard is somewhat complex, entailing specific knowledge of MIB objects as well as a large amount of configuration related to both SNMP agents and management software, the use of SNMP has been proven to be a cost-effective method of achieving a managed system.

Users wishing to experiment further with SNMP are encouraged to download the SNMX program, ACE-SNMP network manager, and ACE-ExAgent programs from DDRI, Diversified Data Resources, Inc.

APPENDIX 1: Useful MIB Objects

One of the challenges associated with learning SNMP is that a large number of data values exist in the SNMP MIB. It is often hard for newcomers to determine where to start, or which SNMP objects are pertinent. This appendix provides a top-level description of some of the standard MIB objects that are usually pertinent when browsing a MIB of any device.

A.1.1 Important MIB Scalar Values

This section lists eight MIB scalar objects that are especially pertinent to the assessment of any managed device. Network administrators wishing to become proficient in SNMP are encouraged to learn these objects by heart, since they will be referred to often when monitoring SNMP devices.

mgmt/mib-2/system/sysDescr.0

The "sysDescr.0" object is a read-only text string that contains a description of the managed device. This description is supplied by the managed device vendor and usually contains the hardware version, operating system type, and other useful information. This is always a good value to check first when browsing the MIB of a device since the value indicates the type of device being browsed.

mgmt/mib-2/system/sysContact.0

The "sysContact.0" object is a writable text string that contains the name of the person to contact for the managed device. This value is typically an e-mail address, URL, or personal name. The value often contains a phone number. This object is also good for testing the writability of the agent (i.e. whether the write community string is valid.)

mgmt/mib-2/system/sysUpTime.0

The "sysUpTime.0" object is a read-only time value that indicates how long the SNMP agent has been running. Note that, because SNMP agents usually start on device bootup, this value normally reflects the elapsed time since the platform was booted. The user can compare this value to other sysUpTime values to determine the relative stability of the managed device.

mgmt/mib-2/ip/ipInReceives.0

This object is a read-only counter containing the number of IP datagrams that have been received by the managed device. These are datagrams that include both the TCP and UDP layers. The value gives a good indication of how busy the managed device is. When compared with the "ipOutRequests.0" object (below) the administrator has a good idea of whether the box is mainly receiving data or transmitting data. Note that, when comparing this value with the ipInReceives value of another managed device, the administrator should scale values by the system up time (or some other elapsed time) of each managed device, since this counter indicates the number of datagrams that have been received since the managed device was rebooted.

mgmt/mib-2/ip/ipOutRequests.0

This object is a read-only counter that complements the ipInReceives value above. Specifically, this is the number of datagrams that have been output by the managed device. The total number of datagrams that have been processed by the device since boot up is the sum of this value and the ipInReceives value above. As with the ipInReceives value, when comparing this value with the ipOutRequests value of another managed device, the administrator should scale values by the system up time of each managed device.

mgmt/mib-2/ip/ipOutDiscards.0

This object is a read-only counter that indicates the number of outgoing IP layer packets that have been discarded by the managed device. If this value is other than zero, there it is likely that a problem with the network cabling or interface card of the managed device. Specifically, IP packets are discarded for one reason: they cannot be transmitted. Users can verify this by disconnecting an interface cable to the managed box and then attempting to transmit data. Once buffers fill up on the managed device, this should cause the ipInDiscards counter to increment.

mgmt/mib-2/ip/ipForwDatagrams.0

This object is a read-only counter that indicates the number of datagrams that have been forwarded, and is useful if the managed device is a gateway, bridge, or router. This counter is not pertinent if the box has only one interface. This value should periodically increment while a router is busy. If this value is not incrementing for a multi-interface device, then the device is not forwarding IP traffic (possibly because of a routing problem)

mgmt/mib-2/tcp/tcpCurrEstab.0

This object is a read-only gauge that indicates the total number of current TCP connections to the managed device. This is a useful metric of how loaded the managed device is. A TCP session can be an HTTP connection, FTP connection, Telnet connection, Mail connection, or any other process connection that uses TCP/IP. This value is a gauge, hence can easily be compared to other tcpCurrEstab values on the network to determine who has the most connections. The value is a good metric of activity and loading on the managed device.

A.1.2 Interface Table MIB Values

There are various tables included in the standard SNMP MIB. Generally, the foremost SNMP table that should be known and understood is the “ifTable” table, which contains the metric information associated with each interface of a device. This information is especially important when managing routers, switches, bridges, or gateways, since the user can determine which interfaces are busiest and which may be experiencing errors.

The “ifEntry” table contains a row for each interface in the system. Unlike some tables, which have complicated indexing, each row in the ifTable is indexed by a single integer value that ranges from one (1) to the number of interfaces for the box. For example, “ifDescr.3” references description of the third interface in the managed device.

The most pertinent objects to be familiar with are as follows:

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifDescr.N

This object is a read-only text string containing a description of the Nth interface. Since the ifTable is not required to present interfaces in a particular order, this value is important to distinguish the type of interface, such as “Loopback Interface”, “WAN Adapter”, etc. The interface description is usually provided by the vendor of the interface card.

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOperStatus.N

This object is a read-only enumerated value that identifies whether the interface is enabled or not. Because not all interfaces may be enabled in a particular managed device, administrators should inspect this object's value to see whether the Nth interface is “up” or “down”. Interfaces that are “down” are not operable. The user can enable an interface via SNMP by setting the value of ifAdminStatus.N, which is the only writable object in the interface table.

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifSpeed.N

This object is a read-only gauge indicating the number of bits per second for the Nth interface. For most agents, this value is assigned on agent startup and does not subsequently change. The value indicates the nominal bandwidth of the interface, and is useful for distinguishing high speed interfaces (such as T1 interfaces) with lower speed interfaces (such as ISDN interfaces.)

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInOctets.N

This object is a read-only counter that reports the number of octets that have been received by the Nth interface of the managed device. This is similar to the “ipInReceives” value described in the previous section, except the value is applicable only to the “Nth” interface. The value gives a good indication of how busy the particular interface is in receiving data. As with the ipInReceives object, this value is meaningful only when scaled by the system up time of the managed device (or other elapsed time.)

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOutOctets.N

This object is a read-only counter that reports the number of octets that have been output by the Nth interface of the managed device. This is similar to the “ipOutRequests” value described in the previous section, except the value is applicable only to the “Nth” interface. The value gives a good indication of how busy the particular interface is in outputting data. As with the ipOutRequests object, this value is meaningful only when scaled by the system up time of the managed device (or other elapsed time.)

/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInErrors.N

This object is a read-only counter that reports the number of input octets with errors on the Nth interface. This value, and the complementary ifOutErrors value, should be zero, or close to zero. If this value is nonzero, the administrator should inspect the cabling and network card for the specified interface. Other factors that can cause errors include noisy network transceivers on the particular subnet, or high interface traffic that causes buffers to be occasionally be flushed.

mgmt/mib-2/interfaces/ifTable/ifEntry/ifOutErrors.N

This object is a read-only counter that reports the number of output octets with errors on the Nth interface. This value, and the complementary ifInErrors value, should be zero, or close to zero. If this value is nonzero, the administrator should inspect the cabling and network card for the specified interface. This counter can also increment if all interfaces are being routed to a particular output interface, causing the bandwidth capability of the interface to be exceeded.

APPENDIX 2: Understanding ASN.1 Syntax

To determine the precise function of any MIB value, the user is referred to the ASN.1 definition files that are compiled into the SNMP network manager's MIB. These ASN.1 definitions are textual descriptions of each MIB object, which are human readable. Users wishing to explore the SNMP MIB in detail should examine the ASN.1 definitions to see the object type, access, and description.

SNMP administrators spend a fair amount of time studying ASN.1 files to determine the capabilities provided by private MIB objects. Although ASN.1 is a formal language, with many nuances, SNMP uses a simple subset of the ASN.1 syntax, described below. SNMP administrators should become familiar with ASN.1 syntax, and are well served to be at ease when reading and making changes to ASN.1 files.

A.2.1 Branch Object Identifiers

Certain SNMP objects have no value, but serve strictly as directories that contain other objects. These directories are defined with the OBJECT IDENTIFIER statement, an example of which is:

```
myBranch OBJECT IDENTIFIER ::= { parentBranch 100 }
```

The above statement defines a directory called “myBranch” whose parent branch is “parentBranch”. The number “100” is the unique object identifier (OID) for “myBranch” within “parentBranch”.

The user can trace back through the ASN.1 file to determine what the parent branch of “parentBranch” is, following the directory upwards until the root “internet” branch is determined. Within parentBranch, more object identifiers can be defined, providing that each object identifier has a unique name and number.

A.2.2 Scalar Object Definitions

Within a branch, the user can define more branches. The user can also define SNMP objects that have specific values. The syntax for declaring an SNMP object is as follows:

```
(objectname) OBJECT-TYPE
     SYNTAX (syntax)
     ACCESS (access)
     DESCRIPTION (description)
     ::= { (parent) (number) }
```

The definition is insensitive to white space, and can be formatted any variety of ways. Traditionally, MIB object definitions are written in a format similar to that above. Each part of the above declaration is described below.

1. **(objectname).** This is the official name of the SNMP object that is defined. ASN.1 requires that all object names begin with a lower case letter. By convention, this name is usually a mixture of upper and lower characters that is unique in the MIB.

2. **OBJECT-TYPE.** This is a required keyword that is always present in any leaf object definition.

3. **SYNTAX.** This is a required keyword indicating that the following token is the type of the object being defined. The SYNTAX defines the "type" of the object (not to be confused with the OBJECT-TYPE keyword, which defines the type of ASN.1 declaration.)

4. **(syntax).** This is the type of the object. A variety of types can be defined, as discussed in the next section. ASN.1 requires that all object types begin with an upper case letter. Various predefined types exist (such as "Counter", "Gauge", "DisplayString", "INTEGER"). Types can also be defined in the ASN.1 file based upon predefined types.

5. **ACCESS.** This is a required keyword indicating that the following token defines the access of the object.

6. **(access).** This is the access to the object. The access will usually be one of the following values: read-only, read-write, write-only, no-access. These access values have been augmented in later versions of SNMP to include other accesses (such as "write-create") but the basic access types apply to the management software and are rarely of any concern to the administrator.

7. **DESCRIPTION.** This is a required keyword indicating the following quoted string is a description of the object to document the functions and features of the object

8. **(description).** This is the text description of the object, used as commentary in the file. The description can span multiple lines of the ASN.1 file and is delimited by open and closing double quote marks. The description is supplied by the person who designed the SNMP agent, and serves to document the MIB value supported by the agent. These definitions can be clear and concise, or complex and misleading depending upon the expertise of the person who wrote the ASN.1 file.

9. **(parent).** This is the parent object container in which this leaf object resides. This value (along with the object number) always follows a "::=" assignment character, and is enclosed in curly braces. The value of (parent) refers to some SNMP object previously defined with an "OBJECT IDENTIFIER" statement. The parent name identifies the branch to which this leaf object is attached, similar to the way a file is always associated with a parent directory.

10. **(number).** Along with the (parent) value is the object identifier number, which indicates the unique numerical value of this object. Throughout the MIB, the (parent) (object) combinations must be unique. For example, if the final location of a MIB object is given as "::= { myObject 22 }", then there can be no other assignment "::= { myObject 22 }" in the ASN.1 definition. (Some flexibility is available with this rule, depending upon the scope of the MIB definitions, not discussed here.)

In addition to the required fields described above, a definition can have multiple optional keywords, such as a STATUS keyword, a UNITS keyword, or INDEX keyword. These extra fields may or may not be used by a network manager, depending upon the management software implementation.

It is important to note that the ASN.1 definition described above reflects the characteristics of the value supported by the SNMP agent. The SNMP agent characteristics are immutable and unaffected by changes the user may make to the definition. For example, the user can change the name of the object, the object definition, and possibly the syntax of the object without affecting the agent's operation. It is common for network administrators to make changes to an object's ASN.1 definition, and compile these changes into the management software, thereby changing the name of an SNMP object, or some other parameter.

When making changes to the object, it is important to note that the location of the object in the MIB, defined using the "::=" operator, cannot be changed. If the user makes such a change to the ASN.1 file, then it is probable that the management software will no longer be able to access the SNMP object.

A.2.3 Object Syntax Definitions

The SYNTAX part of an object declaration requires further elaboration. SNMP defines certain basic types: "Counters", "Gauges", "INTEGERS", "DisplayStrings", "IpAddress", "TimeTicks", and a few others. The syntax of object has various special conventions that allow the user to specific enumerated values to integers, and derive new types based upon the existing types. These special considerations are discussed in the paragraphs that follow.

A.2.4 Integer Syntax

The INTEGER type can be either a basic integer over a range of values, or can be an enumerated type. For example, the following syntax associates specific enumerated values with an SNMP INTEGER object:

```
myEnumObject OBJECT-TYPE
     SYNTAX INTEGER
     {
          first(1),
          second(2),
          third(3),
          fourth(4)
     }
     ACCESS read-only
     DESCRIPTION "An enumerated value"
     ::= { parentObject 22 }
```

In the above declaration, the value of “myEnumObject” can be an integer ranging from 1 to 4, where each of these numbers has a “tag” that labels the specific value. This provides a way of specifying an integer value by a more descriptive name. Enumerated values are actually integer values that are formatted by the network management software (when displayed) to provide more readable values.

The INTEGER type can also be a raw integer representing a unit value of some type. In this case, the integer is interpreted by the management software as a number rather than a label.

A.2.5 Dervived ASN.1 Types

ASN.1 syntax allows the user to define new types based upon the existing predefined types. For example, the following statement derives a new type from an existing type.

```
NetworkAddress ::= IpAddress
```

After making such a declaration, anywhere that “NetworkAddress” is found in the ASN.1 file, the MIB compiler immediately substitutes the “IpAddress” type in its place. Types are often derived from enumerated types to simplify the readability and programming of the ASN.1 file. For example, the following can be used:

```
MyEnumValue ::= INTEGER
{
     first(1),
     second(2),
     third(3),
     fourth(4)
}
```

Following the above declaration in an ASN.1 file, anywhere that the syntax of a value is defined as “MyEnumVal”, the above-enumerated value is substituted. Because many objects may use the above enumeration, this can simplify the programming and readability of the ASN.1 file.

There is an alternate method of deriving types using the special TEXTUAL-CONVENTION declaration. This was a change made to SNMP by the SNMP-V2 proposal, and allows a more verbose statement of a type derivation. The syntax has become widely accepted.

A.2.6 Table Types

SNMP tables have a special “type” statement. SNMP Tables are identical to SNMP branches, except that the objects contained in the table can be considered columns rather than scalar objects. They also have a fairly rigorous set of syntactical requirements that serve to obfuscate and confuse their basic simplicity.

Each table object has syntax as follows:

```
(tablename) OBJECT-TYPE
     SYNTAX  SEQUENCE OF (tabletype)
     ACCESS  not-accessible
     DESCRIPTION (description)
     ::= { (parent) (number) }

(entryname) OBJECT-TYPE
     SYNTAX  (tabletype)
     ACCESS  not-accessible
     DESCRIPTION (description)
     ::= { (tablename) 1 }

(tabletype) ::= SEQUENCE {
     (column1) (column1type),
     (column2) (column2type),
     (columnN) (columnNtype) }
```

As evident above, table syntax is confusing. Fortunately, the syntax can be entirely ignored by the user. The complexity of a table definition can be thought of as an idiosyncrasy of ASN.1 syntax. The basic rules of a table can be summarized as follows:

1. By convention, each SNMP table contains a name incorporating the “Table” keyword. This convention is almost perfectly universal in its implementation. For example, the object “myTable” indicates, by its very name, that it is a table-type object.

2. Beneath each table is a single branch object with a name incorporating the “Entry” keyword. Again, this convention is almost perfectly universal in its implementation. Beneath “myTable” will always be a single branch containing the table data, and that object will have the name “myEntry”.

3. Within “myEntry” is a series of SNMP objects that are identical to the OBJECT-TYPE definitions presented earlier, except that the suffix of these objects do not necessarily have to be “.0”, but can instead be simple or complex indexes to the table rows in dot notation.

The above rules are not strictly codified in the SNMP standard. (For example, a MIB compiler cannot rely on these rules since they do not have to be strictly followed.) However, any friendly ASN.1 file will follow the above rules, thereby simplifying the understanding of the MIB by a reader.

A.2.7 Caveats To ASN.1 Files

Unfortunately, many vendors do not precisely follow conventional ASN.1 syntax in their file definitions. Experience has shown this is especially true with regard to the ASN.1 “IMPORT” directives. Also, third party vendors occasionally take a cavalier attitude towards ASN.1 syntax, incorporating obscure but acceptable syntactical ASN.1 variations that network management software may find troublesome. Such variations in ASN.1 syntax may, from time to time, require some hand editing of these files by the administrator before they can be compiled into the management software.

DDRI, Diversified Data Resources, Inc. 1999
http://www.ddri.com, e-mail: techsup@ddri.com
USA ordering information: 1-800-233-3374
Fax: 1-415-898-7331