The Simple Times™

THE BI-MONTHLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS℠
VOLUME 2, NUMBER 3 MAY/JUNE, 1993

The Simple Times is an openly-available publication devoted to the promotion of the Simple Network Management Protocol (SNMP). In each issue, ***The Simple Times*** presents: a refereed technical article, an industry comment, and several featured columns. In addition, some issues include brief announcements, summaries of recent publications, and an activities calendar. For information on submissions, see page 20.

In this Issue:

Technical Article

Barry Bruins, Network General Corporation

In this issue: *Windows SNMP: an SNMP API for MS Windows Applications*

Windows SNMP is a developing industry standard for Microsoft Windows applications that need SNMP services. The standardization effort began in the summer of 1992 when Microsoft published their *Windows/NT SNMP Programmer's Reference*. This interface that was shipped in the MS Windows/NT SDK did not include provision for asynchronous interfaces (asynchronous interfaces allow applications to issue requests and continue executing instead of blocking while waiting for the response) and was quite different from existing SNMP APIs built upon commercial Windows protocol stacks. Additionally, no provision for SNMP version 2 was made.

Thus a desire emerged to create a common API between Windows and Windows/NT environments that was full featured enough to provide a base for SNMP-based management applications. While the initial discussions were conducted among a closed group, this was opened up in a birds of a feather session at INTEROP March '93. Consensus was reached on two issues: 1) Windows SNMP must include provision for SNMPv2; and, 2) only the manager API would be standardized in the first version. The complexities of a sub-agent interface in an SNMP API were too daunting to attack at this time and would slow down the whole effort.

A second BOF was held at the IFIP Third International Symposium on Integrated Network Management in April. This BOF turned out to be just an informational session as many of the contributors (and commentators) were not present. The remainder of this article discusses the Windows SNMP work as proposed in the April 23, 1993 draft, which is still subject to considerable change.

The important attributes of Windows SNMP are:

- supports SNMP version 1 and version 2;
- provides transport protocol independence;
- supports both synchronous and asynchronous programming models; and,
- allows multiple management applications to co-exist.

SNMPv1 and SNMPv2 Support

The Windows SNMP work provides an important guide to management platform providers in developing APIs that provide evolution to SNMP version 2 transparently. The SNMP version 2 framework has an Administrative Model that can seem extremely complex to the uninitiated. The interaction of privacy, authentication, proxy and context capabilities can seem like a Gordian knot. The good news is that all of this complexity can be hidden from the application developer as well as the user.

Windows SNMP API calls to send and receive messages refer to: an entity handle, a context handle, and a quality of service parameter. By understanding how to use these three concepts, an SNMPv2 application is as easy to write as an SNMPv1 application — in fact, the application need never know which version of SNMP is being used.

The entity database is the local management entity's Party MIB. The Party MIB from SNMPv2 is the collection of the `partyTable`, `contextTable`, `aclTable`, and, `viewTable`. The contents of the Party MIB define what collections of objects the management application can access and the types of access allowed. Depending upon the sophistication of the Windows SNMP stack, there could be zero, one or many entity databases per machine. In SNMPv1-only implementations, the entity database is really non-existent. In SNMPv2 implementations, there can be a different entity database, per-application or per-user, in order to give different privileges of access to the network management data to different users.

The context specifies a collection of managed object resources. In SNMPv1, a context is analogous to an IP address (or domain name) and community string association. The IP address identifies the endpoint of a request and the community string implicitly identifies the accessible MIB objects at that endpoint. In SNMPv2, contexts also identify the MIB objects, but they do so explicitly, so a context identifier can be used to look up source and destination party pairs to conduct the requested operation. In the Windows SNMP API, user-friendly context names are looked up with the `SnmpStrtoCtx()` function.

Quality of service indicates whether authentication and/or privacy is required. The following table shows how these parameters are used in Windows SNMP for SNMPv1 and SNMPv2 operations:

SNMP	Entity	Context	QOS
v1	community string	IP address domain name	0
v2	database name	context name	required services

Local database names and local context names are implementation-specific. (The author believes that some convention is necessary here to achieve application portability.)

Transport Independence

There are no transport specific references in the Windows SNMP API. The transport domain of a particular context is contained within the entity database. An implementation can attempt to use the local context name to determine hints about the transport of choice in two situations: for SNMPv1 transactions, and for unknown cases where default entries are to be created in the entity database.

(A)synchronous Programming Model

Synchronous programming implies that the application will treat an SNMP request as a long subroutine call. This is the simplest model to use when programming because the system takes care of everything. When a call to `SnmpSendMsg()` is issued, the protocol stack sends the request and the application is suspended until the response is received.

For simple tools, this mechanism is fine. For management applications that need to poll multiple devices or keep their user interface responsive, this model will not suffice. This was the only model envisioned in the MS Windows/NT specification because it was assumed that each SNMP request would be spawned as a separate thread to be executed. Multiple threads and thus multiple outstanding SNMP requests could be executing simultaneously. (For really high-powered management applications, the overhead for spawning the environment for a new thread for each SNMP request would be overwhelming.)

Asynchronous programming implies that the API calls will return immediately as soon as the request is queued to the network. The Windows SNMP implementation would then, asynchronously, notify the application via a message to its window that a packet had been received for it. In processing the notification, the application could then retrieve the response and process it.

Multiple Management Application Coexistence

Windows SNMP requires that compliant implementations provide service to multiple applications concurrently. Providing service concurrently means that the API must keep track of which application issued each request so that the sequence number in the response can be used to map back to the application making the request. Additionally, traps may require duplication if multiple applications request them. SNMPv2 `inform-requests` can only go to a particular application as a response is required.

Windows SNMP Specification

Windows SNMP interfaces are divided into the following categories:

- communication functions;
- entity functions;
- message functions;
- variable binding functions; and,
- utility functions.

The communication functions are used both to activate the API and to exchange SNMP traffic: `SnmpOpen()` provides the API with a handle to the window that the management applications wants to be notified on for traps and asynchronous responses, returning a session handle to be used in future API calls. `SnmpClose()` tells the API to release the session. `SnmpSendMsg()` sends a message that the application provides. The software behind the API will check the entity database to determine if there are any party entries in the local party MIB that can communicate to the requested context with the requested quality of service. This function also accepts a timeout value so the application can control the length of time to wait on a response to the message. `SnmpRecvMsg()` is used to receive any asynchronous message such as a `snmpV2-trap`, `inform-request`, or `response` received when using the asynchronous programming model. Finally, `SnmpRegister()` is used to register interest in `snmpV2-traps` or `inform-requests`. An OID is specified to identify which notifications an applications is interested in. Observant readers will note that SNMPv1 traps are not supported by the API. That's right! Although an implementation of Windows SNMP must support SNMPv1, it uses the SNMPv2 format for traps when communicating with a management application. (The SNMPv2 documents explain how to map between the two formats.) By automatically providing this mapping, management applications are simplified and management applications needn't worry about which version of the protocol is in use.

The entity functions are used to specify entity and context information. At present, the functions allow only lookup operations. A future version of the API will likely allow local configuration as well.

The remaining functions are all "utility" subroutines performing a myriad of low-level tasks such as creating, manipulating, and releasing SNMP PDUs, variable bindings, and so on. These are fairly typical SNMP API routines found in many other implementations.

Windows MIB API

The Windows MIB API is a companion specification that allows a management application to walk the MIB object definitions that are known to this platform. While it is an optional part of the Windows SNMP specification, it provides useful features not found on some commercial SNMP management platforms. The most obvious use might be for a MIB browser, having the MIB data on hand which allows applications to pick up `DISPLAY-HINTS` and enumerated values in order to provide a richer user interface.

This specification could be expanded with the SNMPv2 `AGENT-CAPABILITIES` macros to designate which agents implement which objects.

Where things stand

The Windows SNMP Specification has progressed quite a bit since March, and representatives from several companies have indicated that they will support the standard when it is complete and there is sufficient demand. However, there has recently been a flurry of activity on the Windows SNMP mailing list, so some change in the specification is possible before finalization. You can obtain the current version of the Windows SNMP specification, using anonymous FTP from:

```
ftp.netmanage.com
```

There is also a discussion group. To subscribe, send mail to:

```
winsnmp-request@microdyne.com
```

the mailing list itself is called:

```
winsnmp@microdyne.com
```

Industry Comment

Marshall T. Rose

In this issue, I had hoped to publish a companion to a keynote presentation made at the Third International Symposium on Integrated Network Management, which was held in San Francisco in April. As usual, however, the page count on this issue didn't allow it. So, if you're interested, it will be published in the June issue of *ConneXions*. (If you don't subscribe to the industry newsletter *ConneXions*, you should — for information call +1 415–941–3399).

Applications and Directions

Steven L. Waldbusser

In this issue: *A Look at the Host Resources MIB*

As the Host Resources MIB nears the IETF standards track, it is constructive to evaluate what it is and how it promises to improve systems management of internet hosts.

SNMP has been implemented widely on many varieties of platforms, from routers and hubs to workstations and PCs. To date, the standard MIB modules implemented on hosts have been oriented towards the networking aspects of those hosts. The only systems management MIB modules for hosts have been proprietary. Nevertheless, these proprietary modules have been written for many of the host types available such as PCs, Macs, and workstations.

After a fair amount of experience had been gained with these proprietary MIB modules, a desire grew for a standard module which would incorporate the functions common amongst the various vendor-specific modules. Such a MIB would better allow third-party applications to be written that could manage various types of internet hosts. Given that most network installations are multi-vendor and have several different types of host systems, a common MIB module and common applications would allow a network operations center to manage these various types of hosts with a common user interface.

Enough desire grew for this standard host MIB that a working group was formed. That working group has defined a MIB, the Host Resources MIB, that will soon be evaluated with respect to the standards track.

The Host Resources MIB Module

The MIB is divided into six groups of objects: the `system` group, the `storage` group, the `devices` group, the `running software` group, the `running software performance` group, and, the `installed software` group.

The `system` group has several objects that describe overall system parameters such as the number of users and processes, and where the operating system is loaded from.

The `storage` group provides utilization information on all forms of storage on the system, including RAM, disk drives, and, memory buffers.

The `devices` group provides configuration and fault information about all devices on the system. Some types of devices such as printers and disk drives have more detailed information specified than others. For example, one kind of device is a network interface. MIB-II already defines extensive information on interfaces, so the `devices` group really doesn't need to duplicate those features.

The `running software` group lists the software running on the system while the `running software performance` group provides performance data about each piece of software running on the system. The `installed software` group lists all the software installed locally on the system.

Expected Uses

MIB modules should be written to provide variables with known uses rather than providing all possible information in a specific area without regard to its usefulness. The Host Resources MIB was written with this in mind — in fact, some variables were dropped from early versions of this MIB module because their usefulness could not be proven.

The Host Resources MIB provides many types of functions, primarily in the fault management, configuration and asset management, and, performance management areas.

The Host Resources MIB helps a number of system management jobs that a network manager might face. With this MIB module, a network manager can download an inventory of all equipment on various LANs across an organization, without regard to what types of systems the equipment is attached to. In addition to determining how much memory and disk is installed in each computer, the types and versions of other hardware and software components can be retrieved. Obsolete versions of software or hardware can be flagged and incompatibilities between various hardware and software components can be detected. Disk drives can be monitored to make sure that routine backup procedures are being followed and that the disks are not running out of space.

Textual information on a system, such as that found in file names, is returned in a format that allows any international language or character set. This allows the MIB module to monitor systems that have been internationalized, and (again) shows that SNMP is suited for this task.

A Typical Fault Diagnosis Exercise

In a typical environment, the network/systems manager might get a call from a user on a PC or workstation complaining of difficulty running an application. The manager can retrieve information over the network to diagnose the problem. Initially, the manager might check the RAM, disk and buffer usage on the system to see if any allocation failures have been experienced or if any are near their limits. The manager can check to see if too many users are on the system, too many processes are in use, or if the system is otherwise overloading the CPU. For each device, the manager can check the status and and if any errors have occurred that might be the cause of the user's problem — for example, printers indicate whether they need paper or any other attention.

After exploring each of these areas of the system, if the problem has not been found, it is likely a software problem. Software systems are becoming more and more complex, and it is especially important to verify the compatibility amongst various pieces of software, such as the operating system, the windowing system, the network file system, and, the applications software. The Host Resources MIB allows the manager to remotely determine what versions of these pieces of software have been installed, and what versions are running. This allows useful applications to be written that know of these incompatibilities and are smart enough to automatically flag such errors. If a piece of software that is running is causing a problem, the manager can use the MIB (with SNMP's security) to kill that process.

The Host Resources MIB provides a solution to many problems that once required vendor-specific MIB modules. Now that this soon-to-be standard MIB module has stabilized, those functions can be implemented in a vendor-neutral manner. In addition to the advantages this provides to application builders, users are provided with a common interface for performing systems management across multiple types of host platforms. This common interface will prove to be useful to solve a variety of problems users have.

Ask Dr. SNMP

Jeffrey D. Case

Dear *Dr. SNMP*,
I'm a poor and confused user of a wide variety of SNMP agents in our wonderful and complex networking world. I am puzzled, intrigued, and confused by a notion in the definition of MIBs which you might be able to resolve.

Each object in a MIB definition has a `STATUS` field which specifies optional or mandatory. In a popular publication, *The Simple Book*, by a certain Marshall T. Rose, I found the statement

> "It is a convention of the Internet-standard MIB that no object may have an `STATUS` field of optional. All objects are considered to be mandatory."

This I find confusing. Why the need for the `STATUS` field in the first place (but this is only an aside).

What I'd REALLY like to know is the following: suppose an agent supports a certain MIB group, let's say the `interfaces` group of MIB-II, but the device it manages happens to deal not in octets but "nonets". In this case, the `ifInOctets` object can't contain any useful information. But, because of the convention, the agent has to provide it. What value for this counter should be returned in a `get-response`? Are there any magic cookie values, by convention, which indicate that the number I receive is nonsense?

— *Needing Magic Cookie in New Hampshire*

Dear *Needing Magic Cookie in New Hampshire*,
Down on the farm, we have a saying:

> "Truth and roses have thorns about them."

Your question touches two aspects of the SNMP philosophy on which I am sure that Dr. Rose and Dr. SNMP would be in strong agreement.

First, network management agents should always tell the truth. When managing a network, receipt of no information is better than receipt of erroneous information, allowing for appropriate measurement error, of course.

Second, in SNMP, options are generally considered harmful because they lead to waste. This is because agent writers must implement them but manager station applications writers cannot use them. Agent writers must implement them due to market pressures resulting from competitive bidding (some bid writers use these kinds of features to prevent unwanted vendors from being able to bid). A management station application must be able to perform, whether or not an optional variable is present. Exceptions abound, but, in general, if a manager can ever do its job without an optional variable, it can always do without it, so optional things tend to remain unused.

The `STATUS` field is present in all MIB documents, not only standard ones. Standard MIB documents are the minority — there are far more MIB objects defined in other MIB documents, such as vendor-specific MIB documents. Optional status may make more sense in these other MIB documents.

In your example of the system which implements "nonets", the agent cannot and should not implement the variable. It should implement all other objects in the `interfaces` group that it can, but claims should not be made that the agent is compliant with the MIB document, since its implementation is necessarily incomplete.

The agent should return the `noSuchName` error response to an SNMPv1 `get-request`, or the next available object in response to a `get-next` request. In

SNMPv2, the agent should return the `noSuchObject` exception in the `response`. This new exception was added in SNMPv2 to address this problem, and others.

There are no values, by convention or otherwise, which indicate the number is nonsense. All values in the range of a `Counter` variable are valid; hence, none can be invalid. The way that an agent should say that it cannot support a variable is to say that it cannot support that variable in the way that one says that it cannot support that variable. Unfortunately, this makes it more difficult to claim in one's marketing literature that all variables are supported, especially the ones that are not supported. Some vendors have a simple, albeit inappropriate, approach to this. They lie.

Dear *Dr. SNMP*,
I understand a new working group has been chartered to re-write the `interfaces` portion of MIB-II. Why is this necessary? Why can't we just fix the problems without massive changes?
— *Preservationist in Pennsylvania*

Dear *Preservationist in Pennsylvania*,
Down on the farm, we have a saying,

> "It is fruitless to attempt to indoctrinate a superannuated canine with innovative maneuvers."

Or something like that. Dr. SNMP has discovered a new book called *The Saurus*. Even though it is dry reading, Dr. SNMP believes that if he can figure out the plot, this new book will help him provide increasingly sophisticated answers in the future.

The `interfaces` group has had known problems since the late 1980s. Some people think that the interfaces layer is the layer "above" or "behind" a connector. Some people think that the interfaces layer is the layer "below" IP. In systems where these statements are equivalent, there are few problems. However, the plumbing within today's systems is increasingly twisted. Consider a relatively simple example: a router which carries DECnet tunneled within IP packets via an X.25 link duplexed over two serial lines. In this and other cases, it is difficult, if not impossible, to map this structure onto the existing MIB objects. Furthermore, there are no documents which specify how this mapping should be done in standard ways.

Although these problems were known when MIB-II was written, they were not addressed. Similarly, the SNMPv2 effort consciously opted to defer consideration of these problems for several reasons. First, these are difficult problems, for which the solutions are not readily apparent. Second, these are MIB evolution issues which do not have any apparent protocol implications. Third, the effort to define and deploy SNMPv2 was felt to be "big enough" without tackling unrelated MIB redefinition efforts. Finally, an important part of the SNMPv2 coexistence and transition plan is that no MIB redefinition is necessary for compatibility with SNMPv2. Redefinition of MIB 2 as a part of the SNMPv2 design effort would have added confusion to this message.

We should all look forward to the timely completion of the efforts to solve these problems.

Security and Protocols

Keith McCloghrie

With the publication of the SNMPv2 RFCs, more implementors are now taking a look at the specifications. One of the questions which several of them have asked is how much of the SNMPv2 administrative framework, and how much of the Party MIB, is it necessary to implement. This question is particularly relevant since the `MODULE-CONFORMANCE` macros in RFC 1447 only specify the compliance levels with respect to the implementation of MD5 and DES; they fail to document the intended subsets of Party MIB implementation corresponding to the compliance levels. Thus, in this issue, we'll look at minimal implementations of the Party MIB for various types of agents.

Initial Party/Context Identifiers

First, recall that the Party MIB (RFC 1447) identifies two subtrees in the `OBJECT IDENTIFIER` (OID) naming tree, one for use by convention as initial Party identifiers, and the other for use by convention as initial Context identifiers. These conventions define six parties and two contexts for each IP address.

For the six parties defined for the agent having IP address a.b.c.d, three parties are local to the agent and three are local to a manager which wishes to communicate with the agent having that IP address. Each of the three pairs has different authentication and privacy parameters, as follows:

```
initialPartyId.a.b.c.d.1
    noAuth/noPriv executing at the agent
initialPartyId.a.b.c.d.2
    noAuth/noPriv executing at a manager
initialPartyId.a.b.c.d.3
    md5Auth/noPriv executing at the agent
initialPartyId.a.b.c.d.4
    md5Auth/noPriv executing at a manager
initialPartyId.a.b.c.d.5
    md5Auth/desPriv executing at the agent
initialPartyId.a.b.c.d.6
    md5Auth/desPriv executing at a manager
```

The two contexts defined for each IP address, initialContextId.a.b.c.d.1 and initialContextId.a.b.c.d.2, are used by convention to give different access rights to the unauthenticated parties and to the authenticated parties, respectively. The corresponding definitions of access control entries and MIB views are also specified as part of the convention.

The OIDs in the above conventions are specific to agents implementing SNMPv2 over UDP/IP, and we'll use these in the discussion below. However, for other transport domains, similar conventions can be defined using other OIDs.

A Minimal Insecure Agent

Now, let's look at the minimal implementation of the Party MIB by an insecure agent, i.e., one which implements neither MD5 nor DES. An agent of this type supports only noAuth/noPriv parties, and thus each and every request should have identical access rights. Thus, it need have only one MIB view, one context, two access control entries and two parties. The two parties are: initialPartyId.a.b.c.d.1 and initialPartyId.a.b.c.d.2. The one context is initialContextId.a.b.c.d.1. The one MIB view would provide access to everything in the agent's MIB, and thus a single entry in the viewTable (e.g., for internet) would suffice. One of the access control entries allows the manager to issue requests (e.g., get-request) to the agent, using the two parties to access the one context. The other access control entry allows traps to be generated and transmitted to a manager.

With this configuration, any number of managers can access the agent using the noAuth/noPriv parties, and one manager can receive traps from the agent. To support this configuration, there is no need for the agent to support the creation/deletion of entries in any of the tables in the Party MIB; instead, all entries can have a StorageType value of permanent. In fact, the only object in the Party MIB, to which it needs to support write access, is the partyTAddress for initialPartyId.a.b.c.d.2, in order to configure the destination of the traps it generates. (Note that configuring an address for initialPartyId.a.b.c.d.2 does not stop managers at other addresses from using this party for requests/responses, since the agent ignores partyTAddress when sending responses, and instead sends them back to the address from which the request was received.)

To summarize, the minimal configuration is:

```
partyIdentifier:    initialPartyId.a.b.c.d.1
partyIndex:                 1
partyTAddress:              a.b.c.d:161
partyLocal:                 true
partyAuthProtocol:          noAuth
partyPrivProtocol:          noPriv
partyStorageType:           permanent

partyIdentifier:    initialPartyId.a.b.c.d.2
partyIndex:                 2
partyTAddress:              address for traps
partyLocal:                 false
partyAuthProtocol:          noAuth
partyPrivProtocol:          noPriv
partyStorageType:           permanent

contextIdentifier:
                  initialContextId.a.b.c.d.1
contextIndex:               1
contextLocal:               true
contextViewIndex:           1
contextStorageType:         permanent

aclTarget (dest. party):    1
aclSubject (src party):     2
aclResources (context):     1
aclPrivileges:              get, get-next,
                              get-bulk, set
aclStorageType:             permanent

aclTarget (dest. party):    2
aclSubject (src party):     1
aclResources (context):     1
aclPrivileges:              snmpV2-trap
aclStorageType:             permanent

viewIndex:                  1
viewSubtree:                internet
viewType:                   included
viewStorageType:            permanent
```

A Regular Insecure Agent

To allow an insecure agent to send traps to multiple destinations, an additional party and a corresponding additional access control entry are required for each additional destination. These additional access control entries have their aclTarget set to their corresponding additional party, but all of them have initialPartyId.a.b.c.d.1 as their aclSubject.

The additional parties can be set up with null addresses (e.g., 0.0.0.0) at initialization time, and therefore there is still no need to support creation/deletion for any tables in the Party MIB, and there is no need for write-access to any object other than partyTAddress. Since the first trap-destination is configured as the address of initialPartyId.a.b.c.d.2, one way to assign the OIDs for the additional parties would be to use initialPartyId.a.b.c.d.12, initialPartyId.a.b.c.d.22, and so on.

So, the additional configuration (per trap destination) is:

```
partyIdentifier:   initialPartyId.a.b.c.d.12
partyIndex:                12
partyTAddress:             address for traps
partyLocal:                false
partyAuthProtocol:         noAuth
partyPrivProtocol:         noPriv
partyStorageType:          permanent

aclTarget (dest. party):   12
aclSubject (src party):    1
aclResources (context):    1
aclPrivileges:             snmpV2-trap
aclStorageType:            permanent
```

A Minimal Secure Agent

Now let's look at a minimal secure agent, with a single authenticated manager plus any number of other managers which can access it using noAuth/noPriv parties. To be secure it needs to implement MD5 (but not DES). It also needs, as compared to the insecure agent mentioned above, some additional configurations and some additional support for the Party MIB.

These additional configurations are: two authenticated parties, an extra context for use by the authenticated parties, and an extra access control entry to allow the authenticated parties to access the extra context; these would be the conventional initialPartyId.a.b.c.d.3 and initialPartyId.a.b.c.d.4 parties and the conventional initialContextId.a.b.c.d.2 context. Also, since unauthenticated managers will normally be given less access rights than the authenticated manager, a distinct MIB view is needed for each context; it is appropriate that these be set up at installation time to the view subtrees defined by convention for the context accessible by unauthenticated parties.

In its support for the Party MIB, the secure agent need not allow creation/deletion of entries in the partyTable, the contextTable, or the aclTable; instead, all entries in these tables can have a StorageType of permanent. On the other hand, the creation/deletion of entries in the viewTable is useful to allow the authenticated manager to configure how much of the MIB is accessible by unauthenticated managers. (Since a minimal agent will not support access control with instance-level granularity, it should validate the creation of new view entries to ensure that the OID of the specified view subtree is equal to, or a substring of, the OID of at least one of the MIB objects it supports.) Apart from the viewTable, where write access to all columns is needed in order to support creation of new entries, the subset of objects for which write access should be supported is quite small:

- to set the trap destinations; partyTAddress,
- to tell the agent that the manager supports larger response messages: partyMaxMessageSize; and,
- in support of MD5: partyAuthClock, partyAuthPrivate, and partyAuthPublic

So, the additional configuration needed with an authenticated manager is:

```
partyIdentifier:    initialPartyId.a.b.c.d.3
partyIndex:                3
partyTAddress:             a.b.c.d:161
partyLocal:                true
partyAuthProtocol:         md5Auth
partyPrivProtocol:         noPriv
partyStorageType:          permanent

partyIdentifier:    initialPartyId.a.b.c.d.4
partyIndex:                4
partyTAddress:             0.0.0.0:0
partyLocal:                false
partyAuthProtocol:         md5Auth
partyPrivProtocol:         noPriv
partyStorageType:          permanent

contextIdentifier:
                  initialContextId.a.b.c.d.2
contextIndex:              2
contextLocal:              true
contextViewIndex:          2
contextStorageType:        permanent

aclTarget (dest. party):   3
aclSubject (src party):    4
aclResources (context):    2
aclPrivileges:             get, get-next,
                             get-bulk, set
aclStorageType:            permanent
```

As well as these additions, the aclPrivileges for the previous access control entries would likely be reduced to interrogation requests, view 2 would be set to the internet subtree, and view 1 would initially be restricted to the system, snmpStats, and snmpParties subtrees.

A Secure Agent with Multiple Managers

Unlike noAuth/noPriv parties, a separate pair of parties is needed for each manager which communicates with an agent using md5Auth.

There are two reasons why separate parties are needed. First, there is the potential for "clock leapfrog" if multiple managers use the same pair of parties concurrently; "clock leapfrog" occurs when clock synchronization by one manager advances the clock and thereby requires the other manager to clock synchronize, which thereby requires the first manager to clock synchronize, which thereby causes the other manager to ... Second, when a clock value reaches the maximum value, a manager station reset the clock backward and change the secret key in a single operation. This would be problematic if multiple managers share parties and therefore keys because there is no mechanism for the multiple managers to exchange information about the new keys.

For each such additional authenticated manager, a secure agent needs additional configurations for: two additional authenticated parties (one local and one remote), an additional context (allows the different authenticated managers to have different MIB views), and an additional access control entry (to allow the additional parties to access the additional context). At installation-time, the additional parties can be set up with a StorageType of permanent, having the same protocols, parameters, and secrets as the initialPartyId.a.b.c.d.3 and initialPartyId.a.b.c.d.4 parties (i.e., as if they had been cloned from those parties), but with a RowStatus of notInService. This allows them to exist (and remain undeleted since they are permanent) in the table but to be unused until they are activated by setting their RowStatus to active. One way to assign the OIDs for these additional parties would be to use initialPartyId.a.b.c.d.13 and initialPartyId.a.b.c.d.14 for the first additional pair, initialPartyId.a.b.c.d.23 and initialPartyId.a.b.c.d.24 for the second additional pair, and so on.

Therefore, the creation and deletion of entries in the partyTable, the contextTable, and the aclTable is not required to be supported by a secure agent with multiple managers. The creation and deletion of viewTable entries is required, as is write access to the same subset of objects in the Party MIB, as identified above for a minimal secure agent. In addition, write access to aclPrivileges is useful in order to alter the configuration of which managers can issue set-requests.

So, the extra configuration needed per additional authenticated manager is:

```
partyIdentifier:   initialPartyId.a.b.c.d.13
partyIndex:                13
partyTAddress:             a.b.c.d:161
partyLocal:                true
partyAuthProtocol:         md5Auth
partyPrivProtocol:         noPriv
partyStorageType:          permanent

partyIdentifier:   initialPartyId.a.b.c.d.14
partyIndex:                14
partyTAddress:             0.0.0.0:0
partyLocal:                false
partyAuthProtocol:         md5Auth
partyPrivProtocol:         noPriv
partyStorageType:          permanent


contextIdentifier:
                   initialContextId.a.b.c.d.12
contextIndex:              12
contextLocal:              true
contextViewIndex:          2
contextStorageType:        permanent

aclTarget (dest. party):   13
aclSubject (src party):    14
aclResources (context):    12
aclPrivileges:             get, get-next,
                             get-bulk, set
```

More Complicated Agents

After reading the above, you might now be wondering whether any SNMPv2 implementations are required to implement the full capabilities of the Party MIB. Well, the answer is that proxy agents do need to implement the full Party MIB, and SNMPv2 entities acting in a dual role (e.g., those which implement the Manager to Manager MIB) probably need to also. Agents which support multiple local entities (e.g., repeaters and bridges) may not require a full implementation, but will certainly require additional contexts, and possibly additional parties. In addition, those agents which wish to provide the ability to send encrypted SNMP messages will also need to implement DES, the corresponding additional configurations, and write access to additional MIB objects. Similarly, there are other capabilities provided by the Party MIB which agents may wish to provide by implementing the appropriate parts, such as providing write access to contextLocalTime in order to support the writing of values for use only after the next reboot of the agent.

Standards

David T. Perkins

SNMPv2 IS HERE! SNMPv2 IS HERE! Yes, it's worth repeating. As predicted in the last issue, the documents defining SNMPv2 have been published as RFCs and are on the standards track at the Proposed level (the first rung in the three level standards ladder). No other documents have been published as RFCs since the last issue; however, there are two in the pipeline to be published: an updated version of the Network Management Tool Catalogue; and, a MIB module for MultiProtocol Interconnect over X.25. In addition, some RFCs were moved to historic: RFC 1230, the Token Bus MIB, due to lack of operational experience; and, the previous SNMP security documents, RFC 1351, RFC 1352, and RFC 1353, due to the publication of SNMPv2 documents.

Recently Published RFCs

RFC 1441 - Introduction to SNMPv2 (Proposed Standard)
Presents an overview of the version 2 framework of the Internet-standard Management Framework.

RFC 1442 - SMI for SNMPv2 (Proposed Standard)
Specifies how to define the operands of SNMP operations using a subset of the Abstract Syntax Notation (ASN.1). These operands are either management objects or notifications. These definitions are grouped together in ASN.1 modules which are called information modules. An ASN.1 MACRO, `MODULE-IDENTITY`, is defined which is used to identify and specify information such as revision and author of information modules. Two other ASN.1 macros, `OBJECT-TYPE` and `NOTIFICATION-TYPE`, are defined which are used to concisely convey the syntax and semantics of management objects and notifications.

RFC 1443 - Textual Conventions for SNMPv2 (Proposed Standard)
Gives the definition of textual conventions and specifies the initial set of textual conventions. Two of these, `DisplayString` and `PhysAddress`, are known from RFC 1213, MIB-II. Additional ones are based on definitions in existing SNMPv1 MIBs. The most important new one is the `RowStatus` textual convention which requires 12 pages to define!

RFC 1444 - Conformance Statements for SNMPv2 (Proposed Standard)
Defines three ASN.1 macros, `OBJECT-GROUP`, `MODULE-COMPLIANCE`, and `AGENT-CAPABILITIES`, used to group management objects, specify the minimally acceptable conforming implemenation of a MIB module, and specify the actual implementation done in an agent.

RFC 1445 - Administrative Model for SNMPv2 (Proposed Standard)
Overviews the framework used to provide authentication and integrity, privacy, authorizations, indirect operations, and, control of unsolicited notifications. Also defined is the format of messages between SNMPv2 entities. This mind-altering document is scheduled for a non-technical re-write. The document is basically a "quick turn" of RFC 1351.

RFC 1446 - Security Protocols for SNMPv2 (Proposed Standard)
Gives the details of security in SNMPv2 based on the MD5 digest algorithm for authentication and integrity, and the DES algorithm for privacy. The threats, goals, constraints, and, security services are first defined, followed by the mechanisms to realize them. Included are descriptions of solutions to operational concerns such as clock and secret distribution, initial configuration, clock synchronization, crash recovery, and recommended practices. "Ported" from RFC 1352, this RFC is also in need of an editorial revision.

RFC 1447 - Party MIB for SNMPv2 (Proposed Standard)
Defines the MIB objects needed to implement the administrative framework of SNMPv2. This MIB consists of four tables: for parties, contexts, access control, and views. The party table identifies the source and destination sources in SNMPv2 communications for the agent. The context table consists of entries that either identify a MIB family for a particular local entity, or specify the parameters for a proxy relationship, while the view table consists of families of collections of MIB objects. Finally, the ACL table has entries for all allowable combinations of source, destination, context, and operation. Based on RFC 1353, it requires an understanding of RFCs 1445 and 1446 before it can be used.

RFC 1448 - Protocol Operations for SNMPv2 (Proposed Standard)
Shows in ASN.1 and in textual elaborations the format and interpretation of SNMPv2 Protocol Data Units (PDUs). (The format of the messages in which the PDUs are enclosed is specified in RFC 1445.) The SNMPv2 PDU types are: `get-request`, `set-request`, `get-next-request`, `get-bulk-request`, `inform-request`, `response`, and, `SNMPv2-trap`.

RFC 1449 - Transport Mappings for SNMPv2 (Proposed Standard)
Specifies how SNMPv2 maps onto an initial set of transport domains. These are UDP (the preferred choice), OSI CLTS, DDP, IPX, and proxy to SNMPv1.

RFC 1450 - MIB for SNMPv2 (Proposed Standard)
Contains the definitions of the MIB objects that describe the behavior of an SNMP entity. These are basically a

cleaned up version of the SNMP objects from MIB-II.

RFC 1451 - Manager-to-Manager MIB (Proposed Standard)
Specifies a generalization of the MIB objects in the RMON MIB used for detection and notification of events. The alarm table is used to define the objects that are being monitored, and the ID of their associated context. The event table records occurrences of specified events, while the event-notify table identifies the contexts associated with each event.

RFC 1452 - Coexistence between SNMPv1 and SNMPv2 (Proposed Standard)
Shows how to map between SNMPv1 and SNMPv2 operations and management information definitions.

Road Map for SNMPv2 Documents

In total, the page count for the 12 RFCs that comprise the SNMPv2 definition is over 400 pages! How is the best way to read through these documents? First, remember they are "reference" documents and are not "tutorial" documents; hench, the reader should already understand ASN.1, network management fundamentals, and TCP/IP terminology. If the reader has a good grasp of SNMPv1, then the SNMPv2 documents are just a next step. Without this background, another source, such as a book on SNMPv1, should be consulted first.

Of course, the *Introduction to SNMPv2* document should be read first. The next document to read should be *The Structure of Management Information* (SMI), but it is best to peek at the *Protocol Operations* document to get preview of the operations that can be performed on management information. The SMI document is augmented by the the *Textual Conventions* document. Reading of the *Conformance Statements* document can be deferred until the reader is ready to do "comparison shopping". When products start to come out, my journalist friends will be very interested in the conformance statements — these specify the "minimal requirements" for conformance and the "full disclosures" for implementations. Now it is time to read over the *Protocol Operations* document in detail, but don't worry if you don't completely understand MIB views yet.

If you have gotten through the above documents, congratulate yourself and take a break. You just finished reading 135 pages.

Go get your favorite painkiller — aspirin, a pitcher of a powerful alcoholic beverage, or a punching bag to prepare for the next set of documents. These are the *Administration Model*, *Security Protocols*, and the *Party MIB*. If you like real tough brain teasers, get a copy of RFC 1321 (the specification of MD5), and copies of the documents describing DES. When you have finished with these (and if they haven't finished you), give yourself a double pat on the back. You made through another 148 pages.

The *SNMPv2 MIB*, *Transport Mappings*, and *Coexistence* documents are easy to read. They represent a fast 68 page read.

The last bump in the road is the *Manager-to-Manager* (M2M) MIB. You can detour it if you want to finish now. If not, go back and read the RMON MIB. This puts you in the proper context to now read the M2M MIB. The document should make sense if you understood the *Administrative Model* document. (It describes the determination of destinations of `SNMPv2-traps` and `inform-requests`.) You got through this — the last 36 pages.

That's the end of the "road map". When the next issue of ***The Simple Times*** comes out, the IETF will have been on the road — traveling to Europe. In that issue, we'll report on the European reaction to the IETF standards process.

Summary of Standards

SNMPv1 Framework (Full Standards):

- 1155 - Structure of Management Information (SMI);
- 1157 - Simple Network Management Protocol (SNMP);
- 1212 - Concise MIB definitions; and,
- 1213 - Management Information Base (MIB-II).

SNMPv2 Framework (Proposed Standards):

- 1441 - Introduction to SNMPv2;
- 1442 - SMI for SNMPv2;
- 1443 - Textual Conventions for SNMPv2;
- 1444 - Conformance Statements for SNMPv2;
- 1445 - Administrative Model for SNMPv2;
- 1446 - Security Protocols for SNMPv2;
- 1447 - Party MIB for SNMPv2;
- 1448 - Protocol Operations for SNMPv2;
- 1449 - Transport Mappings for SNMPv2;
- 1450 - MIB for SNMPv2;
- 1451 - Manager-to-Manager MIB; and,

- 1452 - Coexistence between SNMPv1 and SNMPv2.

Full Standards:

- 1213 - Management Information Base (MIB-II).

Draft Standards:

- 1398 - Ether-Like Interface Type MIB.

Proposed Standards:

- 1229 - Extensions to the generic-interface MIB;
- 1231 - IEEE 802.5 Token Ring Interface Type MIB;
- 1239 - Reassignment of experimental MIBs to standard MIBs;
- 1243 - AppleTalk MIB;
- 1253 - OSPF version 2 MIB;
- 1269 - BGP version 3 MIB;
- 1271 - Remote LAN Monitoring MIB;
- 1285 - FDDI Interface Type MIB;
- 1286 - Bridge MIB;
- 1289 - DECnet phase IV MIB;
- 1304 - SMDS Interface Protocol (SIP) Interface Type MIB;
- 1315 - Frame Relay DTE Interface Type MIB;
- 1316 - Character Device MIB;
- 1317 - RS-232 Interface Type MIB;
- 1318 - Parallel Printer Interface Type MIB;
- 1354 - SNMP IP Forwarding Table MIB;
- 1368 - IEEE 802.3 Repeater MIB;
- 1381 - X.25 LAPB MIB;
- 1382 - X.25 PLP MIB;
- 1389 - RIPv2 MIB;
- 1406 - DS1/E1 Interface Type MIB;
- 1407 - DS3/E3 Interface Type MIB;
- 1414 - Identification MIB;
- 1418 - SNMP over OSI;
- 1419 - SNMP over AppleTalk; and,
- 1420 - SNMP over IPX.

Experimental:

- 1187 - Bulk table retrieval with the SNMP;
- 1224 - Techniques for managing asynchronously generated alerts;
- 1227 - SNMP MUX protocol and MIB;
- 1228 - SNMP Distributed Program Interface (SNMP-DPI); and,
- 1238 - CLNS MIB.

Informational:

- 1147 - A network management tool catalog;
- 1215 - A convention for defining traps for use with the SNMP;
- 1270 - SNMP communication services;
- 1303 - A convention for describing SNMP-based agents; and,
- 1321 - MD5 message-digest algorithm.

Historical:

- 1156 - Management Information Base (MIB-I)
- 1230 - IEEE 802.4 Token Bus Interface Type MIB;
- 1232 - DS1 Interface Type MIB;
- 1233 - DS3 Interface Type MIB;
- 1283 - SNMP over OSI;
- 1284 - Ether-Like Interface Type;
- 1298 - SNMP over IPX;
- 1351 - SNMP Administrative Model;
- 1352 - SNMP Security Protocols; and,
- 1353 - SNMP Party MIB.

Working Group Synopses

Frank J. Kastenholz

This column is a summary of activities. There is no substitute for actually participating in a working group. Even if you cannot go to the meetings, you can subscribe to the mailing lists. Included in each working group's summary is the address of the group's mailing list. To subscribe, simply append "`-request`" on to the local-part of the address. For example, the submission address for the SNMP general discussion list is snmp@psi.net

so to subscribe, you'd send a message to

snmp-request@psi.net

If you are interested in a group's activities and do not subscribe to the mailing list, you should!

SNMP General Discussion

Submissions: snmp@psi.net

A person posted a question asking about development of an API for Microsoft Windows. The mailing list is winsnmp@microdyne.com

Someone asked a question about the atTable. This person pointed out that MIB-II defines the instance-identification of the atTable to be atIfIndex and atNetAddress, yet there is an additional numeric value between the atIfIndex and atNetAddress components of the instance-identifier. The response said that the additional number is the type-code for atNetAddress, a 1 for IP-addresses in this case.

Someone asked how to add a row to a table with the set-request. The answer that came back was to simply set the variables of the row, and it should be created. An additional question in this line was whether the "read-only" variables should be set in this manner. These variables should have DEFVALs set, or be derivable from, e.g., the instance-identification of the row.

Someone asked how to design a table so that changes made to the table do not take effect until the system reboots. Some suggestions were to differentiate between the "next-initialization" and "current operational" values based on community strings, or on instance-identifiers for tables, or to have separate tables.

Someone asked for guidance on how best to extend the acceptable syntax for the SIZE clause for his freely available MIB parser. Several examples were included in the request. The gist of the extensions were to allow multiple SIZEs to be specified. One response requested that the changes be extended to INTEGERs as well.

A person asked about multi-management-protocol environments. This person will be adding some SNMP-managed network devices into an environment that has some systems managed with CMIP. A dual-protocol manager solution was generally preferred.

There was an extensive discussion on auto-topology, that is, determining network contents and structure via SNMP and other network management tools. One response asked that any auto-discovery processes default to OFF on manager stations, as large networks may experience problems during installation if it is enabled. This is primarily because these processes rely on broadcasts of various types in various ways. This plea was supported by many people. One person wondered whether auto-discovery will work with SNMPv2, due to its security features. If there are well-known, default, parties, that allow minimum function and have noAuth and noPriv, then SNMPv2 is not a problem. In general, this led to the notion that a standard, default, access policy be available in all boxes so that they do not need to be configured to be discovered. This would be equivalent to having a community string of "public" in SNMPv1. The use of a well-known multicast address in this context was also discussed. One tool, used in a commercial product, is a "comprehensive ping" which goes through all IP addresses in a given range and pings them. If an answer comes back, a device has been discovered. Other tools will extend this, looking at MIB tables (ARP, routing, interfaces, etc.) and so on, and attempt to deduce the network structure (as opposed to merely inventorying devices) based on the results. A user mentioned a package called Fremont that attempts to do several auto-discovery functions. A paper is available at ftp.cs.colorado.edu in pub/cs/techreports/schwartz/ASCII/Fremont.txt.Z It was pointed out that not all systems support SNMP, so the results are, at best, inconclusive. Another possible strategy is to "snoop" on the network, and examine the interesting packets (such as routing, ARP, 802.1 Spanning tree, and so on). It was pointed out that, as encryption (not just of SNMP packets) becomes more commonplace, snooping into packets becomes more problematic. The difference between auto-discovery (determining which nodes were on the network) and auto-topology (deducing network structure) was elaborated on. However, the first seems to be a pre-requisite for the second. One person pointed out that there are many degenerate protocols and configurations in networks and that any good auto-topology system must be able to make sense out of them. The needs of security (which wishes to prevent unknown unauthorized people from doing bad things) were discussed, in light of discovery procedures, which require relatively open access to as many systems and MIB objects, by potentially unknown managers, as possible.

Someone asked about mapping the IP Address table, which identifies a single interface for the IP Address, to bridges, which have multiple "interfaces" but one address. One response was to consider the IP "interface" as a virtual interface, layered on top of the bridge and all of its real interfaces. A second approach mentioned was to simply pick one of the bridge interfaces as the IP interface.

Someone had a question about the design of tables. This person was designing a table for an interface and had to have an "interface" and then a channel within that interface. The person wanted to know how to design this sort of "table within a table". The answer was to have two tables, one containing information just for the interface and indexed by the interface number, and the second containing information on the channels and indexed by interface and channel numbers.

Someone asked if there were repositories of just the ASN.1 source of the MIBs (instead of having to manually remove all of the non-ASN.1). A package called `mstrip` was offered as a solution to this problem (it is available from `ftp.synoptics.com` as a part of the `SMIC` package).

A discussion about strategies for re-sending `set-requests` was held. The question arises of what to do when the `get-response` to a `set-request` is not received. Did the `set-request` fail to get to the agent, or was the set successful and the response lost? It was claimed that a manager has no way of telling. Simply resending the original `set-request` could be a problem if the request maps onto some action and the manager truly wants the action to occur only once (e.g., "send $100 to Jim"). Resending the `set-request` with the same `request-id` and relying on the agent to ignore requests with "previously seen" `request-ids` was suggested. It was pointed out that this was not a behavior specified in the protocol. It would also place additional burdens on the agent in that it would have to save additional SNMP state (some point out that SNMPv2, with `RowStatus`, is moving away from this model). Another suggestion was to get the variable that was set and see if it had the "new" or "old" value in it, resending the `set-request` if it had the old value. However, this does not allow "at-most-once" semantics, though intelligent MIB design (i.e., design objects as "set state off or on" instead of "toggle state"). Also, additional MIB objects could be created that allow for such semantics (e.g., timestamps and so on). The possibility of multiple manager stations acting like Laurel and Hardy (one constantly turning something off, the other constantly turning it on) was raised. Locking was also suggested as a mechanism to deal with this, as was the `TestAndIncr` textual convention of SNMPv2. A long and heated discussion of the merits of locking was held — one person felt that locking was the answer to the problem, no on else publicly agreed.

Someone asked which MIB groups a 10BaseT hub should support. Several MIB groups were enumerated as likely candidates, `system`, `interface`, `snmp`, along with the Hub and Ethernet-like MIB modules. In addition, since the hub is probably using UDP for SNMP, the `udp`, `ip`, and `icmp` groups should also be implemented.

Someone asked a question about BER encoding of unsigned integers. The response was that BER, and ASN, do not have the concept of unsigned integers, so encoding unsigned values must be done as if they were signed. Thus, 0xffffffff should be encoded as 0x020500ffffffff.

Someone asked about whether there is a working group to design a MIB for SONET equipment. The AToM working group is responsible for developing this (see below).

Someone asked whether an agent is allowed to silently discard requests that overflow internal buffers. Several polite things were suggested (e.g., send a trap, or increase your buffer size), but no one said it was illegal. It was noted that most all conditions that could cause a packet to be lost within the agent (no buffers, bad UDP checksum, etc.) are counted, so the proper counter, at least, should be incremented.

Someone asked about encoding Asian languages in `DisplayStrings`; specifically, ignoring or relaxing the "printable ASCII" part of its definition. One answer was that you could place UNICODE in `OCTET STRING`s.

Someone asked whether it is legal to index a table other than the `ifTable` with `ifIndex`. The answers that came back all said yes.

A question was asked about how to demultiplex SNMP requests that are sent to a device when that device contains multiple virtual devices. (This is not the same situation as a chassis, where the chassis contains several real physical devices — in this case, each physical device is instantiated entirely in software.) The answer that was posted indicated that this is identical to a proxy relationship; that, in fact, the proxied-for side of a proxy relationship can be virtual.

There was the usual flamefest, carried out on several mailing lists, about whether SNMPv2 is a good thing or not, and whether the SNMPv2 working group should be disbanded or not, and EXACTLY how many pages are in the SNMPv2 specifications (versus the SNMPv1 specifications). It was greatly amusing to watch otherwise respectable members of the community descend in their arguments to a level that could only have been pre-planned, since the absurdity, innuendo, invective, and gratuitous *ad hominem* personal attacks could not have occurred by accident.

A question was asked if the length of OIDs is limited to 64 bytes in SNMPv1 or not. The responses indicated that some MIB compilers, OID printing routines, agents and manager stations might have implementation limits, and that one might wish to limit one's OID lengths for safe interoperability, but there is no limit in the protocol specifications.

Someone asked if there is a MIB for managing OS/2 systems. One was posted to the list.

Someone expressed concern over the IP-centricity of SNMP, and that, since most SNMP managers use UDP/IP, then to be manageable, a device needs UDP/IP, which carries its infrastructural needs with it (ICMP, ARP, configuration, and so on). It was pointed out that use of a common underlying protocol suite enhances ubiquity.

Someone asked whether there is a `perl` interpreter with SNMP support built into it. Several answers came back, describing a package called `snmperl`, available in many different places.

Someone asked when reference implementations of SNMPv2 would be available from CMU or 4BSD/ISODE. One of the principals of these efforts said that they should be available by the end of May.

As usual, several people insisted on carrying out the same conversation on both the SNMP and SNMPv2 mailing lists. I've reported these conversations on only one of the lists, however.

Someone posted a message seeking the SunNet Manager files necessary to manage an HP9000 system. They were pointed to a general repository of such MIB modules, available for anonymous FTP at `zippy.telcom.arizona.edu` in `/pub/snmp/agents`

Appletalk/IP Working Group

Submissions: `apple-ip@cayman.com`

No SNMP-related traffic to report.

AToM MIB Working Group

Submissions: `atommib@thumper.bellcore.com`

This is a new working group. The mailing list started with discussions on ATM connection management, the relationship between the working group's charter and the work done by the ATM Forum on ILMI (interim link management), and service management.

BGP Working Group

Submissions: `iwg@ans.net`

No SNMP-related traffic to report.

Bridge MIB Working Group

Submissions: `bridge-mib@nsl.dec.com`

A call was posted to the mailing list for implementation experience in preparation for a review of the standardization status of the MIB by the IESG. Several responses were directed to the mailing list, presumably others were sent privately.

One person requested information on algorithms that may be used to analyze the locations of nodes on a bridged LAN. A detailed algorithm was posted in response.

An announcement of an Internet-Draft for source-routing bridges was posted. Some minor errors were pointed out.

An announcement for a new version of the original MIB was posted. This version was sent to the IESG for review for advancement to Draft Standard status. The changes to the MIB, as compared to the Proposed Standard version, were all boilerplate.

Character MIB Working Group

Submissions: `char-mib@decwrl.dec.com`

The working group was re-activated to evaluate RFCs 1316–1318 (currently proposed standards) with respect to the standards track.

Chassis MIB Working Group

Submissions: `chassismib@cs.utk.edu`

The minutes from the working group meeting at the Columbus IETF were posted.

An intense and detailed discussion was held about mapping between entities and resources. A problem was pointed out in that the original model could not handle the case where a resource (e.g., an interface) was used by multiple entities (e.g., a bridge and a router). It was suggested that this was a common configuration and ought to be supported. A long response, along with several examples, indicated that this was not a problem and that the original poster was slightly confused. This discussion was too voluminous and detailed to easily summarize in this column.

The MIB went through several "on-the-list" revisions, with new versions posted to the list, along with requests for clarification of various parts, and modification and revision requests.

DECnet Phase IV MIB Working Group

Submissions: `phiv-mib@jove.pa.dec.com`

The working group was re-activated to evaluate RFC 1289 (currently a proposed standard) with respect to the standards track.

Ethernet MIB Working Group

Submissions: `enet_mib@ftp.com`

The Ethernet MIB mailing list has been fairly quiet. The only recorded activity was a question about how to get numbers for the `ifExtnsChipSet` object. The questioner raised a point that one vendor was not represented in the list presented in the MIB. These numbers are `OBJECT IDENTIFIERS` and as such may be assigned out of any OID subtree. For vendors whose parts are not represented in the MIB, the recommended practice is to get an OID from IANA out of the enterprises subtree, and then allocate OIDs to represent their parts from that branch.

A new working group, the Interfaces MIB working group, has been formed to deal with advancing the Ethernet MIB further in the standards track. Most discussion of the Ethernet MIB in the future will probably occur on that working group's mailing list (see below).

FDDI MIB Working Group

Submissions: `fddi-mib@cs.utk.edu`

The working group continued development of a MIB that corresponds to X3T9.5's SMT version 7.3. A draft of this MIB has been developed and posted to the Internet-Drafts repository.

Someone asked about the ranges for `FddiTimeNano` and `FddiTimeMilli`. The questioner noted that the ranges are specified as signed 32–bit quantities and asked whether they should be unsigned instead. The response was that `INTEGERs` are defined as being 32–bit signed quantities so the range for these objects properly is 0..2147483647.

A question was raised about the accuracy of `fddiSMTTimeStamp` and `fddiSMTTransitionTimeStamp`, which are defined as 64–bit, 80ns counters in the SMT specifications. The MIB defines these as 32–bit, millisecond counters. The response was that the working group had discussed the matter at some length and this accuracy was deemed acceptable.

Frame Relay Service MIB Working Group

Submissions: `frftc@nsco.network.com`

This is a new working group.

Host Resources MIB Working Group

Submissions: `hostmib@andrew.cmu.edu`

Several syntax errors in the Internet-Draft were pointed out. Several clarifications were also requested. A final plan of action to finish off the MIB was posted.

IEEE 802.3 Hub MIB Working Group

Submissions: `hubmib@synoptics.com`

A revised version of the MAU mib was posted to the mailing list.

Minutes of the working group's meeting at the Columbus IETF were posted.

Someone asked why `rptrMonitorPortRunts` is not included in the `rptrMonitorPortTotalErrors` error summary. The answer was that runts are considered normal network events. Furthermore, runts occur at a very high frequency, relative to other errors, so the runt count would drown out the count of true errors.

IDPR Working Group

Submissions: `idpr-wg@bbn.com`

No SNMP-related traffic to report.

IDRP for IP Working Group

Submissions: `idrp-for-ip@merit.edu`

No SNMP-related traffic to report.

Interfaces MIB Working Group

Submissions: `if-mib@thumper.bellcore.com`

This is a new working group.

IPLPDN Working Group

Submissions: `iplpdn@nri.reston.va.us`

No SNMP-related traffic to report.

IS-IS Working Group

Submissions: `isis@merit.edu`

No SNMP-related traffic to report.

Mail and Directory Management Working Group

Submissions: `ietf-madman@innosoft.com`

This is a new working group. There was considerable discussion on whether defining managed objects for mail gateways was within the scope of the working group. There was also considerable discussion on possible liaison with ANSI X3. Neither topic was resolved.

Modem Management Working Group

Submissions: `modemmgt@telebit.com`

This is a new working group. There was considerable discussion on objects already defined by the CCITT for the purpose of modem management.

NOCtools Working Group

Submissions: `noctools@merit.edu`

No SNMP-related traffic to report.

OSPF Working Group

Submissions: `ospfigp@gated.cornell.edu`

No SNMP-related traffic to report.

PPP Working Group

Submissions: `ietf-ppp@ucdavis.edu`

The PPP Working group has settled on the four mibs (LCP, IPCP, Bridging, and Security) that have been in development. Final versions of the MIBs were posted to the mailing list. There were some minor comments requesting clarifications. The most significant was in determining what values to return for MIB objects for which no value is available until after negotiations with the peer have completed. The working group decided to have the values of these objects be undefined until the link is up, and text to that effect has been added to the MIBs.

The documents were forwarded to the IESG for review and standardization.

RIP Working Group

Submissions: `ietf-rip@xylogics.com`

Some possible implementation problems with the MIB were pointed out.

Interfaces are referenced in the MIB by their IP addresses. However, a capability of BSD UNIX is to have point-to-point interfaces share an address with a non-point-to-point interface (such as Ethernet). As the RIP MIB is specified, multiple interfaces with the same address would have to be reported.

RIPv2's Domains allow multiple Domains to be run on a single interface, allowing different subsets of routers to be reached with each domain. The MIB does not support this. The note suggested that `rip2IfConfDomain` be added to the INDEX clauses of `rip2IfConfTable` and `rip2IfStatTable`.

Remote Monitoring (RMON) Working Group

Submissions: `rmonmib@jarthur.claremont.edu`

Some problems implementing the History Group with certain hardware were pointed out. Specifically, this particular controller only reports collisions on packets that it actually attempts to transmit, and that the general implementation must do a lot of work to ensure that error type information is correlated with the received, errored packet; in fact, it might be necessary to stop the adaptor card after each packet, while the necessary information is recorded.

Documentation for one RMON implementation was announced. This documentation can be obtained from `dnpap.et.tudelft.nl` in `/pub/btng/rmon-develop.tar.Z`. Several people engaged in a review and discussion of the document on the mailing list.

A brief discussion was held regarding what to do when you cannot support the semantics of a particular object (e.g., the interface card does not count a certain type of event). It was pointed out that the wrong answer (though commonly implemented) was to return 0; the correct thing to do is to not implement the variable and return `noSuchName` when the variable is requested. The correct solution also prevents one from claiming that one has implemented the MIB.

One person asked whether the counters in the ring station table should be reset to 0 when the entry goes from `active` to `inactive` or `forcedRemoval`, and then back to `active`. Some people wish to see the counters set to 0, since this way they reflect an accurate count of what happened since the station entered the ring. It was pointed out that counters have no defined starting value, so a manager station cannot rely on the counters starting at 0. Also, resetting the counters in this way would break with precedent; the manager station should probe the table often enough to detect the condition. The mailing list discussions seemed to prefer the latter (do not reset) options.

Development of an RMON Compatibility Test Suite, as a graduate student project, was announced. A set of basic tests was listed and additional input solicited. Most of the functions were actually testing the SNMP functions associated with the MIB, such as making sure that get-next would get table entries in the correct order, and so on. Additional "weird" OIDs were suggested such as too-short, too-long, and non-existent. It was pointed out that the suite did not actually test the RMON functions, and that such tests should be done. Performance tests were also suggested.

A question was asked concerning the filter matching algorithm. Suppose the filter mask is "L" bytes long and only the first "M" bytes of the mask contain any one bits (M < L). If a received packet is less than L bytes long (but greater than or equal to M bytes long) and the packet matches the filter pattern identified by the first M bytes of the mask, should the packet be accepted? Citing the following text of RFC 1271, most people claim that the packet should not be accepted:

> "If the packet is too short and does not have data corresponding to part of the filterPktData, the packet will fail this data match."

One dissenting opinion claimed that sending the "trailing zeros" might simply be a matter of laziness on the part of the manager station, that these zeros should be removed from the mask, and therefore the packet accepted.

Someone asked whether hostOutPkts, hostOutOctets, hostOutErrors, matrixSD(DS)Pkts, matrixSD(DS)Octets, and matrixSD(DS)Errors should be updated if a packet has a CRC error. The general consensus was that bad packets should not be used to instantiate new rows of the table, but updating existing entries would be acceptable. If the addresses of the bad packet are already in the table, then it is very probable that the addresses of the bad packet are correct (the error was someplace else in the packet).

Quite a discussion was held on the usefulness of attempting to attribute bad packets to specific source addresses. It was suggested that even though there is less than 100% reliability in this attribution, attempting to use the source address to determine the locus of the error packet's source would be useful to help determine if the problem was due to bad hosts, bad packet switches, bad wiring, and so on.

A question was asked, regarding TR-RMON, whether Spanning Tree BPDUs should update sourceRoutingStatsSingleRoutesBroadcastFrames and sourceRoutingStatsSingleRoutesBroadcastOctets. One person responded yes.

Another person asked whether there was an FDDI RMON MIB. The response was that the working group's current tasks are finishing the token-ring RMON MIB, advancing current MIBs in the standards track, and then addressing "RMON-II", which would involve a higher level of statistics.

SNA DLC Services MIB Working Group

Submissions: snadlcmib@apertus.com

This is a new working group. A call for enterprise-specific MIB modules in this area was made.

SNA NAU Services MIB Working Group

Submissions: snanaumib@thumper.bellcore.com

This is a new working group. A call for enterprise-specific MIB modules in this area was made.

SNMP Security Working Group

This mailing list has been terminated, with its activity moved to the SNMPv2 working group's mailing list.

SNMPv2 Working Group

Submissions: snmp2@thumper.bellcore.com

A major discussion was held about export restrictions of DES and security technology in general. There was a lot of talk, with all the obligatory references to the boyz with sunglasses, trench-coats, and short haircuts, incompetent bureaucrats, and so on. Not a lot was said, however, which, this author supposes, is typical for discussions on security technology and the law. Export of encryption technology from the US, and many other countries (primarily those in the COCOM regime), is controlled. How does this affect the DES component of SNMPv2? First and foremost, DES is not required for SNMPv2. Second, it was pointed out by some people that DES implementations are available outside of the US, though several of the implementations are in COCOM countries, and presumably export from those countries is also controlled. One person pointed out that leaving "hooks" in which one could insert controlled technology is also against the COCOM rules. It was pointed out that the COCOM rules cover not only exporting of software; but also boxes containing embedded controlled software, for example, a router containing SNMPv2 with DES would be subject to control. One poster pointed out that the US/COCOM restrictions vary according to both the use of the cryptographic technology (it's generally easier to get a license for authentication than for privacy), and the destination.

The author would like to point out that anyone contemplating exporting SNMPv2, or SNMPv2-bearing devices, containing security code (authentication and/or privacy) should investigate the matter with legal counsel. The exact rules and procedures in each country are different, and you should know what you are doing before you do it.

Someone asked what "serialization" means. The answer is that it means the "encoding of ASN.1 specified data according to the Basic Encoding Rules".

A person asked about whether an API to SNMP should be synchronous or asynchronous. A response said that some agent systems had relatively slow response times (on the order of seconds) and it would not be good to tie up a manager station waiting for a response. Another respondent pointed out that the Windows SNMP API specification currently being developed has both synchronous and asynchronous interfaces.

A major flamefest was held on the SNMPv2 mailing list, cross-posted to the IETF list about whether SNMPv2 was good or not, and whether it was forced down the throats of the IETF. The discussion included whether security was a "good thing" or not. This author found it uproariously funny that one of the main protagonists in this discussion was one of the original people who, a year ago, made the strong arguments that: 1) there should be one transition, from SNMPv1 to SNMPv2 with security; and, 2) we need to have a tightly-focused working group to address these issues. In effect, the complaint was that we got what we asked for.

An intense discussion on whether instance granularity in views is a "good thing" or not. The computational burden of enforcing this granularity was questioned. It was pointed out that an agent could simply refuse to do instance granularity (or any other level of granularity, for that matter) by rejecting attempts to create such views with the appropriate SNMPv2 error codes. It was pointed out that, if an agent allows a view to be created with a certain level of granularity, then that agent must enforce that granularity in subsequent operations.

A discussion was held on whether any form of party-changed notifications is desirable. This could be a trap, or simply timestamping the party table entries with sysUpTime when the entry was last changed. A plea, which no one disagreed with, was posted for thinking carefully about these issues before making any changes.

TCP Client Identity Protocol

Submissions: `ident@nri.reston.va.us`

No SNMP-related traffic to report.

Token Ring Remote Monitoring Working Group

Submissions: `rmonmib@jarthur.claremont.edu`

See RMON above.

Trunk MIB Working Group

Submissions: `trunk-mib@saffron.acc.com`

No SNMP-related traffic to report.

Uninterruptable Power Supply Working Group

Submissions: `ups-mib@cs.utk.edu`

A discussion ensued on whether it is acceptable to return 0 for unsupported objects. The answer is NO, return `noSuchname` instead.

A main item of discussion was the `upsBattery` group, with many of the objects receiving intense scrutiny by the working group members. Some of the issues raised were: 1) it is difficult to get meaningful, reliable, information out of the data presented by some of the objects in the group since the underlying phenomenon being measured can only be estimated, with unknown accuracy; 2) some of the objects, reflecting ambient environmental conditions, may vary in the quality of the information they provide since the sensors providing the raw data may be placed at different locations from one vendor (or model) to the next, and therefore may measure slightly different environments; and, 3) some objects require manual maintenance and, lacking such maintenance, might give a false sense of security.

A strawman proposal for the `input` group was posted to the list. The usefulness of `upsInputBlockouts` was discussed, as were issues relating to determining the voltages on the input.

A proposal for an `alarm` group was posted to the mailing list. The definition of `upsAlarmStopNoticeIssued` was briefly discussed in relation to this group. The syntax of `upsAlarmId` was changed to reflect the fact that `INTEGER` objects are 32–bit signed numbers. Most, if not all, of the alarms were reviewed and analyzed in far too great a detail to go into in this column.

A strawman for the `output` group and `bypass` group was posted.

X.25 MIB Working Group

Submissions: `x25mib@dg-rtp.dg.com`

No SNMP-related traffic to report.

Activities Calendar

- INTEROP August 93
  August 23–27, San Francisco, CA
  For information: +1 415–941–3399

Publication Information

The Simple Times is published with a lot of help from the SNMP community.

Publication Staff

Coordinating Editor:

Dr. Marshall T. Rose	Dover Beach Consulting, Inc.

Featured Columnists:

Dr. Jeffrey D. Case	SNMP Research, Inc. University of Tennessee
Frank J. Kastenholz	FTP Software, Inc.
Keith McCloghrie	Hughes LAN Systems, Inc.
David T. Perkins	SynOptics Communications, Inc.
Steven L. Waldbusser	Carnegie Mellon University

Contact Information

Postal:	***The Simple Times*** c/o Dover Beach Consulting, Inc. 420 Whisman Court Mountain View, CA 94043–2186
Tel:	+1 415–968–1052
Fax:	+1 415–968–2510
E-mail:	`st-editorial@simple-times.org`
ISSN:	1060–6068

Submissions

The Simple Times solicits high-quality articles of technology and comment. Technical articles are refereed to ensure that the content is marketing-free. By definition, commentaries reflect opinion and, as such, are reviewed only to the extent required to ensure commonly-accepted publication norms.

The Simple Times also solicits terse announcements of products and services, publications, and events. These contributions are reviewed only to the extent required to ensure commonly-accepted publication norms.

Submissions are accepted only in electronic form. A submission consists of ASCII text. (Technical articles are also allowed to reference encapsulated PostScript figures.) Submissions may be sent to the contact address above, either via electronic mail or via magnetic media (using either 8-mm `tar` tape, $\frac{1}{4}$-in `tar` cartridge-tape, or $3\frac{1}{2}$-in MS-DOS floppy-diskette).

Each submission must include the author's full name, title, affiliation, postal and electronic mail addresses, telephone, and fax numbers. Note that by initiating this process, the submitting party agrees to place the contribution into the public domain.

Subscriptions

The Simple Times is available via electronic mail in three editions: *PostScript*, *MIME* (the multi-media 822 mail format), and *richtext* (a simple page description language). For more information, send a message to

```
st-subscriptions@simple-times.org
```

with a `Subject` line of

```
help
```

In addition, ***The Simple Times*** has numerous hard-copy distribution outlets. Contact your favorite SNMP vendor and see if they carry it. If not, contact the publisher and ask for a list. (Communications via e-mail or fax are preferred).