The Simple Times™

THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS℠
VOLUME 3, NUMBER 1 FEBRUARY, 1994

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 12.

In this Issue:

Technical Article

Aiko Pras and Jacques Togtema
Twente University of Technology

In this issue: *SNMPv2 at Twente University*

The management group at Twente University in the Netherlands is currently developing SNMPv2 software. The purpose of this article is to provide an overview of this development and give future plans. It is not the intention to go into to much detail — the last section of this article tells how to obtain more detailed information.

Background

The last couple of years the network management group of our university has participated in a number of European RACE and Esprit projects. We worked on management architectures and, as most Europeans, concentrated on OSI and TMN.

After some time however it became clear to us that we needed more experience to judge the merits of the various architectural concepts. This resulted in a decision to start our own implementation project. Given the growing importance of Internet management and the different "atmosphere" of the Internet world, we decided to start implementing the complete SNMPv2 Framework (RFCs 1441–1452). Since we weren't involved in the definition of SNMPv2, our work would reveal whether SNMPv2 is sufficiently defined to allow implementation by non-adepts.

Goals

Our first goal was to learn through experience. To achieve this goal, we decided to discuss and implement all aspects of SNMPv2 ourselves. For example, we implemented the Basic Encoding Rules from scratch, although we could have started from one of the existing SNMPv1 packages.

Our second goal is to share our ideas with the community. We therefore make our software freely available and invite others to comment.

It should be noted that we're not working on a commercial product. If, for example, we have to choose between clarity and performance, we choose clarity.

Features

The two main differences between our implementation and those of others (e.g., CMU and 4BSD/ISODE), is our multi-process structure and our high-level programming interface (API). We will discuss each of them in a separate section.

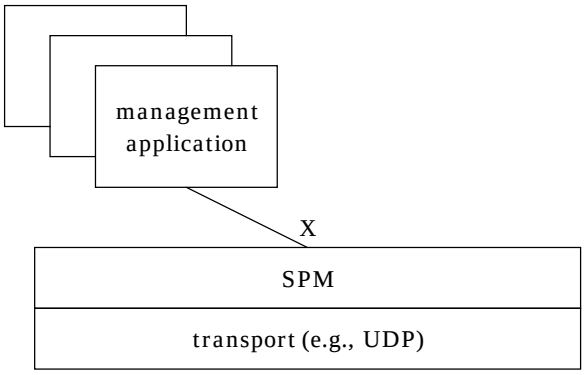
Multi-process Structure

Our software development takes place on SUN Sparcstations running UNIX (SunOS 4.1 and Solaris 2) using the GNU *C* compiler. Although we can't really test our software on other UNIX systems, we try to keep our software portable.

As UNIX allows multiple processes to cooperate, we decided to use this facility and develop a structure in which a single process, called the SNMPv2 Protocol Machine (SPM), serves multiple management applications.

The SPM is responsible for the transfer of SNMPv2 management information between systems and has no knowledge of management application issues. The SPM is therefore not bothered with MIB issues and things like Textual Conventions. The attractive property of our structure is that the SPM can be the same for the manager and agent side: it is the management application that determines whether a system acts in a manager or agent role.

Modification of this role is easy, since other applications (playing different roles) can be connected to the same SPM. Development of dual-role intermediate-level managers (e.g., to support the Manager-to-Manager MIB), is therefore straightforward.



Communication between SPM and management applications uses interprocess communication, such as TLI and sockets. As such, it is even possible to run management applications on different machines. Of course there is a performance penalty in having this multi-process structure with IPC. We believe however that this performance penalty is sufficiently compensated by the improved flexibility and the lower complexity.

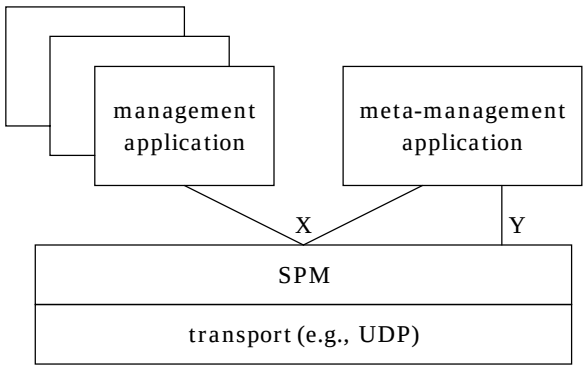
As the first management application becomes active, the SPM is automatically started. The SPM initializes by reading a configuration file and is then ready to serve the application. If other management applications become active, they will be connected to the same SPM. The SPM remains active until all management applications have terminated. Upon termination, the SPM saves a copy of the recent configuration information to disk.

The SPM is implemented in a single-threaded fashion, although we originally considered a multi-threaded approach. The advantage of a multi-threaded approach is that messages can be processed in different threads independent from each other. It guarantees that large messages that require authentication and encryption will not delay the processing of small messages that do not have security requirements. However, after an investigation of DES and MD5 processing times and after we understood the complexity associated with multi-threaded design, we decided to use a single-threaded approach.

Implications

The decision to have a multi-process structure and to separate management transfer functions from management application functions, has important implications. Let's discuss two of them.

The first implication is that management of SNMPv2 itself (meta-management) should not be performed by the SPM, but by special (meta-management) applications. As a consequence, the SPM is not bothered with issues such as maintaining the Party, SNMPv2 and Manager-to-Manager MIBs. This is equivalent to the approach followed with other protocols, such as IP and TCP. Implementors of IP and TCP are not bothered with MIB issues either; it is sufficient if they provide a local interface that allows reading and writing of IP and TCP variables by a special management process. It is the task of this special process to transform local information into the form required by the MIB-II and check the validity of the management operations.



To illustrate the functioning of our meta-management application (MMA), consider the example of changing a party's authentication clock. To change this clock, a `set` PDU must be received by the MMA via the same IPC port (e.g., X) as used for other application exchanges. The MMA (and not the SPM!) implements the rules (as specified by the Party-MIB) for changing clocks: the MMA therefore checks whether the received PDU also changes the authentication key. If this is the case, the MMA uses the local interface port (e.g., Y) to perform the actual clock change.

A second implication is that proxy relationships will not be performed by the SPM, but by special proxy applications. This makes the design of proxy agents straightforward: it is sufficient to understand the management API, it is not necessary to know the details of SNMPv2.

Application Programming Interface

The main difference between our implementation and other implementations (e.g., CMU and 4BSD/ISODE) is the programming interface. As opposed to other implementations, our API hides most of the complexity of SNMPv2 from the management applications. Writers of management applications need little knowledge of SNMPv2, which makes development of applications easier.

Our API is actually a library of C-functions, which must be included in every management application. Details of the IPC mechanism are handled within the API and are therefore not visible to the writer of management applications.

The API functions can be divided into two categories: functions necessary for initialization and termination purposes and functions necessary for sending and receiving SNMPv2 PDUs. To initialize, the application calls the `snmpOpen` function. This function starts, if necessary, the SPM and connects the application to it. To terminate, the application calls the `snmpClose` function.

This function removes the connection with the SPM — if no other applications are connected to the SPM, the SPM will terminate too.

Most of the API functions are used for sending and receiving SNMPv2 PDUs. Most "service elements" (e.g., `get`) require four function calls: two for sending (Request and Response), and two for receiving (Indication and Confirm). The Response and Confirm function calls are necessary for sending and receiving Response PDUs. Although it is possible to use the same Response and Confirm functions for all 'service elements', we decided (primarily for clarity reasons) to introduce separate function calls for each individual "service element".

Since the application can not know in advance to which "service element" a received PDU will belong, the application precedes each Indication and Confirm call with a `snmpLook` call. This call tells the application the type of service element that has been put into the IPC queue by the SPM and must therefore be handled first.

The table below shows all API calls that can be used to send and receive SNMPv2 PDUs. Note that because the SPM translates, at the agent's side, a `get-bulk` into a number of `get-next` calls, there are no Indication and Response calls for `get-bulk`. The table also shows that the SPM, upon receipt of an `inform` PDU from another SPM, automatically generates the `response` PDU.

	Req	Ind	Res	Con
get	x	x	x	x
get-next	x	x	x	x
get-bulk	x			x
set	x	x	x	x
inform	x	x		x
trap	x	x		

Of course, associated with each function call are a number of parameters, including:

- the IP address of remote system;
- the list of variable bindings;
- the request ID;
- the requested security, i.e., "none", "auth" (MD5 is used), or "priv" (MD5 and DES is used); and,
- a string that helps to determine the context, which may be empty.

Status

The first version of our software was released in November 1993. The major part of this software was written by a single student as part of his M.Sc. thesis. This first release should be considered as a "statement of direction" — although it can be used to manage SNMPv2 systems, it is not yet complete, e.g., there are no meta-management applications, so it is not yet possible to modify the Party, SNMPv2 and Manager-to-Manager MIBs.

Experience

At the beginning of our project we assumed SNMPv2 would be simple and easy to implement. This assumption proved to be wrong. SNMPv2 is complex for the following reasons:

The demands put on SNMPv2 are much stronger than those put on SNMPv1. This inherent complexity appears for instance from the fact that 12 RFCs were needed to specify SNMPv2.

SNMPv2 has been specified as a monolithic whole — little attempt has been made to decompose SNMPv2 into a number of building blocks with clearly defined interfaces. We believe that a decomposition of SNMPv2 into transfer, application, meta-management and proxy-oriented parts (along the lines of our approach) would have made SNMPv2 easier to understand.

The RFCs describe various mechanisms into great depth, but they hardly explain their purpose and the way they should be used (this is particularly true for SNMPv2's administrative model). It would for instance be helpful to see a description of how to search through the party, context, access control, and view table to determine the party and context identities that must be included with the PDU.

Still no real problems were encountered. Interoperability testing against the CMU package went smoothly. We may therefore conclude that it is possible for non-adepts to implement SNMPv2.

Future

After our first release demonstrated that it was possible to develop SNMPv2 software, we formed a project team to continue this development for at least another year. Our new team is much bigger than the one we had last year, so we expect to complete all aspects of the SNMPv2 framework, including meta-management, this year. Our plan is to make new versions available on a regular basis (e.g., every three months) Our sources can be obtained via anonymous FTP from `ftp.cs.utwente.nl` in the directory `pub/src/snmp`. Our email address is: `snmp@cs.utwente.nl`. Further information, including all project documentation, is made available via our WWW server:

`http://snmp.cs.utwente.nl:8001/snmp/html/homepage.html`

Industry Comment

Marshall T. Rose

Welcome to the third year of ***The Simple Times***.

With this issue, we're moving to a quarterly distribution cycle. The reason is simple: the coordinating editor simply doesn't have enough time to put together six issues each year. So, we're going to try four issues a year.

As a consequence of this, the *Working Group Synopsis* column is discontinued. Although Fred Baker, Deidre Kostick, and Kaj Tesink have done a wonderful job with each issue, the column will lose too much of its value with a longer distribution cycle. Fortunately, each of these contributors has promised to write a technical article later on for ***The Simple Times***!

Applications and Directions

Steven L. Waldbusser

In this issue: *Deploying SNMPv2*

As people have been tracking the progress of SNMPv2 deployment they have wondered about where the bottlenecks in the deployment process are, what is driving the process, and where products will materialize most quickly. Of course, this is a chicken and egg problem, so the answer to the question "Will the deployment be led by managers or agents?" is largely "Yes!". In an attempt to be a bit more accurate than this, we will compare the current scenario with the deployment of SNMPv1, and then make some observations about the current situation and predictions of the future.

SNMP (version 1) Deployment

In 1988, as the first SNMP specifications were finished, there were two interesting things about the industry: there were very few network management applications or protocols available, and much of the networking infrastructure (gateways and servers) was built on UNIX platforms. The first point created an incredible vacuum that was filled by SNMP. The second meant that a few free and commercial UNIX products easily built a critical mass of deployed products. This was followed fairly quickly by many embedded (i.e., non-UNIX) agents as vendors realized it was quite inexpensive to add an agent to a product and provide another check-off item for customers who had been loudly demanding network management capabilities.

Applications and platforms of varying sophistication came later, and when vendors realized the complexity of building a platform, many scaled back to only providing

applications that managed their products. This author hastens to add that the complexity was not due to SNMP, but due to the difficulties of writing X-based GUIs. There was a great sucking sound as the hierarchical map gobbled up development resources (for scant value to the customer, but that a subject for another time).

SNMPv2

When we look at today's environment, two differences are immediately apparent. The incredible vacuum that existed before is not present because SNMPv1 is handling customers' most pressing problems. Of SNMPv2's many benefits, the three that will drive customer demand are configuration capability, security, and speed. In addition, UNIX implementations are not enough to build critical mass. (This author frequently ponders whether a free DOS/Windows SNMPv2 Host MIB implementation would be an entirely different matter.)

These issues explain why SNMPv2 has not been deployed as quickly as SNMP1. Despite this environment, some vendors have already shipped SNMPv2, each of the major platform vendors have committed to delivering SNMPv2 support this year, and many of the major equipment vendors have made the same commitment. So what will lead the pack?

SNMPv2 Managers vs. Agents

While SNMPv2 agents are simpler to implement in general, this is especially true when one looks at SNMP security. If one is using a commercial or free SNMPv2 implementation to build their agent, SNMPv2 puts no additional burden on the agent except perhaps for providing long-term storage for security parameters. On the other hand, an SNMPv2 platform vendor needs to provide a security administration system and user interfaces for configuring security information. This is the bulk of the work in adding SNMPv2 to an existing SNMPv1 product and thus the major barrier to deployment of managers. While both managers and agents achieve the check-off by implementing SNMPv2, agents certainly get more bang for the buck. There is reason to believe that many agent vendors already know this and are acting accordingly.

Todays networks are built with many closed systems that are managed by open systems. A customer can add SNMPv2 management applications by using one of the shipping commercial products or one of the free implementations, but there is no way for a customer to add SNMPv2 to a router, bridge, or hub. This makes it more important for the vendors of managed systems to act. Of course, the advantage to them is that they are not as dependent on the MS vendors as the MS vendors are dependent on the agents. Customers can make some use of configuration capability, security, and speed in SNMPv2 agents by using the available managers, independently of the availability of SNMPv2 for the popular platforms.

Customer Demands

Customers today may be muted in their requests for SNMPv2 because things are not so bad with SNMPv1, and because they have to push on a lot of vendors to get SNMPv2 deployed on their networks. However, once a customer has a few SNMPv2 agents deployed, it will be clear to the customer which vendor is keeping him from SNMPv2's enhancements with his favorite platform. At this point, the customer will become much more aggressive in demanding SNMPv2 from that vendor.

For these reasons, the chicken-and-egg situation will be broken first by agents, followed closely by applications and platforms. This will all be playing out this year as vendors make good on their commitments.

Ask Dr. SNMP

Jeffrey D. Case

Dear *Dr. SNMP*,
We are working with some agent software that sends back replies on a port other than 161. For example, a `get` is sent from port 3000 to port 161, and the agent sends its `response` from port 4000 back to port 3000. This is disconcerting to us because we can't decode it with a protocol analyzer.

I have looked at the relevant RFCs (SNMP and UDP). The SNMP RFC states only that "the agent receives requests on port 161" but doesn't implicitly state that it should reply to them on this port. So, is this behavior legal?

—*Bummed out Believer from Beaverton*

Dear *Bummed out Believer from Beaverton*,
Down on the farm, we have a saying:

> "You'd be right, but you'd be dead."

When I first learned to drive an automobile, I was taught that in the jurisdiction where I lived, the pedestrian **always** has the right-of-way. Consequently, if a person is standing beside the road, and a large, tri-axle dump truck is hurtling down the hill toting a 20 ton load of gravel, and the person leaps in front of the truck, then the truck **must** stop, because the person on foot has the right-of-way. The fact that this violates physics is irrelevant. Of course, when the truck runs over the pedestrian, while it may be good for the pedestrian's

estate, this has rather drastic consequences for the pedestrian.

You might ask, "What does this have to do with my problem?" There are many things in SNMP which are "legal" but which have rather drastic consequences. By my reading of the SNMP protocol specification, as articulated in RFC 1157, it says:

> A protocol entity receives messages at UDP port 161 on the host with which it is associated for all messages except for those which report traps (i.e., all messages except those which contain the Trap-PDU).

It is noteworthy that it does not say "agent"! It says "protocol entity", and elsewhere in the specification, protocol entity is used when referring to agents and managers. Therefore, the specification appears to require all protocol entities, i.e., both managers and agents, to send and receive request/response messages at port 161.

I can tell from your question that you are well aware (but your supplier may not be aware) that existing practice for the request-response PDUs is as follows:

- A manager station prepares a request message containing an appropriate query or command, with a particular request ID, and a particular community string.
- The manager grabs a port, typically any unused port greater than 1024 (because access to low valued ports are usually restricted to applications with special privilege on systems with real operating systems) and sends the request message from one of the manager's addresses and this port number to one of the agent's addresses and UDP port 161.
- Unless the request message is lost, corrupted, or too large to be received by the agent, then the agent receives the request, conducts a rudimentary parse of it, and extracts the community string.
- The community string is compared with the valid configured communities, typically by scanning a table, to determine if the request is deemed authentic. If it is determined that the request is not authentic, then an error counter is incremented, the request message is dropped (and possibly logged), and a notification is optionally (depending upon the setting of `snmpEnableAuthenTraps`) enqueued to the configured managers, indicating that an authentication failure has been detected.
- If the message is determined to be authentic, then the agent further parses the message, and if successful, processes the message to prepare a suitable reply or a suitable error message. The resulting message is prepared with the same request ID and the same community string as the original request.
- The response is then sent from the agent at the address and port where the request was received to the manager at the address and port which originated the request. This is where there is a mismatch between existing practice and the operations of the system you ask about. In addition, while the SNMP protocol specification is silent about this, the Host Requirements RFC (RFC 1122, page 32) requires an additional check to insure that the reply is being sent to a valid address, i.e., that the source address of the original request was not invalid.

Given this background, it is easy to see that the law of the land may not match perfectly with existing practice, which is directly analogous to the notion that that the lawyers involved would debate whether, in fact, the truck really is or is not required to stop.

You ask if the unit's operation is "legal". Dr. SNMP is an engineer, not a jurist, and is unable to state an opinion on your question at this time. It is regrettable that there are several issues similar to the one you cite in the SNMPv1 protocol specification. Use of `OBJECT IDENTIFIER` fragments which are too large to fit in an unsigned longword and use of `INTEGERs` which too large to fit in a signed longword are but two examples, both of which are addressed in the new SNMPv2 specifications, but we must bear with the current undesirable situation until SNMPv2 is more widely deployed. Use of these questionable practices may be "legal", just as jumping in front of the truck may be legal. However, few would debate whether thrusting oneself in front of a truck in order to create a *roadkill-in-motion* situation is a good idea, and most would agree that deploying systems that don't conform to the existing practice is a similarly bad idea. I think the better question is, "Is it better to be right, or to be alive?" and to avoid being roadkill.

Security and Protocols

Keith McCloghrie

Since the original publication of MIB-I (RFC 1066), in August 1988, several of its MIB groups have needed to be upgraded. The latest such upgrade is for the interfaces group, with the recent publication of RFC 1573 as a Proposed Internet Standard. In this article, we'll look at the needs for, and results of this evolution of the interfaces group.

Dynamic interfaces

The interfaces group defines a generic set of managed objects such that any network interface can be managed in an interface-independent manner through these managed objects. Each interface is represented by a row in the `ifTable` and identified by a unique value of the `ifIndex` object. The description of `ifIndex` not only constrains its value to start at one and be less or equal to the number of interfaces, but also requires that the value for a particular interface must not change except at a restart of the managed agent. This represents a problem for agents which can have a network interface dynamically removed (e.g., when an interface other than the one with the highest `ifIndex` value is unconfigured). The evolution solves this by allowing the value of `ifIndex` to be greater than the current number of interfaces, although it still recommends that `ifIndex` values be assigned contiguously starting at one.

Interface Sub-layers

The original model of the interfaces group was that each network interface represented a complete interface stack which spanned from immediately underneath the internet-layer down to the physical layer. Media-specific information is defined in separate MIBs as an extension of the generic information contained in the `ifTable`. This works fine for interfaces such as ethernet. However, experience with interfaces having multiple sub-layers beneath the internetwork-layer has shown the need to define management information for each sub-layer, For example, an interface with PPP running over an HDLC link which uses a RS232-like connector: each of these sub-layers has its own media-specific MIB module, and thus, needs its own row in the `ifTable`. Thus, the evolution allows each sub-layer to be defined as a network interface, and also defines a new table, the `ifStackTable`, to represent how such interfaces are layered.

Consistent with this new approach is the clarification that there is no requirement that an internet-layer protocol runs at some layer above a network interface. For example, in MAC-layer bridges, for those `ifTable` counters of packets/octets received on an interface and "delivered to a higher-layer protocol", the bridge's forwarding module is considered to be a "higher-layer" to the MAC-layer of each port on the bridge.

In order to prevent a surfeit of `linkUp` and `linkDown` traps, one for each sub-layer, with this new multiple sub-layer model, an object has been added to enable/disable these trap types. By default, the lowest sub-layer of each interface stack has its traps enables, and each higher sub-layer has its traps disabled.

Bit and Character-oriented Interfaces

RS-232 is an example of a character-oriented sub-layer over which (e.g., through use of PPP) IP datagrams can be sent. Since many of the objects in the `ifTable` are defined in terms of packets, it is not possible to have a character-oriented sub-layer represented by a (whole) row in the ifTable. Even fewer of the objects apply to a bit-oriented interface, such as a DS1 link. A further complication is that some subnetwork technologies transmit data in fixed length transmission units. One example of such a technology is cell relay, and in particular Asynchronous Transfer Mode (ATM), which transmits data in fixed-length cells. Representing such an interface as a packet-based interface produces redundant objects.

To address this issue, the evolution makes use of the SNMPv2 capability to define overlapping MIB groups for conformance purposes. It defines an `ifGeneralGroup` which applies to all network interfaces, an `ifPacketGroup` group containing those objects applicable to all packet-based interfaces, and an `ifFixedLengthGroup` containing objects applicable to interfaces which transmit data in fixed-length transmission unit, including character-oriented interfaces. It then specifies conformance statements requiring each interface to implement the ifGeneralGroup and whichever other group is applicable.

Counter Size

Since 1988, the bandwidth of network media has increased significantly. For higher-speed interfaces, the original 32 bit counters (e.g., for octet/packets) wrap around in a shorter amount of time. For example, ifInOctets can wrap in as little as 5.7 minutes for FDDI, and 34 seconds for a 1-gigabit medium! Thus, polling the interface statistics frequently enough not to miss a counter wrap is becoming increasingly problematic.

To address this issue, the evolution was granted a waiver to use the new SNMPv2 64-bit counter type for use with higher speed interfaces. 64-bit octet

counters are specified for interfaces greater than 20Mb/s. 64-bit packet counters are specified for interfaces greater than 650Mb/s. These new counters supplement, not replace, the existing 32-bit counters in order to foster compatibility with existing implementations.

ifOperStatus

Two new states have been added to `ifOperStatus`: "dormant" and "unknown". The dormant state indicates that an interface is not actually ready to pass packets, but rather in a "pending" state, waiting for some external event. An example of where this state will be valuable is for connection-oriented interfaces where specific connections are taken down after a period during which there is no traffic.

ifType

A new textual convention, `IANAifType`, has been defined for the enumerated values of `ifType`. This allows new ifType values to be assigned by the Internet Assigned Number Authority (IANA). The new MIB includes an initial version of `IANAifType`, which will be updated and periodically re-issued by the IANA.

Other Changes

Several other minor changes have also been made, including:

- a new object to report interfaces speeds greater than 2.2Gb/s;
- new objects to count the number of multicast packets and the number of broadcast packets separately;
- the objects, `ifSpecific` and `ifOutQLen`, have been deprecated; because neither of them has proved to have sufficient usefulness to management applications; and,
- updated definitions of the `ifTestTable` and `ifRcvAddressTable`, originally defined in RFC 1229, have been specified.

Standards

David T. Perkins

Since the last issue, there have been five new SNMP-related standards published. These include an updated and extended version of the interfaces group from MIB-II, three MIBs for managing applications, and an update of the DECnet Phase IV MIB as it proceeded to Draft Standard status.

Recently Published RFCs

1559 - DECnet Phase IV MIB (Draft Standard)
This MIB module (in SNMPv1 SMI format) is an updated version of RFC 1289. There were general cleanups to fix typos, and to bring the MIB in conformance with the general editorial style. A new version of the adjacency table was created. A few redundant object types were removed, and the status of seven of the groups was changed from mandatory to optional.

RFC 1565 - Network Services Monitoring MIB (Proposed Standard)
This MIB module (in SNMPv2 SMI format) presents a general framework to monitor network services such as mail transfer agents, or directory service agents. The MIB module is designed to complement the Host Resources MIB (RFC 1514) and also to be application and transport protocol neutral. There are two tables in this MIB module: the first contains basic information about each network service in a system, and the second contains a limited number of attributes about each active association for each network service.

RFC 1566 - Message Transfer Agent (MTA) MIB (Proposed Standard)
This MIB module (in SNMPv2 SMI format) is the first example of a class of network service that may be monitoried. This MIB module applies to generic message relays. The first table measures aggregate incoming and outgoing mail traffic. The second table measures in greater detail the traffic including associations and errors for "groups". The final table is used to show which "associations" belong to each "group".

RFC 1567 - Directory (X.500) System Agent (DSA) MIB (Proposed Standard)
This MIB module (in SNMPv2 SMI format) is for another type of network service, X.500 DSAs. The first table has counters for each incoming operation, each outgoing operation, and errors observed. The next table reports cache performance. The final table tracks interactions with peer DSAs.

RFC 1573 - Interfaces Group MIB (Proposed Standard)
The interfaces group from MIB-II (RFC 1213) was extracted and expanded to create this MIB module (in SNMPv2 SMI format). This 55 page document is the evolution of a key component of the core IETF SNMP model of managed systems. The first 20 pages address general and specific issues with the original interfaces group. The remainder of the document consists of two MIB modules. The first contains the definition of the textual convention used to define each interface type.

This organization allows easy update by the Internet Assigned Numbers Authority (IANA). The second MIB module contains five tables. The first is the original IF table from MIB-II. The second table extends the IF table with new object types, replacements, and ones from the `ifExtnsTable` from RFC 1229. A "stack" table is next, which defines relationships among the sub-layers of an interface. The fourth table is used to perform specific tests on interfaces, replacing a similar table in RFC 1229). And the last table specifies the addresses which each interface may receive packets/frames.

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;
- 1493 - Bridge MIB; and,
- 1516 - IEEE 802.3 Repeater MIB; and,
- 1559 - DECnet phase IV MIB.

Proposed Standards:

- 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 (SMT 6.2) 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;
- 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;
- 1420 - SNMP over IPX;
- 1461 - Multiprotocol Interconnect over X.25 MIB;
- 1471 - PPP Link Control Protocol (LCP) MIB;
- 1472 - PPP Security Protocols MIB;
- 1473 - PPP IP Network Control Protocol MIB;
- 1474 - PPP Bridge Network Control Protocol MIB;
- 1512 - FDDI Interface Type (SMT 7.3) MIB;
- 1513 - Token Ring Extensions to RMON MIB;

- 1514 - Host Resources MIB;
- 1515 - IEEE 802.3 Medium Attachment Unit (MAU) MIB;
- 1525 - Source Routing Bridge MIB;
- 1565 - Network Services Monitoring MIB;
- 1566 - Mail Monitoring MIB;
- 1567 - X.500 Directory Monitoring MIB; and,
- 1573 - Evolution of the Interfaces Group of MIB-II.

Experimental:

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

Informational:

- 1215 - A convention for defining traps for use with the SNMP;
- 1270 - SNMP communication services;
- 1303 - A convention for describing SNMP-based agents;
- 1321 - MD5 message-digest algorithm;
- 1470 - A network management tool catalog; and,
- 1503 - Automating Administration in SNMPv2 Managers.

Historical:

- 1156 - Management Information Base (MIB-I);
- 1161 - SNMP over OSI;
- 1227 - SNMP MUX protocol and MIB;
- 1229 - Extensions to the generic-interface MIB;
- 1230 - IEEE 802.4 Token Bus Interface Type MIB;
- 1232 - DS1 Interface Type MIB;
- 1233 - DS3 Interface Type MIB;
- 1252 - OSPF version 2 MIB;
- 1283 - SNMP over OSI;
- 1284 - Ether-Like Interface Type;
- 1286 - Bridge MIB;
- 1289 - DECnet phase IV MIB;
- 1298 - SNMP over IPX;
- 1351 - SNMP Administrative Model;
- 1352 - SNMP Security Protocols;
- 1353 - SNMP Party MIB; and,
- 1368 - IEEE 802.3 Repeater MIB.

Subscribing to SNMP-related Working Groups

Appletalk/IP Working Group:

- `apple-ip-request@cayman.com`

AToM MIB Working Group:

- `atommib-request@thumper.bellcore.com`

BGP Working Group:

- `iwg-request@ans.net`

Bridge MIB Working Group:

- `bridge-mib-request@nsl.dec.com`

Character MIB Working Group:

- `char-mib-request@decwrl.dec.com`

DECnet Phase IV MIB Working Group:

- `phiv-mib-request@jove.pa.dec.com`

FDDI MIB Working Group:

- `fddi-mib-request@cs.utk.edu`

Frame Relay Service MIB Working Group:

- `frftc-request@nsco.network.com`

Host Resources MIB Working Group:

- `hostmib-request@andrew.cmu.edu`

IEEE 802.3 Hub MIB Working Group:

- `hubmib-request@synoptics.com`

IDPR Working Group:

- `idpr-wg-request@bbn.com`

IDRP for IP Working Group:

- `idrp-for-ip-request@merit.edu`

Interfaces MIB Working Group:

- `if-mib-request@thumper.bellcore.com`

IPLPDN Working Group:

- `iplpdn-request@nri.reston.va.us`

IS-IS Working Group:

- `isis-request@merit.edu`

Mail and Directory Management Working Group:

- `ietf-madman-request@innosoft.com`

Modem Management Working Group:

- `modemmgt-request@telebit.com`

NOCtools Working Group:

- `noctools-request@merit.edu`

OSPF Working Group:

- `ospfigp-request@gated.cornell.edu`

PPP Working Group:

- `ietf-ppp-request@ucdavis.edu`

RIP Working Group:

- `ietf-rip-request@xylogics.com`

Remote Monitoring (RMON) Working Group:

- `rmonmib-request@jarthur.claremont.edu`

SNA DLC Services MIB Working Group:

- `snadlcmib-request@apertus.com`

SNA NAU Services MIB Working Group:

- `snanaumib-request@thumper.bellcore.com`

SNMPv2 Working Group:

- `snmp2-request@thumper.bellcore.com`

TCP Client Identity Protocol:

- `ident-request@nri.reston.va.us`

Trunk MIB Working Group:

- `trunk-mib-request@saffron.acc.com`

Uninterruptible Power Supply Working Group:

- `ups-mib-request@cs.utk.edu`

X.25 MIB Working Group:

- `x25mib-request@dg-rtp.dg.com`

Activities Calendar

- 29th Meeting of the IETF
  March 28–April 1, Seattle, WA
  For information: +1 703 620 8990
- N+Interop Conference and Exhibition
  May 2–6, Las Vegas, NV
  For information: +1 415 578 6900

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
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).