The Simple Times™

THE BI-MONTHLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS℠
VOLUME 2, NUMBER 1 JANUARY/FEBRUARY, 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:

Technology and Commentary
Technical Article 1
Industry Comment 5

Featured Columns
Applications and Directions 6
Ask Dr. SNMP 7
Security and Protocols 8
Standards 9
Working Group Synopses 11

Miscellany
Activities Calendar 19

Publication Information **20**

Technical Article

Samuel M. Roberts, Farallon Computing, Inc.

In this issue: *An Introduction to SNMP MIB Compilers*

An SNMP MIB compiler is an extremely useful tool, not only for authors of SNMP MIBs, but also for implementors of SNMP agents and users of SNMP management applications. In addition to verifying the syntactic correctness of a MIB, a MIB compiler can automatically generate data structures and code required by an agent to implement a particular MIB. A MIB compiler can also make information about managed objects in proprietary vendor MIBs or new Internet-standard MIBs available to a management application.

This article provides an introduction to SNMP MIB compilers. It begins with a description of the function that a MIB compiler performs. It then examines the internal structure of a typical MIB compiler and provides an overview of some of the more common output formats produced by SNMP MIB compilers. Finally, the article concludes with a section describing a number of openly-available and commercial MIB compilers.

What is a MIB Compiler?

A compiler is a program that translates a program written in one language — the source language — into an equivalent program in another language — the target language. Typically, the source language is a high-level programming language, such as C or Pascal, while the target language is a low-level programming language, such as a particular target platform's assembly or machine language.

The source language for a MIB compiler is the language Abstract Syntax Notation One (ASN.1). ASN.1 is not a programming language; it is a language for describing structured information. ASN.1 resembles the data declaration portion of a high-level programming language. The input to a MIB compiler isn't a program, but rather a collection of MIB modules written in a subset of ASN.1. These MIB modules contain definitions of managed objects that correspond to information in network devices that can be manipulated via SNMP.

MIB compilers usually generate various alternate representations of the managed object definitions in the input MIBs. These alternate representations are easier for management applications and agents to process than the ASN.1 representation.

Some alternate representations are actually data structure declarations in a high-level programming language, such as C, that can be compiled and linked into a management application or agent. Others are data files containing "flat" representations of the managed object definitions that can be read into memory by a management application or agent at run-time.

In some cases, MIB compilers output code to assist in the implementation of the input MIBs. For example, a MIB compiler may generate skeleton routines for retrieving or setting the value of a managed object, or routines to generate particular SNMP `Trap-PDUs`.

Structure of a MIB Compiler

Despite the fact that a MIB compiler accepts input written in a data description language rather than a programming language, it is similar in structure to a programming language compiler. It typically consists of a front-end and one or more back-ends. The front-end reads input files containing ASCII text corresponding to ASN.1 MIB modules and constructs an internal representation of the information contained in the MIB modules. The back-end produces one or more output files from this internal representation.

The front- and back-ends of the MIB compiler may be totally independent applications that communicate by means of an intermediate file on disk, or they may be individual modules of a single application that communicate by means of shared data structures in memory.

Although most MIB compilers are implemented as stand-alone programs and generate output files on disk, a MIB compiler may sometimes be integrated directly into a network management application (or even an agent). In this case, the MIB compiler consists only of a front-end and no back-end, and the MIB compiler does not produce an output file. Instead, the management application makes direct use of the internal representation constructed by the front-end from the ASN.1 MIB modules.

The Front-end

The front-end of a typical MIB compiler consists of a *lexical* analyzer, a *syntactic* analyzer, and a *semantic* analyzer, together with support routines such as symbol table management and error handling routines. The lexical analyzer, or scanner, reads the sequence of ASCII characters in the input files and groups them together into a sequence of tokens. Tokens are the basic lexical units of ASN.1. There are keyword tokens, such as "`BEGIN`" and "`END`", punctuation tokens such as "`::=`" and "`..`", identifier tokens such as "`iso`", and numeric tokens such as "`255`".

The syntax analyzer, or parser, takes the tokens produced by the scanner and groups them together into syntactic structures according to the ASN.1 grammar. The token stream produced by the scanner must consist entirely of one or more instances of ASN.1's `ModuleDefinition` syntactic structure; otherwise the input contains syntax errors and the parser generates appropriate error messages. For example, the parser would recognize the following sequence of tokens

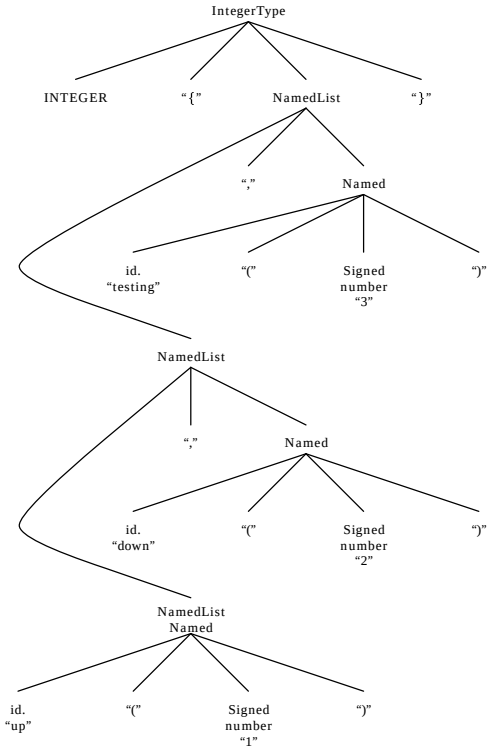
```
INTEGER {
    up(1),
    down(2),
    testing(3)
}
```

as a syntactically correct instance of the `IntegerType` syntactic structure using the following Backus-Naur Form (BNF) productions (or rules) from the ASN.1 grammar:

```
IntegerType ::=
      INTEGER
    | INTEGER "{" NamedList "}"

NamedList ::=
      Named
    | NamedList "," Named

Named ::=
      identifier "(" Signed ")"

Signed ::=
      number
    | "-" number
```

To see that the previous sequence of tokens is indeed an instance of the `IntegerType` syntactic structure, it is helpful to construct a diagram called a parse tree. A parse tree exhibits the syntactic structure of a particular sequence of tokens.

The parse tree for the previous example is as follows:



The semantic analyzer takes the output of the parser and checks it for semantic correctness. Semantic analysis deals with those aspects of ASN.1 that cannot be specified by means of a grammar, but instead depend on context. Even though a particular sequence of tokens may be syntactically correct according to the ASN.1 grammar, it may not be legal ASN.1. For example, the sequence of tokens

```
MyType ::= INTEGER
```

is always a syntactically correct instance of a `TypeAssignment`. However, depending on the context in which this `TypeAssignment` appears, it may not be semantically correct. For example, it would be semantically incorrect if the enclosing MIB module contained a prior assignment to the type identifier `MyType`.

The lexical, syntactic, and semantic analyzers each make use of various support routines, including error handling and symbol table management routines. Error handling routines generate error messages for the user when there is an error in the input to the compiler. Errors can be generated at any phase of the compilation. For example, the scanner will generate an error if it encounters a character that cannot be part of any ASN.1 token, such as an at-sign, or an improperly formed token, such as an identifier that ends with a hyphen. The syntax analyzer will generate an error if it encounters a sequence of tokens that isn't valid according to the ASN.1 grammar, such as a comma in the middle of an `OBJECT IDENTIFIER` value. The semantic analyzer will generate an error if it detects a semantic error, such as a conflict between the syntax specified for an object in a `SEQUENCE` definition and a later `OBJECT-TYPE` definition.

Symbol table management routines keep track of the names used in the MIB modules being compiled and record essential information about each name, such as its type (textual convention, managed object, trap, and so on) and the modules in which the name is defined.

Note that although conceptually it is useful to think about these front-end components as independent tasks, they do not necessarily operate sequentially. For example, the same routine that parses a particular syntactic construct often checks it for semantic correctness. In addition, the scanner is often implemented as a subroutine of the parser, which asks the scanner for the next token when the parser needs one.

Back-ends

Many MIB compilers contain multiple back-ends, each of which produces an output file containing a different representation of the same set of input MIBs. If the front- and back-ends of the MIB compiler are integrated in a single application, then one typically uses command line switches to select a particular output format. If the front- and back-ends are implemented as separate applications that communicate by means of an intermediate file, then one generates the desired output format by executing the appropriate back-end after generating the intermediate file using the front-end.

The output from some back-ends is specific to the implementation of agents, while the output from others is primarily suitable for use by management applications.

Back-ends that Aid in Agent Implementation

Before describing the output from back-ends designed to aid in the implementation of SNMP agents, it is useful to have a basic understanding of the structure of a typical SNMP agent. Most SNMP agent implementations separate SNMP protocol processing from MIB variable access. All knowledge of the SNMP protocol, including the formats of the various Protocol Data Units (PDUs), and the encoding and decoding of ASN.1 data, resides in an SNMP protocol engine. All knowledge of the managed objects in the MIB resides in a set of access functions commonly known as "method routines".

Typically, a C data structure representation of the MIB tree provides the glue between the protocol engine

and the method routines. When the protocol engine receives an incoming SNMP request, it extracts the OBJECT IDENTIFIER name of the object instance whose value is to be retrieved or changed. Using the OBJECT IDENTIFIER prefix that forms the name of the corresponding OBJECT-TYPE, the protocol engine consults the C data structure to find pointers to the method routines specific to that OBJECT-TYPE. The protocol engine invokes the appropriate method routine to perform the requested operation on the specified object instance.

Constructing the C data structure representation of a MIB tree by hand, even once, is tedious at best. Several MIB compilers provide back-ends that generate the required C data structure automatically from a set of input MIBs. (Some agent implementations do not link the MIB tree into the agent at build time. Instead, these implementations generate the tree structure at run-time from a data file containing a "flat" representation of the MIB tree. A MIB compiler is normally still used to generate this data file, albeit using a different back-end.)

Several MIB compilers contain back-ends that assist in the implementation of the method routines for an agent. One of these back-ends generates a C source file containing skeleton function definitions for each method routine. These skeleton functions are typically stub functions with the appropriate return type and argument list. The agent writer fills in the code required to get or set the appropriate MIB variable. A clever compiler may even be able to fill in some or all of the code required to implement a particular method routine.

Another back-end produces a C header (.h) file containing ANSI C and non-ANSI C function prototypes for the method routines. This header file is included by the file containing the MIB tree data structure and the file containing the definitions of the method routines.

At least one MIB compiler contains a back-end that produces a C source file containing functions that an agent can invoke to send SNMP Trap-PDUs corresponding to the TRAP-TYPE definitions contained in the input MIBs. These functions include code to create variable bindings in the Trap-PDU for the variables specified in the VARIABLES clause of the TRAP-TYPE definition.

Back-ends for Use with Management Applications

Most MIB compilers include one or more back-ends that generate output files designed for use by management applications. One output file format common to a number of MIB compilers is known as mosy format after the name of a popular MIB compiler, mosy. A mosy format file includes a flat representation of the MIB tree that management applications can use to map between descriptors and the corresponding OBJECT IDENTIFIERS. It also specifies each object's syntax, access, and status. The following example is an excerpt from a mosy format file generated from RFC 1213 (MIB-II):

```
mib-2    mgmt.1
system   mib-2.1
...
sysDescr    system.1 OctetString read-only mandatory
sysObjectID system.2 ObjectID    read-only mandatory
...
```

Several compilers include back-ends that output extended versions of mosy format. These formats preserve more of the information contained in the input MIBs, such as indexing information for objects in conceptual rows and subtype information for objects with range and size constraints. They also include additional information for objects with enumerated integer syntax to enable management stations to map integer values to the corresponding identifiers.

MIB Compiler Availability

This section describes those MIB compilers which are the most popular and with with the author is most familiar. Mention of a particular compiler does not imply its endorsement, either by the author or by ***The Simple Times***.

There are at least two openly-available stand-alone MIB compilers. The first is named mosy, which is an acronym for *Managed Object Syntax-compiler (YACC-based)*. This compiler is a part of the ISO Development Environment (ISODE), a package for developing OSI protocols and applications. It is available by anonymous FTP from ftp.uu.net in the directory /networking/osi/isode.

The second, named SMIC (the *SNMP MIB Compiler*), generates output files in mosy and extended mosy format. It also produces a more complex SMIC-specific format meant for use with as yet unreleased back-ends. SMIC is available by anonymous FTP from ftp.synoptics.com in the directory /eng/mibcompiler. (Volume 1, Number 4 of ***The Simple Times*** contains an announcement for SMIC.)

In addition, Carnegie Mellon University (CMU) distributes an openly-available SNMP implementation. The CMU SNMP package does not contain a stand-alone MIB compiler. Instead, it provides a library that management applications can call at startup to compile MIB modules from disk into memory. Although the CMU SNMP package can also be used to implement an SNMP agent in a managed device, no stand-alone MIB compiler is available for generating the C data structure representation of the agent's MIB tree. The CMU SNMP package is available by anonymous

FTP from `lancaster.andrew.cmu.edu` in the directory `/pub/smp-dist`.

Both Epilogue Technology, Inc. and SNMP Research, Inc., market commercial SNMP implementations in source code format that developers can license to incorporate SNMP functionality into management stations or network devices. Each company licenses a MIB compiler designed to work with its SNMP implementation.

The Epilogue Technology MIB compiler includes a variety of back-ends that produce MIB representations suitable for both agents and management stations. Other back-ends generate C source code that can assist in the implementation of method routines and the generation of SNMP `Trap-PDUs`.

The SNMP Research MIB compiler consists of two independent programs. The first is an enhanced version of the openly-available `mosy` MIB compiler described above. Its back-end produces an enhanced `mosy` format output file. The second program processes the enhanced `mosy` output and generates a variety of output files suitable for implementing both management stations and agents.

In addition, many commercial network management applications include a MIB compiler for importing information about managed objects, particularly those in vendor MIBs. For example, SunConnect's UNIX-based network management station, SunNet Manager, includes a stand-alone MIB compiler named `MIB2Schema` that translates MIBs into a proprietary format data file (called a schema file) that is read by the SunNet Manager application. Similarly, Hewlett-Packard's OpenView Network Node Manager application contains an integrated MIB compiler that can be used to import information about managed objects directly from MIB modules.

Choosing a MIB Compiler

The intended use of a MIB compiler may affect the criteria used in choosing one. For example, if the goal is to compile MIBs for use with an agent or management application, then it's desirable to use a MIB compiler that can successfully parse MIBs containing some minor ASN.1 syntax errors. Otherwise, a fair bit of time may be spent correcting syntax errors in MIBs, since even some MIBs published as RFCs contain syntax errors.

The `mosy` MIB compiler, and some compilers derived from `mosy`, perform less strict syntactic and semantic checking than other compilers, and as a result they will successfully compile MIBs with some syntax errors. Other compilers perform stricter syntactic and semantic checking, and provide command line switches that invoke less stringent checking.

On the other hand, if you are writing a proprietary vendor MIB, or a new Internet-standard MIB, you probably want a MIB compiler with very strict checking that will catch any errors you inadvertently introduce. SMIC and the Epilogue Technology MIB compiler are particularly good for this purpose.

Finally, most of the MIB compilers mentioned in the previous section are available in source code format, with the exception of those compilers that come with commercial network management applications. As a result, many of the compilers can be modified to generate output in formats they don't already support. Although a compiler designed to generate output for one SNMP implementation can be modified to generate output files for a different one, it is usually best to use the MIB compiler that was designed for use with a particular SNMP implementation or management application.

Industry Comment

Marshall T. Rose

This issue marks the beginning of the second year for ***The Simple Times***.

Sadly, we begin the year with the departure of one of our featured columnists, Robert L. Stewart of Xyplex, Inc. Bob's column, *Working Group Synopses*, is by far the most demanding column in ***The Simple Times***. During our first year, Bob was tireless in reporting and analyzing the goings-on of the two dozen SNMP-related working groups in the IETF. In addition, Bob has also been providing adult supervision as chair of the SNMPv2 working group. As with all of the people who volunteer their time in the Internet community, Bob also has a real life, and his column was proving too much of time-sink. Even though we'll miss Bob's contributions to ***The Simple Times***, he'll still be participating in the IETF process.

Taking over for Bob will be Frank J. Kastenholz of FTP Software, Inc. You probably remember Frank as the guy who stepped in to clean-up the Ether-like MIB mess of 1991. Frank was also one of the people who contributed to the POISED working group activity on adding accountability to the Internet standardization process. So, starting with this issue, the *Working Group Synopses* time-sink is on Frank's shoulders.

As usual, this issue ran over, so there won't be an industry commentary. But, in the next issue, I hope to have something on the "next" step in the evolution of network management.

Applications and Directions

Steven L. Waldbusser

In this issue: *Should user interface information be standardized?*

Many in the network management industry are recognizing that while great progress has been made in network management protocols and MIBs, the effectiveness of network management applications has not kept up. Some members of the standards community have responded to this situation with requests for new types of information to be added to the standard MIB format for the benefit of applications. On the surface, this information is undeniably useful. However, as one digs deeper, one finds that this data simply optimizes an old, tired paradigm, potentially at great cost to the standardization process. Fortunately, a different solution exists that provides this information without the drawbacks that would otherwise exist.

User Interface Information

This application-oriented information consists of additional information for each MIB object beyond those already used. In addition to the information such as type and description that currently exist, this application-oriented information consists of descriptive labels for use on a table or graph, formatting information, thresholds, help text, and so on. This information would be readable by the management station and would direct the management station to provide a better user interface to SNMP data it doesn't otherwise understand.

As an example, consider the information that might be provided about the RMON MIB `etherStatsCollisions` object:

```
etherStatsCollisions APPLICATION-INFO
        -- suitable for column header
    SHORT-LABEL  "Collisions"
    LONG-LABEL
        "Ethernet CSMA/CD Collisions"
    PRINT-FORMAT "decimal"

    USEFUL-STAT  "per etherStatsPackets"

        -- i.e., also useful per second
    USEFUL-STAT  "per sysUpTime * 100"

        -- absolute value isn't interesting
    THRESHOLD
        ".03 per etherStatsPackets"

        -- polling is useful
    VOLATILITY   volatile

    ICON-BIT-MAP
       "120A9847E48C92A001F28437B5900E12
        8A7712C890203D487565D6080C4E7478
        3B921C983A782C08314E78213C019C28
        3E774C182A001B2438A7565D6080437B"

    HELP-TEXT "us-english"
       "A collision is an event on an
        Ethernet network that is a part of
        everyday operation, but when
        excessive, can signal that the
        network is overloaded or is too
        long (especially when the average
        packet size is small). To avoid
        excessive collisions, use Token
        Ring."
```

How this Information is Useful

This data could be read by a management station and put to good use. Some of the information would control how the application displayed data retrieved with SNMP, while some of the information can be displayed to the user to help describe the associated SNMP data. Of course, even with this capability, this generic management station is still "dumb" — it doesn't have the capability to make recommendations to the user based on the data that is received via SNMP. Typical network managers need this intelligence, which can only be adequately provided by applications that are written specifically for a particular application or a particular MIB.

Given that vendor-specific MIBs outnumber standard MIBs by a factor of ten and are (often) less stable, it is difficult for an NMS vendor to support all vendor-specific MIBs effectively. For those vendor-specific MIBs that a network management system vendor cannot build tailored applications for, the additional information can be helpful. The question remains — how and where do we provide this information?

How not to distribute User Interface Information

It has been suggested that information such as this be added to the standard MIB format, i.e., by extending the `OBJECT-TYPE` macro. For several reasons, this is the wrong place to define this information. A MIB is a contract between agent writers and management station writers, drafted by a standards body or a vendor. A MIB describes data for the sole purpose of ensuring that both manager and agent developers are implementing with the same assumptions. (That is why the description text can be cryptic to users.) MIB authors in general, and standards bodies in particular, would be greatly burdened if they took on the additional responsibility

of designing user interfaces. A MIB working group has more pressing things to worry about than keeping labels short enough so that all the columns of a table fit on an 80 column display, for example. In addition, standard MIBs have an international audience. Clearly, it would be inappropriate for a standard to mandate an English-language user interface; similarly, it would be equally inappropriate for a technical working group to spend lots of time translating user interface information into 20 languages (and ensuring that the labels still fit on an 80 column screen)!

Application and user interface design remains an art and properly belongs in the hands of application designers, not MIB working groups. Evidence of this art is that most decisions that must be made in the interface design are based on aesthetic grounds, e.g., is it tasteful to take the vowels out of the labels to make them fit on an 80 column screen? The format of this data itself is not even stable. Very little experience has been accumulated about what information is worth storing and what format to store it in. It would not be appropriate to expose standard MIBs to such instability.

The Right Way

The correct way to provide this capability would be an additional macro, not unlike the example above, that is linked to the MIB object to which it refers. Files of `APPLICATION-INFO` macros could be provided by agent vendors, service providers, consultants, and staff engineers to refine and customize parts of the user interface of network management stations accessing particular MIBs. This would be most appropriate for vendor-specific MIBs which would not otherwise expect application support from all network management stations.

It is becoming more and more widely recognized that network management applications need to become more effective. While it is important to put more work into this, it is more important to put it into the right areas. User interface information doesn't belong in standard MIBs, but could be helpful if available in a standard form from other sources. However, keep in mind that while this is useful, it optimizes the wrong part of the problem. No great leap forward in the state of the art will arise from enhancing old paradigms. For that, there is no substitute for ingenuity, hard work, and running code.

Ask Dr. SNMP

Jeffrey D. Case

Dear *Dr. SNMP*,
Why can't you be more consistent? You insist on implementing everything before it can be accepted as a part of a new specification (like SNMPv2) but sometimes reject other people's ideas without having the benefit of implementation experience. How can you say an idea isn't a good one if you haven't implemented it?
— *Wheedler in Working Group*

Dear *Wheedler in Working Group*,
Down on the farm, we have a saying:

> "You don't need to lick a pump handle in February to know it's a bad idea."

Similarly, I have never (yet) failed to remember my wedding anniversary but I don't have to conduct an experiment to find out whether doing so would be a good idea. Experience has shown that sometimes there are bad ideas hiding behind a good idea that can be found through experimentation, but usually the only things hiding behind a bad idea are more bad ideas. *Dr. SNMP* is able to spend all of what otherwise would be his leisure time discovering that what looked like a good idea isn't, without spending time proving that a bad idea is.

Dear *Dr. SNMP*:
I see that the `readOnly` error status code is a part of the SNMPv2 drafts, having been retained for backward compatibility. Why is this there, given than SNMP version 1 compliant systems should never generate a readOnly status?
— *FAQ from Future*

Dear *FAQ from Future*,
The `readOnly` status code is the topic of some of the most frequently asked questions regarding SNMP version 1.
Down on the farm, we have a saying:

> "Nothing has more lives than an error that you refuse to correct."

One of the goals of the SNMP version 2 effort was to tighten the language of the Structure of Mangement Information (SMI), Management Information Base (MIB), and protocol (SNMP) specifications in order to eliminate ambiguities, avoid frequent errors, and to address frequently asked questions. I'm afraid that retaining this error code will allow us to generate a whole new set of frequently asked questions and answers, some of which may even be correct.

Security and Protocols

Keith McCloghrie

Responsibility for the definition of SNMP version 2 was split across two working groups of the IETF, each one using as their basis those parts of the *Simple Management Protocol (SMP) and Framework* specification relevant to their work. The SNMPv2 working group tackled the major changes, with the SNMP Security working group updating its previous work on the Administrative Framework and Security protocols. The SNMPv2 working group is done and now waits on the SNMP Security working group which, as of this writing, is not yet complete. In this issue, we'll look at the changes the SNMP Security working group has already agreed upon, and mention the issues requiring the other changes still being discussed. In the next issue, we expect to be able to report on the completion of these other changes.

Modified Digest Authentication Protocol

The changes already approved are those which were specified by the SMP. Of these, the major changes concern the *Digest Authentication Protocol*. From previous issues of this column, you will recall that the Digest Authentication Protocol uses the MD5 digest algorithm and loosely synchronized clocks. It is the mechanisms involved in using the synchronized clocks which are changed. One change is the simplification of the *Clock Synchronization algorithm*; the other is the elimination of the *Ordered Delivery mechanism*.

Recall that in SNMP Security, messages are originated by a *party* executing within one SNMP entity, and directed to a party executing in another SNMP entity. For each such party configured to use the Digest Authentication Protocol, there is an authentication clock. The *Message Timeliness mechanism* provides protection against the malicious replay of messages by including a timestamp of the sender's notion of the source party's clock in a message. When a message is received, the timestamp is added to a configured "lifetime" value, compared to the receiver's notion of the source party's clock, and the message is discarded as unauthentic if the timestamp is too old. For this to work, obviously the sender's and receiver's notions of the clocks must be loosely synchronized.

Simplification of Clock Synchronization

The simplification of the clock synchronization algorithm results from extending the *Selective Clock Acceleration mechanism* to apply to the authentication clocks of both the source and destination parties of a message. To facilitate this, the change specifies that the sender's notion of both the source party's clock and the destination party's clock are included in a message. With this extension, the mechanism causes the receiver's notion of each authentication clock in a message to be advanced, if necessary, to match that clock's timestamp value in the received message.

This allows the clock synchronization algorithm executed by a manager to omit: the retrieval of the agent's notion of the authentication clock of the party executing at the agent, the check for the agent's notion of that clock being slow, and the rectification of this condition through a `set` operation. The elimination of the potential need for this `set` operation is a significant benefit of this change.

Elimination of Ordered Delivery Mechanism

The Ordered Delivery mechanism required that any message delivered out of order be declared unauthentic. The SMP specification declared that this was unnecessary, detrimental to the efficiency of network management, and overlapped with other mechanisms of the Digest Authentication Protocol. In particular:

First, there is no security requirement to protect against malicious re-ordering of network management retrieval operations, especially since such re-ordering can and does happen through normal, non-malicious, operation of a network. Thus, the Ordered Delivery mechanism could cause genuine authentic messages to be declared unauthentic due to the normal operation of a network. Note also that this behavior is likely to happen more frequently under conditions of network stress, at which times network management needs to perform at its best.

Second, there *is* a security requirement to protect against malicious re-ordering of network management `set` operations. However, this requirement is not just protection for those issued by one network manager; rather, it must work for `set` operations issued on behalf of all network managers. The Ordered Delivery mechanism, however, operates only on messages sent by a single network manager, and therefore, is not sufficient to provide the required protection. To address this issue, SNMPv2 defines a new MIB object, `snmpSetSerialNo`, which provides the required protection.

Third, another mechanism specified as part of the Digest Authentication Protocol is the Message Timeliness mechanism. There was overlap in the functionality of these two mechanisms. With use of the Ordered Delivery mechanism removed, this overlap was removed. The Message Timeliness mechanism is retained to provide protection against malicious replay of a (retrieval) operation outside of the designated period (i.e., the lifetime) during which replay and/or out-of-order delivery can

occur non-maliciously due to the normal operation of the network.

For these three reasons, the working group approved the elimination of the Ordered Delivery mechanism.

Reduced Requirement for DES

Another change approved by the working group is a reduced requirement for the use of a privacy protocol, i.e., use of the Data Encryption Standard (DES). In particular, it is not required for the changing of party secrets. This change is possible because the distribution of new secrets in SNMPv2 `set` operations is achieved by including in the `set` operation, not the new secret value, but rather the XOR-ed value of the new secret and the old secret. While this provides no protection if the old secret was known by an eavesdropper, neither does the use of DES if the DES secret was already known by an eavesdropper!

A mechanism to further reduce the requirement for DES is one of the issues still being discussed by the working group. It is expected that the creation of new parties using SNMPv2, without the use of DES, will also be possible through having the secrets of new parties initialized as copies of the secrets of an existing party.

Other Changes

Other changes already approved are the updating of other aspects of SNMP Security's Party MIB in accordance with the SNMPv2 changes, including incorporating the values of the new SNMP PDU types in the access privileges variable, and updating the values assigned by convention to the initial party identifiers and their associated views and access control parameters.

Finally, instance-level granularity is now optional in the checking of whether a variable accessed by an SNMP PDU is in the relevant MIB view. That is, SNMPv2 entities acting in the role of an agent are not required to support MIB views where some instances of a particular leaf object type are in the MIB view, and some are not. However, these same MIB views are used in determining which SNMPv2 entities acting in a manager role should receive trap notifications. As such, agent implementors might still wish to provide instance-level granularity in order to allow fine-grain configuration of trap notifications. For example, the sending of `linkDown` traps for different interfaces to different managers requires instance-level granularity.

Changes Still Under Discussion

In the SNMP Security working group deliberations prior to its meeting last November, several issues were raised that had not been addressed by the changes specified by the SMP. Several proposals were put forward at that time to address these issues. The working group is currently trying to come to a consensus regarding these proposals, in their most recent form.

The issue of further reducing the need for DES was mentioned above. The other issues are:

- The so-called "party proliferation" problem, i.e., since parties identify not only particular entities and security properties but also a local MIB view or a particular proxy relationship, separate parties with the same security properties are required for each manager with access to multiple views/proxy relationships. Each such additional party increases the chore of maintaining the party clocks and secrets.
- A reduction in the amount of non-volatile storage needed to implement the Party MIB by agents with limited resources.
- Including a notion of the "temporal" semantics of MIB objects, e.g., not just the current values of MIB objects, but also their values at the next re-initialization/reboot of the agent, or their values at other times.

We'll see how these issues are resolved in this column next time.

Standards

David T. Perkins

In December and January, the Ether-like Interface, DS3/E3 Interface, and DS1/E3 Interface MIBs were published as RFCs. These are all updates to previous versions, and the old versions are now historic. Also published was the MIB for RIPv2. As reported in the last issue, the documents defining SNMP over other transports (OSI, AppleTalk, and IPX) are still in the pipeline, and are not yet published as RFCs.

Recently Published RFCs

RFC 1389 - RIPv2 MIB (Proposed Standard)

This document defines the MIB objects needed for monitoring and controlling systems that implement the RIP or RIPv2 routing protocols. The MIB consists of a global group that has global counters for RIP traffic generated, and routing table updates due to RIP. Following it are a table for *interface* status, a table for *interface* configuration, and a table containing information about interactions with RIP peers.

RFC 1398 - Ether-Like Interface type MIB (Draft Standard)

This document updates the previous version, RFC 1284. The major changes were to remove several objects due to implementation experience.

RFC 1406 - DS1/E1 Interface type MIB (Proposed Standard)

This document updates the previous version, RFC 1232. That document contained a fundamental error of specifying the syntax of many objects as `Counters` when the appropriate syntax should be `INTEGER` or `Gauge`.

RFC 1407 - DS3/E3 Interface type MIB (Proposed Standard)

This document updates the previous version, RFC 1233. That document contained a fundamental error of specifying the syntax of many objects as `Counters` when the appropriate syntax should be `INTEGER` or `Gauge`.

RFC 1414 - Identification MIB (Proposed Standard)

This document extends the `tcpConnTable` of MIB-II by associating user identification information with each connection.

Standards Progression of MIBs

As was previously noted, many MIBs are at the Proposed stage (step 1), but few have moved to the Draft stage (step 2). Why is that, and what can be done? Below are listed some of the factors which have delayed the advancement.

To create a new MIB, a new working group is formed, which produces a document that is submitted for consideration as a Proposed Standard. In the past, after the WG completes the document, it disbands. Thus, no real force remains to keep the effort going. To counter this, WGs will no longer disband, but go dormant for a time while the MIB is being implemented: the WG's charter will be updated, it's mailing list and archives will remain, and the chair will remain filled, but no meetings are scheduled. This change should allow a forum for implementors to communicate their experiences, so that interoperability testing can be planned, and operational experiences exchanged. These are required for advancement to the next stage.

Another factor that has kept many WG members busy over the last year has been SNMP Security and then SNMPv2. The individuals that were involved in developing the existing MIBs have been working on evaluating and implementing these new management thrusts. Now that these efforts are coming to a close, attention can come back to the existing SNMP MIBs.

The above two factors do not account for all of the delay in the progress of the MIBs. The biggest factor that has been shown time and time again in the IETF is leadership by a few key individuals. The effective leaders have been able to "corner" the needed talent, and get commitments by companies who are fierce competitors to work together to accomplish a common goal. The MIB process needs one or two leaders.

In the next issue, a profile will be given of a successful IETF leader. Maybe some of ***The Simple Times*** readers will recognize themselves and step up to the challenge.

Summary of Standards

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).

Draft Standards:

- 1398 - Ether-Like Interface Type MIB.

Proposed Standards:

- 1229 - Extensions to the generic-interface MIB;
- 1230 - IEEE 802.4 Token Bus Interface Type 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;

- 1351 - SNMP Administrative Model;
- 1352 - SNMP Security Protocols;
- 1353 - SNMP Party 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; and,
- 1414 - Identification MIB.

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);
- 1238 - CLNS MIB;
- 1283 - SNMP over OSI; and,
- 1298 - SNMP over IPX.

Informational:

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

Historical:

- 1156 - Management Information Base (MIB-I)
- 1232 - DS1 Interface Type MIB;
- 1233 - DS3 Interface Type MIB; and,
- 1284 - Ether-Like Interface Type MIB.

Working Group Synopses

Frank J. Kastenholz

With this issue of ***The Simple Times***, I am taking over this column from Bob Stewart. Over the past year, Bob has done an outstanding job in reporting SNMP activities in the IETF's working groups. I hope that I can live up to the standards that Bob set.

This column contains my own distillation of the conversations that go on in the various working groups. Rather than attempting to report statement by statement what has gone on, I have usually tried to condense the messages that make up a discussion and present the gist of the thread.

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 summary of *The Economist's* year-end article "The Good Network Guide" was posted. This article placed in the Internet in the company of *Skull and Bones*, *Rhodes Scholars*, the *Muslim Brotherhood*, *Freemasons*, and *Opus Dei*.

The perennial question of when to return `readOnly` status was raised. The poser of the question indicated that *The Simple Book* indicates some conditions when a `readOnly` status should be returned, but the SNMP RFC does not. It was pointed out that *The Simple Book* is wrong; and that the SNMP RFC does define `readOnly` but does not specify its use. Whenever an attempt is made to `set` a variable which is not writable, `noSuchName` should be returned. The reason is that the variable is not in the MIB view that is applicable to the `set` operation. One message further went on to state that no response should be sent by the agent as attempting to do a `set` without having the proper privileges should be considered an authentication failure.

A question about how to handle zero-length integers was asked. One response said that this is a parse error and you should increment any parse-error counter and discard the packet. This view was supported by a second response which said that this encoding is not allowed under the Basic Encoding Rules (BER).

A question on how to configure SunNet Manager to be able to access a MIB was posted. A response was posted directing the original questioner to a SunNet Manager users' mailing list, `snm-people@zippy.arizona.edu`.

A question about SNMP Security work was asked. The question revolved around the party concept. The person asking the question wanted to know how to differentiate different people requesting management operations on an agent, assuming that a management station had a single party. The response pointed out that a management station could have multiple parties associated with it and that the management station had to perform any user/party associations. The response also pointed out that the current SNMP Security RFCs (RFCs 1351, 1352, and 1353) are being updated and incorporated into the SNMPv2 work and that fielding products based on the current RFCs was not recommended. Several questions were posted looking for mechanisms, comments or advice about standard methods for extending SNMP agents for managing network applications. There were some direct responses from vendors.

In addition, a discussion of the "dispatcher" concept developed. In this concept, a special de-multiplexor listens to the standard UDP port. This module then dispatches requests to the correct *sub-agents* which listen to their own, individual, ports. A configuration file provides all of the bindings needed between the sub-agent's ports and the OIDs. Some technical concerns with the approach were raised, such as how to deal with a sub-agent that has to support a subtree in the domain of another agent, and how to deal with different sub-agents handling different instances of variables, such as rows of a table.

Someone asked if there is a standard way for running SNMP over a serial line, such as SLIP or PPP. This person suggested that by using dial-up connections in this manner, a manager station could connect to network elements on the far side of a network partition.

A question regarding the `ifTable` in dynamic environments was asked. In some environments, the number of interfaces can change without requiring the management system to be re-initialized. This is a problem since the definitions of `ifNumber` and `ifIndex` require that the `ifTable` be a fixed array. The response suggested that the entries in the `ifTable` remain in existence but be removed from all MIB views. One person asked if Gauges must be read-only. The protocol does allow Gauges (and Counters or Timeticks) to be read-write.

A new version of the *Beholder*, an RMON-compliant Ethernet Monitor, from Delft University of Technology was announced; it is available for anonymous FTP from `dnpap.et.tudelft.nl` in the directory `pub/btng`. There is also a Beholder mailing list: `btng@dnpap.et.tudelft.nl`.

Interest in the auto-discovery problem was expressed. SNMPv2 should at least have the hooks necessary to define auto-discovery. Some considered it ill-mannered not to answer when asked your name. There was some resistance to reviving this discussion. It was pointed out that to be useful, there needs to be a well-known community or party with read-only access to the `system` group and a requirement to respond to requests received on a broadcast address. It is believed that SNMP Security will have the needed required initial parties. MIBs need more information, such as short and long descriptions, the name of the MIB, more descriptive detail on indexes, and so on. One person pointed out that HEMS had such information.

Appletalk/IP Working Group

Submissions: `apple-ip@cayman.com`

On January 25, 1993, a query on the status of the *SNMP over AppleTalk* draft was posted to the list, noting that the Internet-Draft had expired. The response was that publication was imminent.

BGP Working Group

Submissions: `iwg@ans.net`

If traps are lost, counters of state changes are needed if the state change traps are lost.

A new Internet-Draft of the BGP MIB has been published.

Bridge MIB Working Group

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

A request for the draft of the Source Routing MIB was posted to the list. A question, asking when new values for the Max Age, Hello Time, and Forward Delay become operational, was posted. A response noted that 802.1D says that new values take effect at initialization time and when becoming the root bridge.

Character MIB Working Group

Submissions: char-mib@decwrl.dec.com

A request was made to add enqHost(6) and enqTerm(7) to the enumerations for charPortInFlowType and charPortOutFlowType. This is needed to support HP-style flow control. This was believed to be reasonable if it represents a common need.

The problem of supporting multiple flow-control methods in charPortInFlowType and charPortOutFlowType was discussed. It was suggested that the enumerated BIT STRINGs of SNMPv2 could be used for this. The question of whether or not existing implementations actually have this problem was posed.

The question of whether the Character MIB covers virtual ports, such as on an X.25 gateway was asked. The only response indicated that this was the intent.

The chair of the working group requested reports of implementation status for the MIBs. This information is needed as the MIBs are considered for Draft Standard. At least 5 vendors indicated that their agents support the MIBs and at least three network management stations include the MIBs.

Chassis MIB Working Group

Submissions: chassismib@cs.utk.edu

For a while, the mailing list was temporarily off-line.

A request was made to have both nominal and actual values for sensors.

A proposal was made in the Washington DC IETF meeting to change models. The goal is a more generic MIB with less complexity. In this model the chassis is a generic container. The physical view of the chassis has compartments of certain types, while the logical view has entities with certain network functions. An entity is composed of one or more physical modules. The concept of a segment has been removed as it is specific to some implementations. However, some people pointed out that the segment is still a useful element since they can simplify certain relationships in the chassis. The group decided that they would be kept and modeled as entities. This proposal received a lot of support.

Several questions were raised on how Chassis MIB relates to possible hardware implementations:

- Are segments internal or external, to the link entities?
- How does a slot relate to set of MIBs?
- How does the Chassis MIB tie slots, entities, and ports together?
- Is the OID in chasEntityEntry a branch or a leaf?
- Does chasEntityTable provide validation for requests to the entity?

There was a question as to how a virtual slot, such as a built-in manager, should be numbered. One suggestion was to use a DisplayString with the real name of the slot. Another proposed solution was to index the slot table by an integer and an OID indicating the content of the slot. The latter method was considered easier to do.

The issue of how to identify sensors that are slot-specific rather than chassis-wide was raised. The solution is to use the slot number as an index, with 0 indicating chassis-wide.

A new internet draft was published on January 14, 1993.

DECnet Phase IV MIB Working Group

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

No traffic to report.

Ethernet MIB Working Group

Submissions: enet_mib@ftp.com

On January 15, 1993, the announcement of the publication of the latest version of the MIB for Ethernet-like interfaces was forwarded to the list. This MIB is available as RFC 1398.

The Ethernet MIB Working Group has terminated. The mailing list will remain active as a forum for implementors.

FDDI MIB Working Group

Submissions: fddi-mib@cs.utk.edu

On January 5, 1993, a request for the status of the MIB was posted to the list. A response indicated that the MIB is being revised to conform to ANSI SMT 7.2 (the current MIB is based on SMT 6.2).

On January 15, 1993, a message was posted that was an exhortation for all participants of the working group to contribute any notes, text, or other information that they may have from the November IETF meeting in order to help the document's editor to complete his task. On January 21, a response was posted that had several points on this topic:

- The SMTTraceMAXExpiration object should not be in units of nanoseconds, since the default value of the object is 7 seconds. Millisecond resolution was proposed.

- The SetCount variable should be removed since it is not clear how the object can guarantee consistency and the proper reliable locking which it is supposed to provide. Part of the problem is that the SetCount object was optional.

- It was pointed out that PORTRequestedPaths is not a single integer, as specified in the FDDI MIB, but rather is three separate integers. It was suggested to turn this MIB object into a 3 byte OCTET STRING, each byte representing one of the integer values.

Host MIB Working Group

Submissions: hostmib@andrew.cmu.edu

On January 7, 1993, in response to an announcement on the publication of a new draft of the MIB, a message was posted asking for the rationale for the hrDiskStorageMedia and hrFSType objects. The message also indicated that there were discrepancies between the types defined for hrDiskStorageMedia and hrStorageTypes. The post also asked how space for a jukebox should be calculated (three alternatives were suggested: space of currently accessable platter, space of all platters in the box, and space of all platters now in data base that can be loaded into box). More types for hrFSTypes, covering network file systems such as NFS, AFS, and RFS, were proposed.

Hub MIB Working Group

Submissions: hubmib@synoptics.com

An inconsistency in the wording of the MIB was brought up. It was pointed out that the DESCRIPTION for the rptrNonDisruptTest says that the test does not change the repeater's state. However, the text also says that a rptrHealth trap will be sent. Later on, the description of the rptrHealth trap indicates that the trap is sent only when the status of the repeater changes. One response suggested that any test implicitly changes the repeater's state. Another response suggested that the rptrHealth trap is supposed to be sent only if the test actually caused a change in the repeater's state (e.g., if it found a problem in the repeater). Another response alluded to the original IEEE draft, suggesting that the trap would be used to mimic the confirmed nature of CMIP actions; the trap is used to inform the management station when the test is completed. It was then pointed out traps are not reliably transmitted, plus, if two managers run the same test at about the same time, each management station would get the trap indicating the completion of its own test and that of the other management station. A final suggestion was that the description of the trap be changed.

A question was raised about why there is no object in the Repeater MIB to indicate the current state of a port or link. It was pointed out that a link integrity object will be in the MAU MIB.

IDPR Working Group

Submissions: idpr-wg@bbn.com

On December 31, 1992, a notice was posted that the IDPR MIB is being updated and a new Internet-Draft would be posted shortly.

On January 4 and 5, 1993, a general discussion on the structure of the MIB took place. Some of the salient points of that discussion were:

- A PG to VG map was proposed. One area of concern was whether there should be a PG-to-VG or VG-to-PG map. The former was preferred as some people indicated a preference for Physical over Virtual models.

- Some editorial ambiguities were pointed out, and corrections proposed.

- The use of separate tables for different address types was discussed. One person suggested the use of a mechanism where any address type could be represented in the address field. A second person indicated a preference for having explicit ASN.1 descriptions for each data type in the MIB and not leaving it up to an application to figure out what format some particular data item is.

- The issue of whether certain management operations should be reflected as changes to the configuration file, or immediate changes to the state of the router, was discussed.

- In addition to this, the issue of how to specify the syntax for configuration files was discussed. An objection to explicitly including this syntax in the MIB specification was raised, though a DisplayString could be used, allowing agent-specific syntaxes.

IDRP for IP Working Group

Submissions: idrp-for-ip@merit.edu

At the Washington DC IETF meeting, the chair asked for volunteers to work on the MIB. This working group can be expected to start development of a MIB shortly.

IPLPDN Working Group

Submissions: iplpdn@nri.reston.va.us

No traffic to report.

IS-IS Working Group

Submissions: isis@merit.edu

No traffic to report.

NOCtools Working Group

Submissions: noctools@merit.edu

On January 11, 1993, an announcement was posted that the IESG had approved the latest NOCTools document for publication as an Informational RFC.

OSPF Working Group

Submissions: ospfigp@gated.cornell.edu

No traffic to report.

PPP Working Group

Submissions:iietf-ppp@ucdavis.edu

No traffic to report.

RIP Working Group

Submissions: ietf-rip@xylogics.com

On January 5, 1993, an announcement was posted indicating that the MIB for RIP version 2 was published as RFC 1389.

On January 12, 1993, an announcement was posted indicating that the RIP version 2 working group was being terminated as it had finished the work for which it was chartered (development of the MIB, among other things). The mailing list will remain in existence as a forum for implementors.

TCP Client Identity Protocol

Submissions: ident@nri.reston.va.us

The Identification MIB was published as RFC 1414.

Remote Monitoring (RMON) MIB Working Group

Submissions: rmonmib@jarthur.claremont.edu

The working group's mail address has been changed. Note the new address above.

A brief discussion of the performance of token-ring networks with "lots" of bridges between source and destination occurred. This had to do with the efficiency of RIF field processing.

On December 29, 1992, two questions were posted on implementation issues for RFC 1271: First, comments for the statistics group indicate that the statistics start at 0 when entries in the tables are created. It was pointed out that this contradicts the notion that Counters in SNMP do not have a specific starting value. Some of the discussion centered on whether the agent or the manager station should maintain baseline values. One message said that Channels might be used as data sources for the various tables. If and when this happens, the counters in the tables will not be deltas on a single base value, but completely different values. There was no resolution to this issue on the mailing list. Second, when rows are under creation, it is not clear what the proper response of the agent should be when `get` or `get-next` requests are received for the row. Both not having the row available in the operations' MIB view and using place-holder values were suggested as responses. It was pointed out that the former method would not cause problems to management stations since management stations must be able to handle missing objects. It was also pointed out that providing proper `DEFVAL`s for all objects that need them, which is good MIB design, would eliminate this problem since there would always be a correct value to return for any object. Some of the objects cannot have `DEFVAL`s since they would contain values that are known only at run-time and cannot be known to the MIB writer (one example is `hostTopNControlHostIndex`).

The discussion on what value to start counters at evolved into a discussion on whether Channels were going to be added to the MIB or not, and a general discussion on what valid sources for data were. One note asked whether a repeater port can be a source, and if so how to identify that port since repeater ports are identified by group and port numbers in the Repeater MIB.

Consensus was called for on a couple of issues: the simplified `RingStationOrderTable` and not having a history of `RingStationOrderTables`. There was no dissent.

A question was asked about how to maintain the count of `ringStationDuplicateAddresses`. One response was that the the Error Code subvector of the Report Active Monitor Error MAC frame can be used for detecting

duplicate addresses. If this subvector is set to 0x0003 then a Duplicate address is detected.

A relatively short question about RMON Filter Groups was asked: if a filter is set up then does that filter apply to the packet capture group only or does it apply to all other groups? There was a large discussion. Most responses indicated that filters apply only to the packet capture group. One response said that only in early implementations are the Statistics groups attached to `ifTable` entries, thereby limiting their input to unfiltered packet streams. Eventually, statistics groups could be attached to channels, and channels are filterable.

This discussion briefly evolved into a discussion about having channels being the data source for other channels. It was pointed out that this was discussed earlier and rejected since changing channels simply connects filters together, which can always be done on a single channel.

Several announcements were posted to the list. A paper for the January 1993 USENIX conference on the Berkeley Packet Filter was announced. It is available for anonymous FTP from `ftp.ee.lbl.gov` as `bpf-usenix93.ps.Z`.

SNMP Security Working Group

Submissions: `snmp-sec-dev@tis.com`

The main activity of the SNMP Security working group has been the upgrading of the original SNMP Security RFCs (RFCs 1351, 1352, and 1353) for the SNMPv2 effort.

SNMPv2 Working Group

Submissions: `snmp2@thumper.bellcore.com`

A question was asked about where the version number in the SNMP packet was changed for SNMPv2. It was pointed out that the SNMPv2 packet does not have a version number and that the two packets can be distinguished by their different ASN.1 tags.

A discussion was held on the upgrading of MIB-II to SNMPv2 standards and conventions. Some people asked whether this would be done, and if so when and how. The basic resolution was that MIB-II would be upgraded in a "group-by-group" manner after the SNMPv2 work is completed. At the same time, it was pointed out that work needs to be done to revise parts of MIB-II to reflect the latest implementation and operational experience.

A question was raised whether compatibility with existing MIBs is a requirement that constrains the SNMPv2 effort or not. A side point was made that standard MIBs are not that important since it is the device or enterprise-specific MIBs that provide real manageability.

The question of why a lot of SNMP statistics were removed was asked. The answer was that these statistics did not provide useful information and were simply a burden on agents.

A proposal to change a part of the introductory text of the SNMPv2 document was made. This was moved to the security group.

The working group considered having a special meeting in December to finish its work. This meeting was not held; the working group decided that it could complete its work via electronic mail.

An issue, originally raised on the mailing list before the Washington DC IETF meeting, was brought back. The possible problem was that for certain types of requests, an agent might need some "thinking" time before it could complete or refuse the request. It was pointed out that this is a problem for manager stations in timing-out and retransmitting requests. Is the lack of response due to the agent thinking or the request or response getting lost? The original note stressed that this could be a severe problem for `set` requests. One suggested solution was to solve the problem by MIB design (adding additional variables and semantics). Other suggestions included use of multiple parties, extending the `RowStatus` textual convention, and so on. Several people pointed out that this might be a small problem and the effort to fix it might be out of proportion to the problem being fixed. Use of "cute mibs" was decried as being unappealing. Other suggestions were to generate a "completion" trap, have a special "completion" status variable which could be polled, or reverse the normal manager/agent relationship and have the agent `set` a variable in the manager that indicates completion. There were many comments to the effect that, whatever the solution is, it is important to keep it simple. This prompted a response which pointed out that the initial premise of SNMP was that it was interim until a better solution was developed and that SNMP has lost its interim status, implying that complexity might be acceptable now. This all got subsumed in the great "Spatial/Temporal" discussion.

A general issue was pointed out, that certain MIB operations have *spatial* and *temporal* aspects to them. For example, setting variables in configuration NV-storage versus running memory. Some general solutions were offered, such as extending the OID space by adding such spatial and temporal information; the other proposal was to include the needed information in the Party and access control tables. The former proposal had the property that each variable-binding identified its own time and space contexts. It was suggested that this might be an implementation nightmare. One post

claimed that parties and the table were for security, while the issue under discussion was not a security issue; the two are different and should be solved by different methods. A rather nasty exchange then ensued. This was a particularly amusing exchange to watch. It was complete with offerings to resign by one of the working group's chairs, insinuations that people's motives were sinister, lawyerly discussions on the scope of working group charters and historical revisionism. All this while keeping pretense of discussing architectural versus implementation issues and protestations that the parties to the exchange really were all working together to produce the best technology. In short, the true technical content was minimized. The chair of the working group pointed out that the proposal that used OIDs to contain spatial and temporal information was more in the nature of an interesting approach and did not have enough detail to warrant being called a proposal. This conversation was also carried out on the SNMP Security working group's mailing list.

As reported in an earlier edition of ***The Simple Times***, the minutes of the Washington DC IETF meeting were posted to the list. A question as to why the `Set2Default` and modified `get-bulk` proposals were not discussed. It was reported that there was no time at the meeting and that discussion would continue on the mailing list, with a resolution deadline of December 4, 1992. Several minor problems with the minutes were pointed out and a revised version was posted.

After much discussion, the chair asked for consensus on not adopting `Set2Default`. There was no objection.

The chair asked for consensus on not adopting the Modified `get-bulk` proposal. Some objections were stated, i.e., `get-bulk` should return only data that was asked for. Additional data might imply additional kernel requests, larger PDUs, more ASN encoding/decoding and so on. These were not seen as compelling reasons to adopt the modifications (similar ones were rejected at the IETF meeting). It was pointed out that the increase in efficiency is at best marginal. Argument raged for a while about who was better at performing simple arithmetic and who had the better model of operations and likely use to support their claim that `get-bulk` needed modification or not. Alternative modifications were also proposed, all rejected. In the end, while it was agreed by all that `get-bulk` is not quite as efficient as it could be, `get-bulk` stood as originally proposed, its efficiency being deemed good enough most of the time.

A proposal was made that the MIB references in the `AGENT-CAPABILITIES` macro need to include MIB version information in order to identify which version of a MIB is supported by an agent. This was adopted. A similar mechanism was also adopted for the `MODULE-COMPLIANCE` macro.

A proposal was made for allowing a deletion function with assertions on row contents. There was little support for this proposal and the chair called for consensus to drop this proposal. One person objected to dropping it, feeling that lack of support for the proposal is not a sufficient reason.

A concern was raised that when creating rows, it might not be possible for the agent to check to see if all needed data are present and consistent because of the ordering of the variables in the PDU. This was pointed out to be a non-problem since the protocol defines a conceptual two-phase process, wherein the data are first validated, and then the actual `set` occurs. An implementor also offered a three-step process, where the variables in the `set` are first checked individually (correct syntax and range), and then checked as a group for consistency.

The SNMP Security work has caused some changes to two SNMPv2 documents. Brief descriptions of the changes were posted. An objection was made that they represent changes to the SNMPv2 work without the consensus of the SNMPv2 working group. This discussion re-started the usual finger-pointing seen earlier with the spatial/temporal discussion. Fortunately, this was averted when one of the parties learned about time zones.

A discussion on the `undoFailed` return code was held. This was started by the suggestion that a requirement be made that all variable-bindings for a particular row appear contiguously in a PDU. This particular idea was discarded earlier by the working group. The relation to `undoFailed` had to do with figuring out the meanings of errors when it is returned. One person pointed out that `undoFail` should never happen since a reasonable agent would validate everything first, preventing an undo from ever occurring. This person also pointed out that if you cannot rely on "doing" changes, you cannot rely on "undoing" them. The suggestion was made to replace the undoFailed code with `agentHasJustShotItselfInTheHead`. The fact that an undo should never occur was generally recognized, though `undoFailed` was kept in the realization that not all agents would be on systems where the proper locking and synchronization primitives would be available. A secondary discussion suggested removing `undoFailed`, but keeping `commitFailed`, on the basis that `undoFailed` ought never occur. It was claimed that the two were a package deal and both should be kept. This claim was disputed, but it was pointed out that having a `commitFailed` implied a two-phase commit algorithm, which required an `undoFailed`. The chair suggested that since no bug has been demonstrated in the error codes under discussion, they be left intact.

A proposal was made to add an additional value to the RowStatus textual convention. This value would be changeAndGo. A row's status object would be set to this value when the manager wished to make changes to existing instances of the row. The proposal suggested that this value would help managers resolve situations where one manager is in the process of reading and then changing values in the row while another manager deletes the row. The proposal suggested that the first manager would set the row's status column to changeAndGo and, if the row had been deleted before the set operation took place, an inconsistentValue error would be reported on the status column. A response pointed out that setting the status column to active would have the same effect and that an additional value was not needed for RowStatus. No further discussion took place. Furthermore, it was ruled out-of-order since the deadline had been passed and the proposal did not fix a clearly stated and widely agreed upon bug.

A proposal for bringing back the row creation and deletion PDUs was made. One reason offered was the complexity of the so-called "RMON polka". It was pointed out that this issue was discussed for quite some time at the IETF meeting and that the consensus of the group was that these PDUs did not solve the problem. The chair ruled continued discussion out of order unless a bug with RowStatus was found. The chair was ignored amid insinuations that attempts were being made to rubber stamp the original SMP proposals. Naturally, such accusations were forcefully denied and another series of messages flowed back and forth consuming bandwidth and patience and producing nothing useful. Eventually, technology crept back into the discussion and a suggestion was made to refine the wording of RowStatus to address the lingering concerns or issues that the creation and deletion operations were meant to deal with.

One of the continuing problems was the need to ascertain a valid instance-identifier to use when creating rows (sometimes referred to as the "take a stab in the dark" problem). A suggestion to allow agents to return different values than were in the set request was made. One alternative suggested was to define a MIB object that returned a unique value every time a get operation was issued against it. In order to put the problem in perspective, a proposed list of requirements that any solution must address was posted. This list also compared the RowStatus and create/delete PDU approaches. This list received some discussion. One message noted that some of the problems that create and delete are alleged to solve are, in fact, general problems with set operation and if a solution is developed, it should be a general solution.

The chair called for consensus that create/delete PDUs were not to be adopted. This brought even more discussion than in the past. One message pointed out that while create and delete do not solve all problems with row creation and destruction, neither does RowStatus. Those backing the proposal claimed that the group had consensus that create and delete solved 95% of the problems. Many messages agreed with the chair's call. The chair made another call. One argument was that RowStatus adds state; it was pointed out that this does not, in fact, add state to the protocol, merely the MIBs, which already have state. Another claim was that only very wide tables needed RowStatus; for narrow tables it is an excessive burden; however, the burden of supporting multiple row-creation methods was also claimed to be the heavier.

The conversation then turned back to refining and updating RowStatus. One issue that needed to be addressed was timeouts in some of the states. Many of these issues were addressed by making the wording clearer and more explicit. Several messages discussed the need to have a single method of creating and deleting rows. Having unique methods for each row was viewed as making management overly complex. It was also pointed out that RowStatus does not bar other methods from being devised. One person pointed out that there already are many methods of row creation in the field today and likely to stay there in the future. Having a single method in SNMPv2 would not be as big a win as would first seem.

A new version of RowStatus was posted that addressed several of the issues. This proposal then received continued discussion. The create/delete issue was still brought up. This issue was rejected with an explanation of how the problems that only create/delete could fix are, in fact, fixable via mechanisms already available. Several typographical errors in the revised RowStatus text were identified and corrected. Clarifications to text covering explicit descriptions of the legal and illegal state transitions and that some tables might not allow setting when the row is active were proposed and accepted. Addition of text covering the effects of access control was also proposed and accepted.

Another version of RowStatus was posted, reflecting the concerns and changes to date. Including a state transition table in the RowStatus description was proposed and adopted. Having a table was felt to be clear and unambiguous. After proposing the general idea of a state table, there was extensive discussion on the exact states, actions, and transitions that the table would have.

One significant concern was the apparent inability to do "single-PDU" row-creations. This was solved by using two different RowStatus values: createAndGo

and createAndWait. The former would immediately instantiate the row while the latter would let the manager incrementally build the row and then instantiate it (the original RowStatus approach). This received much support. Consensus was called for on the revised RowStatus. It was obtained. Naturally, as more people reviewed the work, there was more discussion after the consensus.

A query was made as to whether there would be a SNMP2 technology demo at the 93 Spring INTEROP. The reply is that no official demos were planned, however, individual vendors might have their own demos and the upcoming IFIP WG6.6 Symposium would have a demo.

Most significantly, the working group's chair called for the final consensus of the working group. This brought out a minor flurry of last minute minor changes. Otherwise, consensus was reached and it seems that the working group has accomplished its objective. Naturally, even after the call, discussion continued.

A suggestion was made to add an additional clause to the OBJECT-TYPE macro. This clause, the HELP clause, would be used by management station vendors as a place where they can add additional help information. The proposal pointed out that the text in the DESCRIPTION clause is usually written for developers and implementors, not for network operators, and the latter may find the text difficulty to understand. The only response to the proposal pointed out that, while the idea was a useful one, had merit, was similar to ideas that were previously discussed and should be looked at in the future, the proposal was made after the group's deadline for new proposals.

An error in the definition of the DisplayString textual convention was pointed out. They syntax should have a SIZE constraint. This correction was accepted by the Working Group. As an aside, it was also pointed out that many implementations do not properly format a DisplayString according to the rules for NVT. It was suggested that additional text be added to the DisplayString definition, stating what is and is not legal in NVT. It was also pointed out that it is legal for the SYNTAX clause in an OBJECT-TYPE macro to specify a smaller SIZE than is in the TEXTUAL-CONVENTION.

As a result of implementation experience, it was determined that a new error code is needed for the set operation. This code is needed for the case where an attempt is made to create a new instance of a variable and, for some reason, that particular instance cannot be created, because the name would conflict with existing information in the agent (e.g., trying to create an access control entry for a party that doesn't exist). The proposed new error code is inconsistentName and indicates that the variable doesn't exist, the agent is able to create instances of the corresponding object type, but the name in the variable-binding is inconsistent with the current state of the agent's MIB. One response suggested overloading the meaning of notWritable. This was rejected as the meaning of the notWritable error code would become ambiguous.

A message correcting the location of the working group's archive was posted to the list. The archive is on thumper.bellcore.com in the file pub/davin/snmp2-archive.

Trunk MIB Working Group

Submissions: trunk-mib@saffron.acc.com

On January 8, 1993, two announcements were posted to the IETF mailing list. These announcements were for new Internet Drafts for the DS1/E1 and DS3/E3 MIBs. These versions of the MIBs incorporate comments made during their Last Call period.

UPS MIB Working Group

Submissions: ups-mib@cs.utk.edu

The results of a survey of the implementability of variables in the MIB was posted to the list.

X.25 MIB Working Group

Submissions: x25mib@dg-rtp.dg.com

On January 11, 1993, a new version of the X.25 Multiprotocol Interconnect MIB was posted to the mailing list.

Activities Calendar

- INTEROP 93 Spring
  March 8–12, Washington, DC
  For information: +1 415–941–3399
- 26th Meeting of the IETF
  March 29–April 2, Columbus, OH
  For information: +1 703–620–8990
- IFIP Symposium on Network Management
  April 18–23, San Francisco, CA
  For information: +1 415–512–1316

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).