The Simple Times™

THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS℠
VOLUME 4, NUMBER 3 JULY, 1996

The Simple Times is an openly-available publication devoted to the promotion of the Simple Network Management Protocol. In each issue, ***The Simple Times*** presents technical articles and featured columns, along with a standards summary and a list of Internet resources. In addition, some issues contain summaries of recent publications and upcoming events.

In this Issue:

The Simple Times is available via both electronic mail and hard copy. For information on subscriptions, see page 24.

Towards Useful Management

Chris Wellens, Interworking Labs
Karl Auerbach, Precept Software

SNMP's greatest success is in providing the framework to deliver management capabilities for highly focused, device specific applications. The industry needs to move beyond this accomplishment.

The purpose of this article is to consider new concepts and capabilities in network management. These range from incremental enhancements of the current state of affairs to wild-eyed dreaming.

The approaches are these, in order of increasing departure from current practices:

- enhanced MIB definitions with greatly increased MIB semantics, in particular, the creation of "meta variables";
- embedding of management applications into devices, with control interfaces exported to humans via HTTP/HTML based web pages;
- replacement of the SNMP access method with one based on HTTP;
- replacement of the SNMP access method with one based on long term "associations";
- simple management by delegation through the use of script MIBs;
- semi-autonomous area managers; and,
- network management "worms".

These approaches are not mutually exclusive.

MIBs Are Precious

During the SNMP years, we've come up with reams and reams of MIB definitions. These MIBs comprise the collected thoughts by experts of what exactly constitutes the valuable data points needed to monitor and control a device. These MIBs are the most valuable legacy of SNMP.

Along with the MIBs themselves, we have learned the value of concise, machine-parseable MIB definitions.

These form the fundamental vehicle by which a general purpose management station can learn the about the devices under its control.

Myths

Network management in the Internet has been the product of many myths. At least two of these have been shown to be mere vapors:

The Myth of The Collapsing Network:
Connectionless transports, such as UDP, have been advanced as necessary for network management because of their ability to work when the network is failing. To put the myth contrariwise, management using TCP is deemed impossible because the myth asserts that TCP streams will break but that trusty UDP will get through to save the day.

We must first recognize that there is a distinction between the "network management" of monitoring and capacity planning from the "network management" of troubleshooting. For convenience, we'll refer to the former simply as "network management" and the latter as "troubleshooting".

Nearly 100% of network management occurs when networks are not failing.

When today's networks break, it is usually due to either a hard connectivity failure or a routing failure. In either case, neither TCP nor UDP get through.

Error bursts and congestion failures do occur, but these tend to be transient, and whether performed by the TCP engine or in a network management station's SNMP retry logic, the packets do tend to get through eventually. It is interesting that with TCP's congestion avoidance algorithms, TCP based streams behave in a way more likely to alleviate the congestion than unregulated UDP streams.

Quality of service controls (such as RSVP) are coming to the Internet. We expect management traffic will get the ability to request priority. This will help ensure that as long as a pathway exists, there will always be a way to monitor and control the net no matter how congested it gets.

Troubleshooting is a distinct branch of network management and requires tools and techniques quite different from those used for continuous monitoring and control. In troubleshooting, SNMP is, at best, a tertiary level tool with value rather below that of "ping", "traceroute", "nslookup", and "mtrace".

The Myth of the Dumb Agent:
How often have we been told that agents are simple-minded devices that can't support anything other than a simple SNMP agent? Even if that were true nine years ago, an assertion to which our experience speaks to the contrary, it is completely untrue today.

Today's network devices often contain processors and memory exceeding that of our management platforms of a few years ago. Already these devices perform numerous autonomous operations and have considerable protocol stacks already in place.

Today's network devices are capable of managing themselves, if given the opportunity. (We must admit, however and unhappily, that there are is a very large class of price sensitive devices in which every corner that could be cut was cut, including the time to read the relevant specifications or perform any interoperability testing.)

Next Stop Where?

So where should we take network management? The next sections discuss a few ideas, ranging from the incremental to the radical.

Meta Variables

The "Meta-Variable" concept has been around for at least the last six years. It is simple to do and requires no changes to existing protocols or agent implementations.

A meta-variable is simply a MIB variable which exists only in the MIB definition document. Each meta-variable is defined as a function of real MIB variables.

Meta-variables would be used by MIB designers to express useful derivations that can be made from the raw data. This could capture a significant body of empirical knowledge which today is rarely, if ever, recorded.

The function may be simple, such as the dividend produced when an error counting `Gauge` variable is divided by `sysUpTime`. In this case, the result would be an average error rate.

Or the function may be more complex, like something that takes the second derivative with respect to time of that error counting `Gauge`. This function would highlight significant changes in the error rates on an interface, which is a far more useful indicator of trouble than an average error rate.

To reify these meta-variables, a management station would have to perform the function. This implies that the function must be expressed by some procedural statement that can be mapped down to basic SNMP `get` and `set` primitives and polling. One might say that the functions would be best expressed as simple scripts.

The definition of these meta-variables and the functions used to generate them would be expressed in standard MIB definition documents with appropriate formalities so that they could be machine parsed and utilized by a management station.

An extension of the meta-variable concept is to place intermediary devices in the network whose role is to compute these meta-variables and export them as real SNMP variables in a MIB specific to those intermediary devices. Another extension is for the SNMP agents themselves to compute the meta-variables, in which case they become real-variables.

Embedded Management Applications

The World-wide Web is everywhere. Everybody has a browser. These browsers are a standard user interface available to any application which chooses to communicate using the Web's native protocol, HTTP, as specified in RFC 1945.

Although SNMP itself is relatively "simple", it takes some work to build the MIB support in an agent, and considerably more work to build the management support to utilize the MIB data, and a great deal of work to deploy the manager onto the various network management "platforms".

An HTTP/HTML management server embedded in a managed device, with underlying TCP is not significantly more complex or memory intensive than an SNMP agent with mechanisms supporting generalized lexi-ordering and arbitrary collections of objects in a `set`. (It is easy to vastly underestimate the amount of work required for an agent to handle an arbitrary collection of proposed values which may arrive in an `set` request.)

If one looks at many of today's workstation-based management platforms, one quickly realizes that they are really not much more than a collection of device-specific add-ons.

Those add-ons could be just as easily created by having a device export highly device specific web pages with controls and user interface paradigms. For example, management platforms take pride in the fact that they can project a rendering of a managed device, so that the operator can point at a port to invoke a control panel for that specific port. This is pretty routine stuff for a typical web server.

The device vendor ships one, self contained product. That product includes its own management functions and does not depend on anything except that WWW browsers are reasonably uniform and ubiquitous. With respect to its management functions, the vendor controls the horizontal and it controls the vertical; the vendor controls everything about the device and its management, from operation to GUI. It's an extremely attractive proposition.

The great drawback of this approach is that it requires human intelligence to comprehend the WWW forms presented by a device. If one accepts the proposition, as we do, that in the long-term, networks should perform significant self-management, then this approach represents a substantial danger that we will end up further from our goal rather than closer.

Using HTTP as an Access Method

SNMP should not be confused as being network management. Rather SNMP is merely an access method used by a management station to read and write items in an agent's MIB.

The myths of "The Collapsing Network" and "The Dumb Agent" have forestalled many efforts to consider a connection-oriented alternative to SNMP.

Today's Internet is successfully carrying an enormous transaction load using the World-wide Web's HTTP, which is a TCP-based protocol. HTTP transactions follow a very simple life-cycle:

1. Client creates a TCP connection to the server.
2. Client transmits an HTTP operation, usually a GET or a POST, to the server. Although both can be used to carry additional information from the client to the server, POST has no restrictions on the size or structure of that information.
3. The server responds with an HTTP header followed by a MIME-typed chunk of binary data of arbitrary size. This data may be literally anything that can be reduced to binary. It may be the familiar HTML of WEB pages, a JPEG image, or instructions to the browser how to launch an MBONE viewer.
4. The connection is closed.

HTTP's major shortcoming is that it doesn't do enough work per TCP connection. Efforts are underway to reduce this weakness.

One could readily conceive of a number of ways to encode MIB information in that chunk of binary data. It could be truly binary, with its own MIME type. Or it could be embedded in HTML as readily identifiable, machine parseable, structured comments.

One might think that the real issue with this approach is how to map `get`, `get-next`, `get-bulk`, and `set` onto this scheme.

However, the real issue is whether we really need the `get*` trinity at all. The `get` operation is the only silver-bullet of the three SNMP retrieval operations; the latter two are merely means to get past SNMP's limited data unit imposed by the myth of "The Collapsing Network". As such, all three retrieval operations could be collapsed into a single `get-subtree` operator that takes a single parameter, an object identifier, and returns all objects

which are prefixed by that OID. For convenience, we ought to define the subtree traversal to return the objects in lexicographic order; and, for efficiency, we should allow a list of prefixes and allow the return of multiple subtrees.

So, how would this actually be mechanized over HTTP?

Consider SNMP queries as the equivalent of a WWW form in which the user or management station simply lists the MIB objects it wants to obtain or set values into. The SNMP response would be the web page returned as a result of processing the input form.

For processing efficiency, this result need not be encoded in a way that could be directly presented to a human user. The data could be handled either by a special application which speaks HTTP or by a management plug-in to a WWW browser.

One very attractive feature about this approach is that it may be able to piggyback on those WWW security features which are falling into place.

The main drawback of this scheme is that it can be highly intensive in its use of TCP connections, but as has been mentioned, the WWW community is already facing and, hopefully, resolving this problem.

It has been argued by some that this approach would degenerate into a prodigious number of short TCP connections, each retrieving only a small number of MIB variables. This is a valid concern. It has also been argued that the gain offered by TCP is not so great when comparing with the `get-bulk` operator. This is true, however, `get-bulk` is not widely deployed (yet). And it merely changes the point at which the curve of TCP efficiency crosses that of SNMP efficiency; it does not change the fact that, as MIB retrieval size increases, TCP becomes more efficient than UDP-based SNMP.

Using long-lived SNMP Associations

Consider the proposition that there exists a long-term relationship between a management station and managed devices on the network.

In SNMP, this relationship is somewhat vague and tends to be indirectly visible as polling by managers (to determine ongoing device status), trap destination configuration in agents, and table management in RMON devices. In the various SNMPv2 proposals, this relationship was made manifest through the various administrative frameworks.

Why not go the next step to acknowledging the relationship and creating an explicit manager-agent "association"? This association would be composed of security and other state information and there would exist, whenever possible, an open transport connection between the manager and agent. (When that underlying transport connection fails, the two ends would attempt to reconstruct it and re-synchronize their association state.)

This approach vastly simplifies the issues of security--authentication and privacy exchanges would occur at association startup and would be cross-checked at important points in the association (typically in the form of a handshake when re-building transport connections and as cryptographic-checksums embedded as integrity checks in the various transactions crossing the association.)

This approach also obtains a significant performance improvement over today's SNMP when moving any significant amount of data. (With today's TCP protocol engines for small queries, however, there may be three or four packets crossing the net rather than the two for UDP based SNMP, although the comparative analysis can be rather complex and highly subject to packet loss rates and the TCP windowing and ACK behavior of a given TCP implementation.)

Scripting MIBs

Through the use of a MIB one can insert a script into a device, start its execution, poll for completion (or await a trap), and fetch the results.

One can imagine, for example, a script that monitors the variables in a device watching for tell-tale signs, such as a rapid increase in an error rate. The script could then either report the problem, trigger additional diagnostic tests, or take corrective action. (The latter two would require sophisticated scripts.) Scripts are ideal mechanisms to evaluate and act upon meta-variables, as described earlier.

Scripts are often expressed in a simple interpretive language. Each line of the script is simply a row in a table of octet string variables.

This is not a new idea: some years ago David Levy of SNMP Research published a "Script MIB" and the University of Delft allows a management station to inject Scheme language programs as RMON filters.

The real difficulties of all script approaches are not the scripts themselves or the language used (although, as one can expect, there are there are competing camps advocating TCL, Scheme, Java, Python, APL, RPG, Cobol, or French.)
The difficulties are these:

- *script security:* Can the script be kept within bounds? This is a difficult issue because, almost by definition, network management implies the exercise of discretionary control. If network management is to have the ability to make beneficial changes, it almost necessarily has the power to cause damage if misused.

- *script integrity:* One wants to be sure that the script being executed is actually the one intended. In Java, there are already authorities who will place their imprimatur on a script with the guarantee that "this script is safe" according to some criteria.

- *resource control:* Scripts are programs and as such they can consume memory and computing cycles and potentially other resources. How does one put a quota on a script?

- *script control:* A management station needs to be able to take control of an executing (or run-away) script. There needs to be a way to halt scripts.

- *script recovery:* A script can have a lifetime longer than the memory of the management station which started it. It is important that a management station can learn of the existence of scripts it created.

- *expressive power:* There is a great deal of room for differences of opinion regarding the fundamental actions which a script can invoke. Our own experience is that the primitives should be reasonably high level and should include the following:

  - ICMP ping (with control over packet size, packet contents, IP options, and retry intervals). This ping should capture round trip times, loss rates, data inconsistencies between what is sent and what is returned, and any ICMP unreachable messages. On multi-homed machines, the script should have control over which interface is used to send the packet.
  - Traceroute (with control over source routing, packet sizes, retry intervals, and maximum and minimum TTL).
  - Path MTU discovery.
  - DNS lookup tools.
  - SNMP operations.

- *script migration:* With a script MIB, the migration of scripts from one machine to another is not an issue, since the management station creating the script controls the migration.

- *Script debugging:* Scripts are programs and programs have bugs. Initially we can expect scripts to be fairly simple and amenable to simple debugging techniques. However, as scripts grow in complexity, we will need means to trace their execution, trap exceptional conditions, set breakpoints, and inspect script variables.

Initially we can expect scripts to be fairly simple: a good first step might simply be to watch what human managers do and use the scripts as simply macros for commonly executed sequences. Over time, as experience grows, scripts should grow in sophistication and, as we learn to trust them, given more power to take limited actions without asking for human permission first. This leads us to the next step:

Semi-Autonomous Area Managers

The notion of scripts opens up the possibility that one can design a system to delegate the monitoring and control tasks from a high level manager to subordinate "area managers" in close proximity to those devices that they are managing.

This is not a new idea. Professor Yechiam Yemini of SMARTS has been building tools using these techniques for many years. Java's popularity is extending this idea to areas other than network management.

The basic idea is "management by delegation", the superior level manager creates a script which it downloads into the area manager for execution. The area manager is, of course, a multi-threaded device and can execute many scripts simultaneously, perhaps on behalf of multiple superior managers.

An area manager would usually be given authority over devices with which it has inexpensive, low latency communications. One might conceive of an area manager's span of control as a single LAN or a group of LANs connected with a single high performance router.

The area manager might interact with the end devices using scripts, but it is far more likely that it would be done using traditional SNMP. One of the benefits of the proximity of the area manager and the ultimate devices is that the high bandwidth and presumably low packet loss rate would allow SNMP exchanges to be done rapidly and with minimal data distortion due to non-atomic snapshots of device tables.

An interesting possibility of area managers is that if they are equipped with out-of-band communications paths they can play a very useful role in network troubleshooting. Anyone who has ever repaired networks knows that you always need to be in at least two places at once. An area manager running a pre-loaded script can act as a troubleshooter's remote eyes and ears. For example, an area manager might be running a script which says:

> Watch the network traffic and routing protocols and periodically ping sites off the local net to confirm outside connectivity. Should outside connectivity fail, perform traceroutes and report the results using the out-of-band channel.

Network Management Worms

The term "worm" in networking comes from the John Brunner's book *The Shockwave Rider*, and refers to a program that moves about a network, from computer to computer. It has become a rather pejorative word due to the widely-reported Internet virus of November 3, 1988. However, worms are potentially very valuable. For example, many years ago at Xerox PARC there were worms that propagated through the facility's computers at night to perform diagnostics on otherwise non-busy workstations.

In terms of network management, worms are really scripts that can replicate and migrate. They are really just the next step in the continuum that begins with script MIBs. Some researchers in the network management community are already working with migratory programs. They can perform network device discovery using a worm that migrates through the network and sends a report back to a central logging address whenever the worm moves to a new machine.

Effective use of worms requires that they be "safe", that they have finite lifetimes and limited appetites for network and computing resources, and that they can themselves be located, managed, and terminated.

Summary and Conclusion

In this article we have illustrated a few ways that network management can become something better than it is today. We have taken a rather opinionated position, not because we believe we are right (although we hope we are), but rather to try to ignite new work in network management.

None of the ideas presented here are impossible. Any one could be developed and deployed within 12 months.

Some Experiences with Emerging Management Technologies

Barry Bruins
Cisco Systems

The hype about Web-based management is reaching a fever pitch, ant it is beginning to sound like HTTP is the solution to all problems. As network management professionals, we've been exposed to many technologies that were supposed to solve all our problems ("open" platforms, CMIP, DME, and so on). What we need to do is to cut through the hype and understand what role HTTP technology can play in network management and where we shouldn't throw the baby out with the bath water.

Web technology is being employed in both device management and network management applications. We will discuss each in turn.

But, before we begin, a bit of terminology is in order:

- *HTTP:* a fairly simple session protocol built upon TCP. It has some unfortunate security pitfalls such as sending the password in the clear in every request.
- *HTML:* a textual presentation protocol designed to direct presentation software in the appropriate on screen formatting.
- *Applets:* downloadable bundles of executable code that run in the browser's machine environment.
- *Web Servers:* software that supports HTTP connections and, typically, forks customized scripts to satisfy those requests.
- *URL:* the universal name space for the web that has been the key to efficient and easy to use navigation between information sources.

Using the Web for Device Management

To the author's knowledge, the first use of a Web browser to perform device management was in a Localtalk switch developed by Tribe Computer Works. They developed a simple, easy to use console interface. Devices that have a small number of parameters can use the features of HTML 2.0 to get input in simple forms and display device status in a simple textual manner.

Other companies have done this level of functionality as well. In the 1980's, nearly all network devices had console ports supporting VT100 style terminals. This was the ubiquitous terminal of the day. Most devices implemented a TCP stack, many times just to provide remote access to the console. In the 1990's, a web browser has replaced the VT100 as the ubiquitous terminal of choice. Telnet servers not even installed on many devices. This observation has fueled the drive for HTTP consoles in network devices. However, just as console interfaces were not a good candidate for standardization when VT100's were the chosen interface, these web interfaces are quite different and not good candidates today.

Using the Web for Management Applications

Web interfaces to network management applications are also becoming very popular. One of the first well publicized applications was done by the network operations team at the Stanford Linear Accelerator. The SLAC web

site allows public access to network performance graphs and other information. Many companies are using internal web sites to display network status and performance information to a wide audience. These approaches are often much more cost effective and provide much easier access to information than standard network management platforms. If you think about the cost of a Unix workstation running a commercial management platform, it's easy to see why this approach is popular. In fact, many vendors of management applications are moving as fast as they can to retrofit their applications for the Web. The network management platform of the future may only have Web-based user interfaces.

These are examples of how web technology has improved network management. But what's the next step? How far can we go? What impact does Java have?

Console Command Mapping

To explore these questions, I ask your indulgence while I discuss what has been done at Cisco Systems. This is only to give background on the observations.

Cisco began experimenting with embedding Web interfaces in routers in mid-1995. We wanted to give the maximum capability to a browser-based user interface with minimum impact to software image size. Initially, we didn't think it was possible to translate the entire Cisco command language to the new paradigm. However, after some experimentation, a scheme to use the same command parsing functions as the TTY interface to generate web pages was devised.

In internal data structures we had the names of the commands and the help strings to go with them. The HTTP code extracts those strings from the parse tables and formats them in HTML. Each command is turned into a link. For example, `show interface ethernet0` is turned into the URL:

http://routername/exec/show/interface/ethernet0/

This approach means that anyone with a web browser can issue a router command and the result will come back into the browser window as scrollable text.

We thought this was interesting and went off to solve other problems such as organizing these links into useful subsets, creating forms for specific functions, and so on. But, it was our customers that surprised us! We found that customers are building Web-based pages for help desk guides. As help desk personnel are trying to track down problems, they follow step-by-step procedures on web pages. Some of these steps on the web pages are actually links. It may say "click here to show router interface statistics." This would be much more straightforward than directing the user to telnet to the router and execute a command. Through the use of frames, the procedure can stay on the screen while the result of the procedure can show in another frame.

Having full device commands as web operations can greatly simplify automation of simple network operations tasks. If different links to different devices are listed in the series of web pages, then multiple devices can be woven together in the investigation. Keep in mind that the output that is given to these operations is designed to be human friendly, and not necessarily machine readable.

Java-based SNMP

When the Alpha release of Java was made available, several people thought they should make a Java version of SNMP. When asked why, the answer was invariably something like

> "If you build it they will come."

So a particularly persistent individual took the latest CMU code and started porting. When the first version came out, people came from all around to see SNMP running in Java. When this poor soul admitted that the varbind encoding routines were left in the C programming language, he was ridiculed. So he went back to work and finished the job. Next, fancy Java gauges were tied to the SNMP stack. When several gauges monitoring separate MIB objects were put up in a window for an Interop demo, the criticism continued:

> "You're going to clog the network with this polling if you have separate PDU's for each gauge."

So a scheme to make the SNMP class an applet in it's own right and use inter-applet communications was devised. This exercise shows that there is no end to the things that have to be redeveloped when a new paradigm comes along.

But we weren't out of the woods yet, there were two more problems. The first was that this was getting to be an awful lot of code to download to a browser every time you want to use it. Second was Java's network security paradigm that dictates that an applet can only open a socket to the server that the applet was served from. This has a couple of unfortunate implications. If we want to do SNMP to a device from a Java applet, that device must be sophisticated enough to be able to download an SNMP stack over HTTP. We changed the router to download applets from its flash file system, but that's only useful for demonstrations. Additionally, if we want to put up a browser with gauges monitoring two

different devices, I'd need two copies of the SNMP applet. One copy would have to come from each device.

SUN is interested in relieving these and some related implications, and it seems a more sophisticated applet security model is on the way. Regardless, some applets will have to be trusted more than others. I guess some will ship with "root privileges" *sarcasm*! These limitations are not present for Java applications that are installed in the local environment and run from the Java interpreter and not from a browser. So the benefit of platform portability is preserved while the benefit of dynamic loading of client software is not.

New Paradigms for Web-based Management

There are two new network management paradigms on the horizon. Both will use the Web to legitimize the adoption of their model. These new approaches are coming from both Sun and Microsoft. In each case, object models are asserted for network devices. These object models attempt to abstract useful properties and functions among similar devices. Once abstracted it will be much easier for programmers and automated tools to manage these objects. In the case of Sun, object abstractions are in Java. Sun is attempting to standardize the look and feel of applets that know how to manipulate these devices. Their gambit is to either, get devices managers to implement Java directly in the devices, or to have a proxy agent that will convert the new protocol to SNMP. The value is in the user interface objects that can be reused time and again. These objects can be combined on a web page with almost no effort.

In contrast, Microsoft addresses the effort required for their new paradigm with programmer tools that generate much of the code to use their object abstraction automatically. They recognize that when an engineer has a choice he won't want to write user interface code to the new model and then write proxy code to map the new model to SNMP. Most engineers will use SNMP directly so they can get their work done.

Network and system management nirvana seems to be stated something like this: if we can make all devices look alike, we can make management simple. The trouble is that even if vendors would like to convert all of their MIB's to some new object paradigm, the last thing they want is for their devices to look just like their competitors. Each vendor will insist on having a way for their competitive advantage to be manipulated and displayed. If you add a bell to the box, marketing will want to see the bell that they defined in the management system. Commonly useful object models will quickly decay from the baggage that is placed upon them.

The success of new paradigms is predicated on them providing significantly more value than the old paradigm. It remains to be seen if the power of the web can push these new paradigms over the top.

Why SNMP won't die

On paper, each new thing that comes along tries to address the weaknesses of the existing scheme. Sometimes it's a protocol weakness. Sometimes it's a new abstraction to make things simpler. The SNMP SMI is what saves SNMP. New approaches either try to get more rigorous to add capability (i.e., inheritance), or try to make it simpler (i.e., just use HTTP). What we've learned over the years is that MIB definitions serve us well as "simple to understand contracts" between device implementors and application implementors. Attempts to add rigor are met with resistance from device implementors that aren't being driven by the market for anything stronger. Attempts to "just use HTTP" will fail because the definitions of the interfaces won't be strong enough to do beyond flashy interfaces.

Better ideas than SNMP come and go. Most fail because they don't realize that SNMP represents a pretty nice balance between the possible and the practical.

Overview of a Web-based Agent

Patrick Mullaney
Cabletron Systems

This article is intended to give an overview of the implementation strategy for a basic Web-based management agent. It begins with an introduction to fundamental concepts involved in the implementation of a Web-based agent. Next, it examines some details encountered in the implementation of such an agent. The article concludes with a discussion of the advantages/disadvantages of this approach to managing a device.

A Brief Introduction to Web Technology

The Hypertext Transfer Protocol (HTTP) is the primary transfer protocol used by the World-wide Web. The HTTP model is an extremely simple one.

The typical transaction is one in which the client establishes a connection to the server, issues a request, and waits on a response from the server. The server upon receiving the request from the client, processes the client's request, sends a response, and then closes the connection.

HTTP Requests:

The most common request methods for HTTP are GET and POST. The GET method is used to retrieve a resource from the server. The POST method is used to send

information to a resource on the server. A resource is defined as "any data object or service".

An example of a GET method might be:

```
GET /ModGenConf.html HTTP/1.0
Accept: image/gif
Accept: image/jpg
Accept: image/x-xbitmap
User-Agent: Lean Mean BrowserBoy/ Strictly beta
If-Modified-Since: Monday, 1-Jan-96 19:04:09 GMT
```

while an example POST method might be:

```
POST /ModGenConf.html HTTP/1.0
Accept: image/gif
Accept: image/jpg
Accept: image/x-xbitmap
User-Agent: Lean Mean BrowserBoy /Strictly beta
DATE: 06%2F06%2F96&IP=134.141.48.201&SAVE=YES
```

A blank line is used to delimit each method.

HTTP Responses:

When the server receives a request from the client, it locates the resource indicated in the request and performs the method requested by the client on that resource and transmits the result back to the client. In the case of a GET method, the server simply forms a response by including the resource in the response. For the POST method, however the server passes the DATA included in the POST to the resource and receives a result to include in the response back from the resource.

Responses begin with a status line consisting of the protocol version of the server followed by a response status code, and an optional response status message, e.g., "HTTP/1.0 OK". Typical response codes and messages include:

- 200: Resource found
- 400: Unintelligible Request
- 401: Unauthorized Request
- 404: Requested Resource not found
- 405: Request method not supported
- 500: Unknown server error
- 503: Server capacity reached

Following the status line are one or more header/value pairs and then the actual data.

Similar to the request, the response then includes an optional list of header/value pairs. Two of these headers fields, `Content-Type` and `Content-Length`, indicate the media type and the length, respectively, of resource being returned. The media type is given as a MIME type. MIME is defined in RFC 1521, and can be used to identify any media encoding type including application-specific types.

Authentication:

HTTP uses a simple, extensible challenge-response authentication mechanism.

The server issues a authorization challenge by sending the client a 401 (unauthorized request) response. In the response, it identifies one or more supported authentication schemes along with whatever parameters are necessary for achieving authentication via that scheme. One of these parameters indicates a *realm*, or protection space, on the server. It allows the server's resources to be partitioned into multiple protected regions, each with it's own authentication scheme and/or authorization database.

The client, upon receiving a 401 response, can re-issue the request with a `Authentication` header that identifies the authentication scheme in use along with the necessary credentials to prove its identity for the realm in question.

RFC 1945 defines an authentication mechanism called "basic", which all clients are encouraged to support it. This model employs user-ID/password credentials for a realm.

The Hypertext Markup Language:

HTML is a platform-independent document description language. The language is a subset of SGML, which is a more elaborate ISO document standard. (This should sound familiar to SNMP people!) HTML was design to be used over low bandwidth communications links. Ironically, this makes it ideally suited for exporting formatted information from management agents, which are generally embedded systems. The general philosophy of the language is to not control every aspect of the display. Through the language, the page designer gives hints to the display station as to the layout of a page, it avoids the overhead of controlling every aspect the display down to the pixel level. This minimizes the amount of information that needs to be transferred to display a particular resource (i.e., a document). This lack of complexity gives HTML, regardless of the particular document, it's familiar "look and feel" and ease of operation.

Proper page design philosophy has been to minimize bandwidth used by minimizing the amount of information a document contains (image content in particular). This philosophy also benefits management agents, which generally don't have a large amount of persistent storage, by only requiring them to store a minimum of information at the agent.

HTML uses "markup" tags to denote regions of text as having specific characteristics. The tags serve as instructions to the browser on how to render a region of text. These tags are portions of text surrounded by the less-than (<) and greater-than (>) characters. For

example, the <B> tag indicates that the browser should bold the text following tag and a </B> indicates to the browser that the bolding should end. HTML provides tags for formatting of text, inclusion of graphic images, navigation to other documents (hyperlinks), standard form controls (text boxes, radio buttons, and so on).

The web-based agent implementation described within this article exports HTML formatted management documents to a standard web-browser. No specialized management station software is needed to use the agent. There are three issues to consider: document design, agent implementation, and authentication.

Document Design

Our web-based management agent has three distinct classes of documents: static, dynamic, and form.

The static document is a document whose contents never change. It can be built into the agent in the form of a static data structure, put in the agent's persistent storage, or referenced by the agent by using a URL naming a supporting device (a supporting server or another agent). Examples of static documents are graphic images, help/informational documents, and HTML documents used entirely for the user's navigation to other documents (via hyperlinks).

The next type of document is a dynamic document. This document's contents have the potential to change over time. The contents of a dynamic document are assembled at run-time upon a request for the document from the client. This is the most common type of document supported by our agent. It can be used for many purposes such as dynamically displaying the current values of statistics kept at the agent, and current values of user configurable operational parameters at the agent.

The final type of document is a form document. This type of document is used to modify the current operational mode of the device or agent. Form documents can be either static or dynamic themselves. The controls (e.g., textbox submission fields, two-state buttons, or multi-choice selection boxes) on forms often have current state values associated with them. These state values must be inserted into the control upon a request for the form document.

The layout of documents for our agent is done with the aid of a commercially-available HTML layout editor. This tool is augmented by an in-house developed tool which gives the document designer the ability to associate certain fields within a document with dynamic (or "live") data within the device and, in the case of form documents, supply an action method for each control element to be executed upon form submission.

Document and Agent Implementation

Once the HTML document has been designed, a code generator is run on the document to generate supporting code and data structures (as a C++ class). The supporting code consists of methods for registering the document with the agent, the document name method, the document serialization method, the document authentication realm method, and for a form document, a method for the processing of control elements submitted in a POST request. The registration method simply adds the document to a directory of documents maintained by the agent. This directory is simply a data structure used to hold and retrieve documents by name(Request-URI).

Upon a request from a client, the agent searches the directory using the Request-URI as a key to find a particular document.

For a `get` request, the document's serialization method is called when the agent receives a request for the document. The results of the serialization method are the current contents of the document. These results are returned as the data section of the HTTP response.

In the case of a POST request, the document's serialization routine is called after the document has processed the data section of the request. The document's control element parsing method separates the data section of the request into control/value pairs. Each pair represents the submitted value of a specific control on the form. Then, for each control, the parsing method calls a specific user-supplied method (defined in the document design step), passing the value associated with the control. The serialization results for a form after processing a POST method is completely up to the form designer. The results could simply be the form document with the new values used in the form submission, or be a document indicating the success or failure of the submission.

The agent itself can be configured to be single- or multi-threaded. The advantage of using multi-threaded server is primarily increased throughput. For many management implementations, a singly threaded server is fine. (Browsing a Web-based management agent isn't particularly exciting!)

Another advantage of a multi-threaded agent is that when used with persistent connections, the agent will be able to service more than one client at a time. Persistent connections are aimed at solving the latency problem with TCP slow-start. Persistent connections were added ad-hoc in HTTP 1.0 and will become standard in HTTP 1.1. The disadvantages are possible manager conflict and resource consumption issues.

Authentication

The agent uses the basic authentication scheme described in the previous section.

It uses two authentication realms (protection spaces) within the device, read-only and read-write. The agent maintains two passwords, one for read-only access and one for read-write access, which are stored in persistent storage on the device. (This is quite similar to SNMP, and on some our devices, the SNMP community strings will be used as the passwords). A read-only password only allows the client (browser) to access pages in the read-only realm. The read-write password allows access to both realms.

Each document in our agent is associated with one realm. If a client fails to authenticate itself with the agent for a particular document, the agent challenges the client with that document's realm (read-only or read-write). The agent can determine a document's realm using the document's authentication realm method, which simply returns a string indicating the realm. In our implementation, informational documents typically only require a read-only level of access, but may occasionally require read-write if they contain privileged information. Form documents are generally not accessible at the read-only level of access.

Now let's look at how this approach to Web-based management compares to the traditional SNMP approach.

Advantages over the SNMP Approach

The advantages of using HTTP to export HTML-based management screens from a device are many.

First, there is no specialized management software necessary to configure or monitor a device.

Second, the versioning problems that typically occur when an older agent or manager doesn't support the new and possibly required features of the other are eliminated. These problems occur when either a new MIB is supported at the agent and not supported (displayed) at the manager, or when a new version or feature of the manager expects a minimum of MIB support from an older agent. By using a Web-base agent to export screens directly, both agent and manager do not need to be updated simultaneously. This same problem is only exacerbated in a multiple vendor environment. Many vendors supply devices requiring some level of vendor specific MIB support. This complicates the job of the management station vendor by requiring them to support additional screens for these vendor specific MIBs. (MIB browsers are simply inadequate for this task!) Often, this leaves the network manager with no choice but to buy specific management applications for each vendor's device and then learn to use all of them. By delivering the management information via a Web-based agent, the application is delivered with the network device and the common familiar interface of a Web browser exists for all applications.

The next major advantage is that of platform-independence and location-independence for the application. A network manager can access his Web-controlled network from anywhere, on any platform. All that is needed is a general-purpose Web browser, which is standard equipment for the vast majority of platforms.

A final major advantage is seamless integration with on-line documentation. Context-Sensitive help and documentation may be accessed via hyperlinks embedded directly into the agent's management pages. Additionally, configuration and management can be driven entirely from an on-line instruction manual, in tutorial fashion.

Possible Future Advantages

Another area where HTTP could be applied is in the transport of unformatted management information between communicating entities. Because HTTP uses TCP, it can transfer large amounts of data in a single transaction. A management application layered on HTTP can transfer large amounts of data to/from the device without having to break up of data for the transport layer.

Also, security options for HTTP are becoming available. Security for commercial applications using HTTP over the Internet will force a solution in this area. A management application that uses HTTP can then leverage these security features. In addition, using one security mechanism between multiple applications would greatly ease the burden for network administrators who often have to configure security for all applications.

Disadvantages over the SNMP Approach

There are a few possible disadvantages and concerns with using this technology for network management applications.

The first issue is the latency of HTTP when used for small transactions. This is due to the protocol being layered over TCP. This has been raised as an issue against using HTTP as the transport for generic MIB information (instrumentation at the agent). Latency for HTTP is an issue for ALL HTTP transactions, not just in this specific case. The HTTP community is aware of the problem and are working toward solutions. One solution is to use persistent connections, and thus amortize connection overhead over multiple connections. The theory is that if you want one resource from a server, you are probably going to want another.

The next issue of concern applies mainly to the approach of exporting formatted management information from agents. The concern to this approach would be that the burden of display is placed on the agent. This would be a valid concern for only the "lowest" of low end devices. This overhead is fairly minimal and varies based on the desired complexity of the documents exported from the agent. It primarily manifests itself as additional memory overhead (for document formatting) on most agents.

The last issue of concern is that HTTP requires a TCP implementation. Many devices already support other applications that require TCP, e.g., telnet, so this will not be an issue for them. In fact, for those implementations with existing telnet support, an HTTP/HTML implementation could replace the telnet functionality, and thus require little or no extra memory overhead.

In Conclusion

By now, it should be clear that SNMP and HTTP share a number of similarities.

Both provide the client (manager) the ability to modify or retrieve specific resources within a server (agent). SNMP provides a standard mechanism for identifying resources and a standard representation of those resources (a media type). HTTP provides the ability to identify and transport resources of any media type between communicating entities.

Devices that support both SNMP and HTTP will duplicate functionality. This is why it may be desirable to have an HTTP/SNMP interoperability standard. Interoperability with SNMP could be a very simple matter in my opinion. It could be as simple as defining a new MIME type for SNMP operations. This type could be the current encoding standard for SNMP, BER, a newly defined encoding standard, or both. Current SNMP operations over HTTP could then take advantage of the protocol's ability to transfer large amounts of information, and a new `get-subtree` operation could also be defined to explicitly take advantage of this feature, as is currently under discussion on the SNMP mailing list <snmp-request@psi.com>.

The CyberAgent Framework

Ray Burns, Mary Quinn
FTP Software

As operating systems vendors embed the platform-independent Java language into their kernels, new opportunities for distributed network management arise. Java applications and applets written for particular management tasks can easily be run from a variety of platforms. While this is an improvement over the os-dependent applications which exist now, these types of applications rely on the SNMP protocol and are limited in the types of functions they can perform.

CyberAgent is FTP Software's intelligent agent technology based upon Sun's Java programming language. An intelligent agent is a mobile, independent piece of software that travels the network and accomplishes tasks remotely. CyberAgent classes supplement the Java Development Kit and provide templates for agent development.

CyberAgent technology provides a remote execution capability that, when integrated into management applications, may go a step beyond SNMP-based management applications. Using intelligent agent technology, you can write agents for specialized network management purposes.

This article discusses the CyberAgent framework, agent development in network management and how you can write enhanced SNMP management applications using the CyberAgent technology.

CyberAgent Features

There are for key features of the CyberAgent Framework:

- mobility among TCP/IP connected computers;
- platform-independent among Java-enabled computers;
- choice of security levels; and,
- ability to carry data and programs.

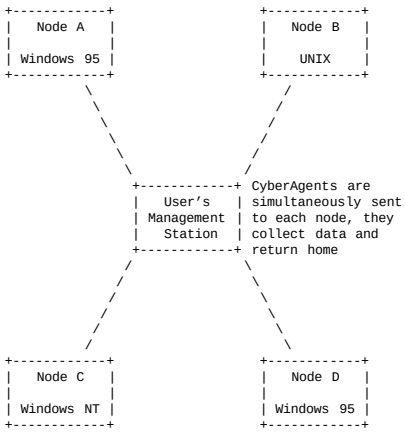
CyberAgents are mobile agents and the infrastructure that supports their mobility. A CyberAgent is a mobile, autonomous program written in the Java programming language. Application Programming Interfaces (APIs) transform an arbitrary Java application into a platform-independent CyberAgent that travels from computer to computer within TCP/IP networks under the sender's control. CyberAgents can also deliver and execute platform-dependent programs within a choice of security models.

Mobility among TCP/IP connected computers

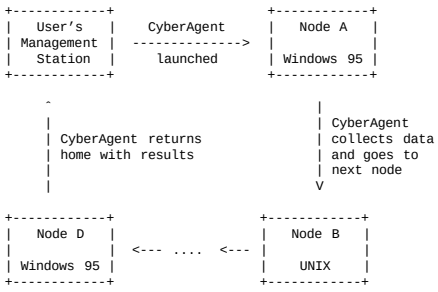
CyberAgents move between computers on a network to perform their tasks. Within limits imposed by the infrastructure and the chosen security model, CyberAgents decide when and where to move. With one exception, CyberAgents may only visit computers on their destination list, a list of computer user and machine names (or IP addresses) defined when a CyberAgent is launched. In addition, CyberAgents can visit the "home" computer that the CyberAgent was launched from.

Another factor governing CyberAgent movement is the travel plan. A travel plan describes the general mode that CyberAgents visit computers on the destination list and, like the destination list, is defined when a CyberAgent is launched. You can choose between two travel plans at launch time: *radial* or *sequential*.

In a radial travel plan, a CyberAgent appears to travel from the home node to each of its destination nodes simultaneously and then return home. Actually, multiple copies of the CyberAgent are dispatched in rapid succession, one to each destination node. If so programmed, each of the CyberAgent copies may then return to the home computer.



In a sequential travel plan, a CyberAgent is launched from the home node and travels to node A where it performs a task and maybe collects some data. After completing its task on node A, it continues its journey to node B and performs its task on node B. After completing its task on node B, it repeats the process of traveling to the next node and performing its task until it reaches node D, the CyberAgent's last stop. Then, if so programmed, the CyberAgent returns its data (and even itself) to home. The sequential travel plan is sometimes described as a store-and-forward plan.



The exact sequence of computers visited can be altered by the CyberAgent under program control. While restricted to the destination list, the CyberAgent can move at will among those computers, collecting information, depositing data in local files, executing Java and non-Java programs, sending data home, and performing any function that a program confined to a single computer can perform.

Asynchronism:
CyberAgents may move from one computer to another without waiting for tasks started on the first computer to complete. Consequently, multiple copies of the same CyberAgent may be in various stages of execution at multiple different computers in a network. Each CyberAgent copy executes completely independently of any other copy, asynchronously generating data as programmed. In this way, a single CyberAgent can harness the power of every computer on the network.

While this feature is powerful, it raises the issue of coordination between CyberAgents and the information they may collect, especially when multiple CyberAgents (or multiple copies of the same CyberAgent) arrive at the same computer. CyberAgents communicate (and coordinate) through CyberAgent messages which may contain any native Java object. Information collected asynchronously may be aggregated into a single file at the home computer through synchronized file access methods that are transparent to the CyberAgent programmer.

Control Flow Structures:
New control flow structures are required to exploit the power that mobility and asynchronism provide. Fixed (non-mobile) programs use `for` and `while` control flow structures routinely to execute essentially the same set of instructions many times. Similar control flow structures for CyberAgents execute essentially the same set of instructions independently on many computers.

Mobility control flow parallels control flow for fixed programs; both have initialization, iteration, and termination phases. Mobility initialization occurs prior to CyberAgent launch, iteration is the code executed at each computer on the destination list and termination occurs after the CyberAgent returns home. Each of these phases is implemented as a CyberAgent Application Program Interface (API) method.

A CyberAgent controls its own movements by determining where it is, setting its target destination (where it is to go next) and then sending itself to the target destination. The normal case is to go to the next destination on the list. Alternately, the CyberAgent may decide to go to any other destination on the list or even to break out of the destination list and send itself home. Sending itself home is analogous to a `break` statement in a fixed control flow structure.

Platform-Independence

CyberAgents can be sent to and execute on any computer on a TCP/IP network with a Java runtime environment and CyberAgent classes. Java runtime environments (essentially a Java interpreter and class loader) are available for many different platforms and operating systems. In particular, Java runtime environments are available for most Unix systems along with Windows NT and Windows 95. Because many computers on TCP/IP networks support one of the above operating systems, Java applications on these networks are largely platform-independent. CyberAgents inherit platform independence from Java.

Choice of Security Levels

Normally, network security comes into play when we want to execute, read or retrieve something from a server. Network security protects the resource (program, information or file) from unauthorized access. CyberAgents introduce new security requirements: instead of initiating the use of a resource from a server (pulling), a CyberAgent may be sent (pushed) to a computer bringing programs or data without active involvement. Most users would like to restrict such access to trusted users. Upon arrival of a CyberAgent at a computer, the user has legitimate concerns about authentication (is this CyberAgent from a trusted source), integrity (has this CyberAgent been altered), and privacy (can anyone else access the contents of this CyberAgent). To address these concerns, CyberAgents offer a choice of three security modes: *clear*, *password*, and *encrypted* using Data Encryption Standard (DES) or RC2 encryption.

In clear mode, CyberAgents are not password protected or encrypted and all CyberAgents that arrive are accepted without any attempt at authentication or integrity checking (appropriate if everyone on your network is trusted).

In password security mode, CyberAgents are not encrypted but only CyberAgents signed with a password shared by you and the sender are accepted. Password security mode provides a degree of authentication and integrity verification. Prior to launch, the password is hashed with the CyberAgent's contents to generate the password signature or Message Authentication Code (MAC). When the CyberAgent arrives at its destination, the recipient uses the common password to recover the MAC from the CyberAgent signature. The recovered MAC is then compared to a new MAC computed on the CyberAgent's contents. Successful signature decoding and MAC comparison indicates that the CyberAgent did indeed arrive from a trusted source and was not altered in transit.

In DES security mode, CyberAgents are both encrypted and signed with a key (known to the recipient and the sender). The DES digital signature provides strong authentication and integrity assurance. Prior to launch, the sender encrypts the CyberAgent with the private key. Then the key is hashed with the CyberAgent's contents to generate the digital signature or MAC. When the CyberAgent arrives at its destination, the recipient uses the private key to decrypt the CyberAgent and to recover the MAC from the CyberAgent signature. The recovered MAC is then compared to a new MAC computed on the CyberAgent's contents. Encryption ensures that privacy is protected and successful signature decoding and MAC comparison indicates that the CyberAgent did indeed come from a trusted source and was not altered in transit. RC2 encryption is similar to DES and is available for customers outside the United States and Canada.

Each of the security modes can be applied to CyberAgents from individuals or from a member of a community. A community shares the same password or key among all its members and is more convenient than having a separate password or key for every individual with whom you want to exchange CyberAgents.

Ability to Carry Data and Programs

CyberAgents may carry data and programs with them to their destinations. Both data and programs are carried as appendages to the CyberAgent itself, but still protected by the selected security mode. Programs are not limited to the Java language; they may be in Java or may be platform-dependent: each may be executed with a CyberAgent API. A CyberAgent may carry several platform-dependent versions of a program on its journey and execute the appropriate version depending on the platform it lands on.

Java Runtime Requirement

A Java runtime environment and CyberAgent classes must reside on each computer in a network that receives and executes CyberAgents. Most major operating systems vendors have made public commitments to embedding Java runtime into their operating systems, so this requirement may be short-lived.

Performance Limitations

Agent execution speed depends primarily on the Java interpreter and the hardware platform. Assuming the same platform for speed comparisons, the current Java interpreter (version 1.02) executing Java code is slower than executing compiled native C language code by a factor of about 30. However, several vendors offer

just-in-time Java interpreters that mitigate or remove this limitation, depending on how iterative the Java application (or CyberAgent) is. Just-in-time compilers accelerate execution by retaining the native instructions generated by the interpreter and re-using the native instructions the next time the same code sequence is encountered. So if a CyberAgent spends most of its time in the iteration phase (the normal case), then it benefits greatly from just-in-time interpretation.

Applications

Configuration and performance management are two areas of network management where CyberAgents seem particularly well suited. Data must be collected from a device in both areas. The Java environment and CyberAgent classes provide access to all information available on a machine, whether provided by the operating system, application programs, a DMI Management Interface or even an SNMP Agent. The local retrieval mechanism can be determined by a CyberAgent based on the hardware device operating system. CyberAgents provide an alternative transport and data collection mechanism to SNMP, reaching information not available within the SNMP framework and keeping the data collected confidential.

Even SNMP-instrumented data may be impossible to access in certain cases. For example, because no industry-standard extensible agent protocol exists, software from multiple vendors sometimes cannot coexist on a device. In these cases, each vendor provides an SNMP agent for managing their product. Due to SNMP port contention, the user is forced to choose one agent and must forgo remote access to data normally available via the others. Using a CyberAgent, this data may become remotely accessible if an alternate local method for retrieval of the information is provided by the vendor.

CyberAgents can be used to enhance SNMP-based fault detection and management applications. These applications can detect error conditions either actively (such as by device polling), or passively (such as by receiving traps and informs from the devices). The information is passed to the network administrator either visually, by sounding a bell or via e-mail. In general, however, notification that a condition exists is the most that the application can do. Sometimes, these conditions are time-critical and cannot be diagnosed or rectified after the fact. Historic logging of data indicates that problems occurred but there is no way to reconstruct the conditions that caused them. Enterprise-specific trap handling is an application for CyberAgents. In the case where the device is sending notification of an internal problem, an agent can be dispatched to gather more detailed information about the state of the system and return it to the manager.

CyberAgents can be used as components in a system to monitor (and possibly resolve) IP addresses conflicts. CyberAgents can periodically update a centralized network database of IP addresses in use and detect when a conflict occurs and determine which user is authorized to use the address. Another CyberAgent can then notify the unauthorized party to change their address. In some cases, depending on network stack vendor and operating system, a CyberAgent may even replace the duplicate with an unassigned address.

CyberAgents are also good candidates to perform software distribution and updates. A CyberAgent can carry a software installation package to target computers, copy the package to the target computer, notify the user that new software files are available and prompt the user to initiate installation.

In Conclusion

Applications for CyberAgents are just beginning to be explored. CyberAgents and Java open up new opportunities for managing networks of devices because of Java's platform independence and CyberAgent mobility. CyberAgent mobility expands the scope of device control and monitoring because a CyberAgent has local access to all data and controls on a device. CyberAgents can be key players in network management, either stand-alone or as components of SNMP-based management applications.

The SNMP Framework

Keith McCloghrie
Cisco Systems

In the last two issues, this column examined most of the SNMPv2 changes which are incorporated into the latest RFCs. Specifically, in the previous issues, we examined the changes to the SNMPv2c administrative framework (RFC 1901), to the SNMPv2 protocol (RFC 1905) and transport mappings (RFC 1906), and to the SMI (RFC 1902) Textual Conventions (RFC 1903) and Conformance Statements (RFC 1904). In this issue, we'll cover the remaining changes: to the SNMPv2 MIB and to the Coexistence document.

Changes to the SNMPv2 MIB

RFC 1907 has a number of changes from RFC 1450:

- The `system` group from MIB-II is now included in this MIB.

- References to the Party MIB (RFC 1447) have been removed.
- The SNMP-statistics counters defined in RFC 1450 (`snmpStatsPkts`, and so on) are deleted because many of them were defined through references to the administrative framework. They are replaced by including the snmp group in MIB-II. However, only some of these objects from the snmp group are defined with a status of `current`: `snmpInPkts`, `snmpInBadVersions`, `snmpInASNParseErrs`, and `snmpEnableAuthenTraps`; the rest of the snmp group from MIB-II is obsoleted.
- Two new objects are defined: `snmpSilentDrops` which counts the number of received requests which were silently dropped because the appropriate response message was too large to transmit; and `snmpProxyDrops` which counts the number of requests received by a proxy agent because a proxy agent failed to forward them successfully.
- The Object Resources (`sysOR`) group is moved to be under the same subtree as the `system` group, with `sysORID` now specifically defined as being the value of an invocation of the AGENT CAPABILITIES macro. A new object is also defined within the `sysORTable`: `sysORUpTime` which is the timestamp of when the resources corresponding to a `sysORID` value were last instantiated.
- The counter of the number of traps sent to a particular destination is deleted (since it referenced the administrative framework).
- the definitions of the `linkUp` and `linkDown` traps are deleted (since they are included in RFC 1573).
- a NOTIFICATION-GROUP containing `coldStart` and `authenticationFailure` is defined. The conformance statement requires implementation of this group.

Changes to the SNMPv1/v2 Co-existence

RFC 1908 defines a number of changes from RFC 1452:

- - There are two changes to the procedure for updating an SNMPv1 MIB to conform to the SNMPv2 SMI:
  - the MAX-ACCESS of an auxiliary object no longer needs to be changed to `not-accessible`.
  - an object defined as an enumerated INTEGER with a zero-valued enumeration does not now need to be obsoleted.
- When updating a definition of an SNMPv1 TRAP (other than the original six standard traps) to an SNMPv2 Notification, the OBJECT-IDENTIFIER name of the notification is formed by extending the OID of the Trap's ENTERPRISE clause with two sub-identifiers; the first extra sub-id is zero, and the second is the integer value of the trap's definition. This insertion of a zero sub-identifier is required so that the same notification is generated by an SNMPv2 implementation of a converted definition of an SNMPv1 trap as is produced by the conversion by a proxy agent of the same SNMPv1 trap.
- An INCLUDES clause (e.g., in an invocation of an AGENT CAPABILITIES macro) is allowed to reference an OBJECT IDENTIFIER subtree defined in an SNMPv1 MIB; such a reference is defined as including all mandatory leaf objects underneath the referenced subtree.
- The destination to which a proxy agent sends traps is defined as implementation-specific (rather than being defined by the administrative framework).

Frequently Asked Questions

Kaj Tesink
Bell Communications Research

Back to Basics.

Q: MIB Browsers v. Compilers

Could someone please explain me what is the difference between a MIB Browser and a MIB Compiler?

A: Response:

Tom Georges provided the most concise answer:

A MIB Browser is an application that allows one to view the available variables in an SNMP agent's MIB by issuing a `get` request, (or possibly a series of `get-next` requests. Most are terse and low level, meaning that they convey the raw information available from the device and make no decisions on the data, which a network management application would do.

A MIB Compiler is a program or script that converts a MIB file in ASN.1 notation (format of a MIB definition) to a file that can be read and interpreted by a specific network management application. A MIB compiler may also be used in the generation of SNMP agent code fragments and/or structures. Some network management applications do not use a MIB compiler but parse the ASN.1 file directly.

Q: Use of OIDs

Are there any requirements regarding the use of OIDs? For example:

1. A valid OID must start with the first subid to be 0, 1 or 2. If the first subid is 0 or 1, then the second subid MUST be in the range of 0-39 inclusive.
2. I assume any objects to be managed via SNMP must be registered within the `internet` subtree, so MIBs to be monitored will start with OIDs of 1.3.6.1 and the 5th subid will be in the range of 1-6 inclusive (assuming SNMPv2 is accepted).
3. I thought, that any MIB has to be registered under `internet` somewhere. Same like with `sysObjectID` and such, they will be registered under enterprises somewhere, right?

A: Response:

1. Right.
2. Wrong. It can be any OID. SNMP doesn't care. Your code shouldn't either.
3. Wrong. It can be any OID. SNMP doesn't care. Your code shouldn't either.

Q: Use of OPAQUEs

I'm still confused about this Opaque data type. Should I use it?

A: Response:

No, no, no. Opaque once seemed like a good idea but was quickly dreaded by implementors. It is still around for backward compatibility reasons but new use is prohibited. RFC 1902 explicitly states that "The Opaque type ... shall not be used for newly-defined object types." Consequently, kzm replies to an Opaque usage proposal:

> "In particular, you claim that the Opaque solution can be implemented by method routines without changing the agent's protocol engine, and by applications without changing the platform's protocol engine. But even if that is true, it is only because you are having the method routine and/or application take over part of the functionality of the protocol engine, i.e., you are advocating that both method routines and management applications implement their own ASN.1 encoding/decoding functions. In fact, it is precisely because Opaque requires this kludge that the SMI prohibits any further use of Opaque. If it wasn't for backward compatibility, Opaque would not be defined in RFC 1902."

Industry Comment

Marshall T. Rose
Dover Beach Consulting

In terms of failures, the inability of the SNMP community to standardize on an administrative framework runs a poor second to its failure to move beyond "instrumentation" in products to "solutions" in the marketplace. However, developments in Internet technologies other than SNMP now give us the opportunity to re-think device management, the way in which operators interact with networking devices, such as a router, hub, or bridge, (or a logical entity comprised of many physical entities with a single management interface).

This issue of ***The Simple Times*** may be controversial in that it provides a forum for SNMP friend and foe to suggest and analyze emerging management technologies and their relationship to SNMP. Even so, it is important to explore these topics because SNMP and these emerging technologies will have to become good, life-time neighbors. As such, we should understand the synergy between them, both to allow SNMP to prosper, and to ensure that the new technologies aren't used beyond their limits.

In this editorial, we'll focus primarily on the marketplace to understand one of SNMP (few!) failures, and the opportunities which have arisen.

Incentives in the Marketplace

In any market environment, one can understand the outcome by studying the players and their incentives. At present there are three sets of players in the networking management market:

- agent/NMS developers;
- device/application vendors; and,
- operators.

The Internet-standard approach to networking management has been to achieve a balance between the sets of players and between the players in each set. For example, SNMP represents a "shared contract" between an agent and a management station as to how instrumentation is to be defined and interpreted. The market has achieved a steady-state under these balances, but unfortunately the outcome is less than optimal – lots of instrumentation is available to manage, but few management solutions are available to operators.

The reason is due to how value is perceived and how it is actually added. At present, device vendors receive little benefit for their efforts at making their products manageable – they can add only instrumentation. In

order for the customer to perceive the value, the device vendor must rely on NMS developers and application vendors. This makes it difficult for a device vendor to differentiate themselves in their segment.

Further, standardized instrumentation is only defined for those aspects which are common to most devices. With every device having some proprietary functionality and most protocol standards being interface specifications, rather than functional specifications, full instrumentation, especially for configuration, always requires some aspects which are implementation-specific. This presents an insolvable problem for application vendors who must focus on standardized instrumentation in order to appeal to all device vendors in a segment. Thus, the operators are provided at best with incomplete solutions.

In brief, there is no longer any incentive towards innovation in device management, there is only an incentive to standardize "more MIB variables". We've done a good job at this. (Although one might observe that the number of new MIB modules has declined in recent years, suggesting that this too is running out of steam.)

Regardless of the breadth, width, and quality of standards defining instrumentation, a failure to achieve solutions in the marketplace is a failure of standardization. Does this mean that SNMP is a failure? No, of course not. But clearly, the current arrangement has achieved steady-state for device management, and, by itself, SNMP has proven inadequate.

So, we need to change the incentives! Note that there are only two first-tier players: device vendors and operators. Our solution must enable their roles to have a greater influence on the market. To do this, we'll take advantage of an unrelated development in Internet technology.

A New Model for Device Management: DM/W3

The latest client/server craze is called the *intranet* and it refers to the practice of defining a network service using HTML and HTTP as the client/server interface. Provisioning a service involves writing server-side applications without doing client development – clients are simply Web browsers. While one can argue about the technical merits of the Web protocols, one can not argue with the success they have achieved in the marketplace, for good or ill.

At present the client/server model for device management divides its tasks as:

- the server is an agent running on a device;
- the client is a management application running on an NMS;
- the "control" rules are defined by SNMP; and,
- the "data" rules are defined by the instrumentation.

Suppose we were to co-locate an HTTP server with an SNMP agent, and then provide a scripting language that executes on the device to manipulate the instrumentation and generate HTML. The DM/W3 model divides its tasks as:

- the server is an HTTP daemon on the device;
- the client is a Web browser;
- the "control" rules are management applications residing on the device; and,
- the "data" rules are defined by HTML.

We could define URLs which take the form:

http://device/application

where `device` is the domain name (or IP-address) of the device and `application`, if present, referred to an script available on the device. (If `application` wasn't specified, a page would be returned that specified the scripts available.) In most cases, a form would be returned asking for the parameters with which to run the application.

This clearly provides a non-SNMP framework, and we'll use this fact to our advantage. For example, here's how we'll dispose of the usual administrative and security problems:

- authentication: up to the HTTP server;
- security services (message integrity, replay protection, privacy): use either SSL or S-HTTP; and,
- authorization: up to the application based on the authentication and security services.

Our point is not to try and export an SNMP administrative framework through either the Web or SSL, rather to employ a parallel framework.

A New Set of Incentives

To understand the value of this approach, consider how the incentives for device management have changed. Because applications run server-side (on the device), device vendors now control applications which get written for a device.

This control now gives them an incentive to differentiate their products from others in the same segment. Use of Web technology provides an open interface to

vendor-proprietary functionality. As such, device vendors are rewarded when they spend more time on adding management to their products.

For example, a device vendor might write one or two dozen different management applications which are carefully tailored to exploit the special features of the device. Ease-of-use, functionality, performance, and other issues can be high-lighted. When an application completes, the page returned can contain links for further information, such as invoking related applications with preset parameters, or fetching help information, and so on.

No longer is support for network management simply a checklist item for device vendors – it becomes an important consideration for decision-makers. Indeed, operators will now wield their purchasing power based on smoking guns, not finger-pointing. If a vendor's device lacks management solutions, the operator needs blame only the device vendor, not the NMS developers or application vendors. However, operators will now take on a new responsibility – they will have to clearly express the kinds of functionality they expect to see in the devices now that the middlemen aren't clouding the issue.

Device Management Revisited

SNMP will still be used for device management – by enterprise management applications running on an NMS – to gather information, respond to notifications, and so on. While it appears to be hopeless to use standardized instrumentation to effectively management individual devices, this instrumentation is valuable, precisely because it does smooth over the differences between devices in a heterogeneous network.

How does DM/W3 impact the other players besides those in the first-tier? Agent developers will still provide SNMP engines to device vendors, along with a low-impact HTTP server and a device-side scripting language. But, NMS developers will need to integrate their products with a Web browers. Although many vendors working on applications for device management will have to become more involved with device vendors, things will probably be easier for client-side application vendors as they switch from windowing protocols to HTML programming.

The Role of Standardization for DM/W3

Although the Web protocols have achieved success in the marketplace, HTTP is technically sub-standard in comparison to other Internet protocols. (Don't blame the designer, HTTP wasn't designed to operate in today's Internet.) So, over time a new HTTP-like protocol should emerge, one which honors the first principles of SNMP and is thereby server-friendly. True, an HTTP server won't be as lightweight as an SNMP agent, true it uses TCP instead of UDP, and so on; even so, it is important to optimize HTTP to be widely-scalable, and this is why adherence to the SNMP design philosophy, wherever possible, is critical. Naturally, this new protocol isn't specific to device management, all Web-based applications could make use of it.

I predict that two areas in the IETF will see considerable change. The Operational Requirements area will find itself the center of activity in specifying functional requirements (the "what", not the "how"). Further, the Network Management area will likely be disbanded: each of its few remaining ongoing instrumentation efforts can be defined in the appropriate area of the IETF, and the remaining standardization efforts, those dealing with the "internal organization of the SNMP layer" (IOSL) such as v2 security, agent extensibility, rmon/entity mib, etc., are probably best put in the Applications area.

While it might seem strange to suggest that the NM area will eventually go away, this is perhaps inevitable. There simply isn't much SNMP innovation going on these days and we should consider it a stable, mature protocol in the Internet suite, which really doesn't merit an area of its own. Management issues are best left to each of the individual areas. As a final argument, consider that all that's left in the NM area is the IOSL-related working groups. If there wasn't an NM area today, it would be hard to argue that those working groups merit their own area. As such, if makes sense to simply move the IOSL-related work to the Applications area.

Standards Summary

SNMPv1 Framework

Consult the latest version of *Internet Official Protocol Standards*. As of this writing, the latest version is RFC 1920.
Full Standards:

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

Proposed Standards:

- RFC 1418 - SNMP over OSI;

- RFC 1419 - SNMP over AppleTalk; and,
- RFC 1420 - SNMP over IPX.

SNMPv2 Framework

Draft Standards:

- RFC 1902 - SMI for SNMPv2;
- RFC 1903 - Textual Conventions for SNMPv2;
- RFC 1904 - Conformance Statements for SNMPv2;
- RFC 1905 - Protocol Operations for SNMPv2;
- RFC 1906 - Transport Mappings for SNMPv2;
- RFC 1907 - MIB for SNMPv2; and,
- RFC 1908 - Coexistence between SNMPv1 and SNMPv2.

Experimental:

- RFC 1901 - Introduction to Community-based SNMPv2;
- RFC 1909 - An Administrative Infrastructure for SNMPv2; and,
- RFC 1910 - User-based Security Model for SNMPv2.

MIB Modules

An *unofficial* index of IETF MIB modules is available.
`http://www.simple-times.org/pub/simple-times/html/`
Full Standards:

- RFC 1213 - Management Information Base (MIB-II); and,
- RFC 1643 - Ether-Like Interface Type (SNMPv1).

Draft Standards:

- RFC 1493 - Bridge MIB;
- RFC 1516 - IEEE 802.3 Repeater MIB;
- RFC 1559 - DECnet phase IV MIB;
- RFC 1657 - BGP version 4 MIB;
- RFC 1658 - Character Device MIB;
- RFC 1659 - RS-232 Interface Type MIB;
- RFC 1660 - Parallel Printer Interface Type MIB;
- RFC 1694 - SMDS Interface Protocol (SIP) Interface Type MIB;
- RFC 1724 - RIP version 2 MIB;
- RFC 1742 - AppleTalk MIB;
- RFC 1748 - IEEE 802.5 Token Ring Interface Type MIB;
- RFC 1757 - Remote Network Monitoring MIB; and,
- RFC 1850 - OSPF version 2 MIB.

Proposed Standards:

- RFC 1285 - FDDI Interface Type (SMT 6.2) MIB;
- RFC 1315 - Frame Relay DTE Interface Type MIB;
- RFC 1354 - IP Forwarding Table MIB;
- RFC 1381 - X.25 LAPB MIB;
- RFC 1382 - X.25 PLP MIB;
- RFC 1406 - DS1/E1 Interface Type MIB;
- RFC 1407 - DS3/E3 Interface Type MIB;
- RFC 1414 - Identification MIB;
- RFC 1461 - Multiprotocol Interconnect over X.25 MIB;
- RFC 1471 - PPP Link Control Protocol (LCP) MIB;
- RFC 1472 - PPP Security Protocols MIB;
- RFC 1473 - PPP IP Network Control Protocol MIB;
- RFC 1474 - PPP Bridge Network Control Protocol MIB;
- RFC 1512 - FDDI Interface Type (SMT 7.3) MIB;
- RFC 1513 - Token Ring Extensions to RMON MIB;
- RFC 1514 - Host Resources MIB;
- RFC 1515 - IEEE 802.3 Medium Attachment Unit (MAU) MIB;
- RFC 1525 - Source Routing Bridge MIB;
- RFC 1565 - Network Services Monitoring MIB;
- RFC 1566 - Mail Monitoring MIB;
- RFC 1567 - X.500 Directory Monitoring MIB;
- RFC 1573 - Evolution of the Interfaces Group of MIB-II;
- RFC 1595 - SONET/SDH Interface Type MIB;
- RFC 1604 - Frame Relay Service MIB;

- RFC 1611 - DNS Server MIB;
- RFC 1612 - DNS Resolver MIB;
- RFC 1628 - Uninterruptible Power Supply MIB;
- RFC 1650 - Ether-Like Interface Type (SNMPv2);
- RFC 1666 - SNA NAU MIB;
- RFC 1695 - ATM MIB;
- RFC 1696 - Modem MIB;
- RFC 1697 - Relational Database Management System MIB;
- RFC 1747 - SNA DLC MIB;
- RFC 1749 - 802.5 Station Source Routing MIB; and,
- RFC 1759 - Printer MIB.

Experimental:

- RFC 1187 - Bulk table retrieval with the SNMP;
- RFC 1224 - Techniques for managing asynchronously generated alerts;
- RFC 1238 - CLNS MIB; and,
- RFC 1592 - SNMP Distributed Program Interface (SNMP-DPI); and,
- RFC 1792 - TCP/IPX Connection MIB Specification.

Informational:

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

Historic:

- RFC 1156 - Management Information Base (MIB-I) (see RFC 1213);
- RFC 1161 - SNMP over OSI (see RFC 1418);
- RFC 1227 - SNMP MUX protocol and MIB;
- RFC 1228 - SNMP Distributed Program Interface (SNMP-DPI) (see RFC 1592);
- RFC 1229 - Extensions to the generic-interface MIB (see RFC 1573);
- RFC 1230 - IEEE 802.4 Token Bus Interface Type MIB;
- RFC 1231 - IEEE 802.5 Token Ring Interface Type MIB (see RFC 1748);
- RFC 1232 - DS1 Interface Type MIB (see RFC 1406);
- RFC 1233 - DS3 Interface Type MIB (see RFC 1407);
- RFC 1239 - Reassignment of experimental MIBs to standard MIBs;
- RFC 1243 - AppleTalk MIB (see RFC 1742);
- RFC 1248 - OSPF version 2 MIB (see RFC 1252);
- RFC 1252 - OSPF version 2 MIB (see RFC 1850);
- RFC 1253 - OSPF version 2 MIB (see RFC 1850);
- RFC 1269 - BGP version 3 MIB (see RFC 1657);
- RFC 1271 - Remote LAN Monitoring MIB (see RFC 1757);
- RFC 1283 - SNMP over OSI (see RFC 1418);
- RFC 1284 - Ether-Like Interface Type MIB (see RFC 1398);
- RFC 1286 - Bridge MIB (see RFC 1493 and RFC 1525);
- RFC 1289 - DECnet phase IV MIB (see RFC 1559);
- RFC 1298 - SNMP over IPX (see RFC 1420);
- RFC 1304 - SMDS Interface Protocol (SIP) Interface Type MIB (see RFC 1694);
- RFC 1316 - Character Device MIB (see RFC 1658);
- RFC 1317 - RS-232 Interface Type MIB (see RFC 1659);
- RFC 1318 - Parallel Printer Interface Type MIB (see RFC 1660);
- RFC 1351 - SNMP Administrative Model;
- RFC 1352 - SNMP Security Protocols;
- RFC 1353 - SNMP Party MIB;

- RFC 1368 - IEEE 802.3 Repeater MIB (see RFC 1516);
- RFC 1389 - RIPv2 MIB (see RFC 1724);
- RFC 1398 - Ether-Like Interface Type MIB (see RFC 1643);
- RFC 1441 - Introduction to SNMPv2 (see RFC 1901);
- RFC 1442 - SMI for SNMPv2 (see RFC 1902);
- RFC 1443 - Textual Conventions for SNMPv2 (see RFC 1903);
- RFC 1444 - Conformance Statements for SNMPv2 (see RFC 1904);
- RFC 1445 - Administrative Model for SNMPv2;
- RFC 1446 - Security Protocols for SNMPv2;
- RFC 1447 - Party MIB for SNMPv2;
- RFC 1448 - Protocol Operations for SNMPv2 (see RFC 1905);
- RFC 1449 - Transport Mappings for SNMPv2 (see RFC 1906);
- RFC 1450 - MIB for SNMPv2 (see RFC 1907);
- RFC 1451 - Manager-to-Manager MIB;
- RFC 1452 - Coexistence between SNMPv1 and SNMPv2 (see RFC 1908);
- RFC 1596 - Frame Relay Service MIB (see RFC 1604);
- RFC 1623 - Ether-Like Interface Type MIB (see RFC 1643); and,
- RFC 1665 - SNA NAU MIB (see RFC 1666).

Subscribing to SNMP-related Working Groups

- 100VG-AnyLAN MIB Working Group
  <vgmib-request@hprnd.rose.hp.com>
- Application MIB Working Group
  <applmib-request@emi-summit.com>
- AToM MIB Working Group
  <atommib-request@thumper.bellcore.com>
- BGP Working Group
  <iwg-request@ans.net>
- Bridge MIB Working Group
  <bridge-mib-request@pa.dec.com>
- Character MIB Working Group
  <char-mib-request@decwrl.dec.com>
- Data Link Switching MIB Working Group
  <aiw-dlsw-mib@networking.raleigh.ibm.com>
- DECnet Phase IV MIB Working Group
  <phiv-mib-request@jove.pa.dec.com>
- Distributed Management
  <disman-request@nexen.com>
- Entity MIB Working Group
  <entmib-request@cisco.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@hprnd.rose.hp.com>
- IDR Working Group
  <bgp@ans.edu>
- Interfaces MIB Working Group
  <if-mib-request@dtl.labs.tek.com>
- IP over AppleTalk Working Group
  <apple-ip-request@cayman.com>
- IPLPDN Working Group
  <iplpdn-request@nri.reston.va.us>
- IPv6 MIB Working Group
  <ip6mib-request@research.ftp.com>
- ISDN MIB Working Group
  <isdn-mib-request@combinet.com>
- IS-IS for IP Internets 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 IGP Working Group
  <ospf-request@gated.cornell.edu>

- PPP Extensions Working Group
  <ietf-ppp-request@merit.edu>
- RIP Working Group
  <ietf-rip-request@xylogics.com>
- Remote Network Monitoring Working Group
  <rmonmib-request@cisco.com>
- Routing over Large Clouds Working Group
  <rolc-request@nexen.com>
- SNA DLC Services MIB Working Group
  <snadlcmib-request@cisco.com>
- SNA NAU Services MIB Working Group
  <snanaumib-request@cisco.com>
- SNMP Agent Extensibility Working Group
  <agentx-request@fv.com>
- SNMPv2 Working Group
  <snmpv2-request@tis.com>
- TCP Client Identity Protocol
  <ident-request@nri.reston.va.us>
- DS1/DS3 MIB Working Group
  <trunk-mib-request@cisco.com>
- Uninterruptible Power Supply Working Group
  <ups-mib-request@cs.utk.edu>
- X.25 MIB Working Group
  <x25mib-request@dg-rtp.dg.com>

Internet Resources

Automated Services

Automated services are available in the Internet, provided "as is" with no express or implied warranty. Each service accepts a MIB module in the body of a message.
MIB module checking:

- Emissary <mib-checker@epilogue.com>
- mosy <mosy@simple-times.org>

MIB module conversion:

- convert SNMPv2 module to SNMPv1
  <mib-v2tov1@simple-times.org>
- convert MIB module to HTML
  <mib-2html@simple-times.org>

Source Implementations

Source implementations are available in the Internet, provided under various no-fee licensing terms.
Agents:

- Beholder: an RMON agent for UNIX
  ftp://dnpap.et.tudelft.nl/pub/btng/
- CMU SNMP: an SNMPv2u agent for UNIX
  ftp://ftp.cisco.com/ftp/kzm/cmusnmp.tar.gz
- UT-snmpV2: an SNMPv2 agent for SPARCs
  http://snmp.cs.utwente.nl/
- WILMA: an SNMP agent for UNIX
  http://www.ldv.e-technik.tu-muenchen.de/dist/INDEX.html

Compilers:

- mosy: a MIB compiler
  ftp://ftp.cisco.com/ftp/kzm/snmptcl.tar.gz
- SMIC: a MIB compiler
  <dperkins@scruznet.com>
- snacc: an ASN.1 compiler
  ftp://ftp.cs.ubc.ca/pub/local/src/snacc/

Platforms:

- NOCOL: a network monitoring package for UNIX
  ftp://ftp.navya.com/pub/vikas/
- Scotty: a Tcl-based environment for management applications
  http://www.cs.tu-bs.de/ibr/projects/nm/
- snmptcl: a Tcl-based environment for management applications
  ftp://ftp.cisco.com/ftp/kzm/snmptcl.tar.gz
- SNMPY: a Python-based environment for management applications
  http://www.rdt.monash.edu.au/\~{}anthony/snmpy/
- Tricklet: a Perl-based environment for management applications
  ftp://dnpap.et.tudelft.nl/pub/btng/
- WILMA: an X-based monitoring package for UNIX
  http://www.ldv.e-technik.tu-muenchen.de/dist/INDEX.html

Other Resources

- IETF Home Page
  http://www.ietf.cnri.reston.va.us/
- The Simple Web
  http://www.cs.utwente.nl/~schoenw/ietf-nm/

- SNMP Testing FAQ
  `http://www.iwl.com/faq.html`
- User-based Security Model (USEC) Resources
  `http://www.simple-times.org/pub/simple-times/usec/`

Publications

SNMP, SNMPv2, and RMON: Practical Network Management

- Author: William Stallings <ws@shore.net>
- Publisher: Addison-Wesley
  `http://www.aw.com`
- ISBN: 0-201-63479-1
- Available: June, 1996

A comprehensive treatment of SNMP-based standards, including a description of the protocols, MIBs, and practical issues. Covers SNMPv1, the 1996 version of SNMPv2, the original RMON1, and the current version of RMON2.

The Simple Book: An Introduction to Networking Management

- Author: Marshall T. Rose
  <mrose.dbc@dbc.mtview.ca.us>
- Publisher: Prentice Hall
  `http://www.prenhall.com`
- ISBN: 0-13-451659-1
- Available: April, 1996

This is the revised 2nd edition discussing both SNMPv1 and the latest information on SNMPv2. Accompanying CD includes both agent and management application software for UNIX.

Publication Information

Featured Columnists

Keith McCloghrie	Cisco Systems
Marshall T. Rose	Dover Beach Consulting
Kaj Tesink	Bell Communications Research

Contact Information

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 via electronic mail, and must be formatted in HTML version1.0. 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 in three editions: HTML, ASCII, and PostScript. For more information, send a message to

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

with a `Subject:` line of

```
help
```

Back issues are available via either the Web or FTP, i.e.,

```
http://www.simple-times.org
ftp://ftp.simple-times.org
```

look under `/pub/simple-times/issues/`. In addition, ***The Simple Times*** has several hard copy distribution outlets. Contact your favorite SNMP vendor and see if they carry it.