The Simple Times™

THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS
VOLUME 7, NUMBER 1 MARCH, 1999

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 as an online journal in HTML, PDF and PostScript. New issues are announced via an electronic mailing list. For information on subscriptions, see page 22.

Bulk Transfers of MIB Data

Ron Sprenkels, University of Twente
Jean-Philippe Martin-Flatin, EPFL

Since the original days of SNMP back in early 1988, the requirements for managing IP-based networks like the Internet have changed considerably. An important change is that the total amount of management information that needs to be transferred has increased greatly. Not only did the size of traditional MIB data grow, for example IP routing tables and TCP connection tables, but also new types of management information appeared, for instance accounting tables, which tend to be bulky. The widely deployed SNMP version 1 was not designed for transferring large amounts of data. The overall latency of such transfers can be quite high and the way in which the SNMP messages are encoded for transmission over the network is not particularly efficient. The new version 3 of the SNMP protocol, while improving on other issues like security and access control, does not improve the transfer of large amounts of MIB data sufficiently, even though it provides a `get-bulk` operation.

In this article, we look into ways of making bulk transfers of MIB data between SNMP agents and managers more efficient. We consider a bulk transfer to be the transfer of several hundreds of kilobytes of MIB data in a single logical transaction. For bulk transfers, our objectives are:

- to reduce the end-to-end latency (i.e., the total time to transfer a set of management data between an agent and a manager, including marshalling, unmarshalling and network transfer);
- to reduce the network overhead (i.e., the ratio between the amount of bytes transferred over the network and the actual management information); and
- to improve the retrieval of SNMP MIB tables (by both reducing latency and network overhead for the particular case of table retrieval).

These objectives share a common goal: to improve the scalability of network management in the IP world.

Improving scalability has become necessary because both the number of systems to be managed, as well as the amount of management information per system has increased.

This article is structured as follows. First we discuss what we consider the three main problems with bulk transfers: latency, network overhead and table retrieval. Next we discuss three different approaches to solve these problems. The first approach aims to be a small evolutionary change to the current SNMPv3 framework, requiring minimal changes to existing SNMP manager and agent implementations. As such, this approach is envisaged to be useful in the short term. The second approach uses a mixture of SNMP and other protocols. The third approach discusses alternative protocols and encodings, abandoning the SNMP protocol and associated BER encoding altogether. This approach will therefore take longer to design, implement and deploy, and is envisaged to be useful in the longer term. This approach also serves as food for thought, and is intended to solicit discussion on future Internet management frameworks and protocols.

Problem #1: Latency

Currently, retrieving large amounts of MIB data involves a high number of PDU exchanges over the network. When using the `get-next` operator, the retrieval of large tables with many rows requires at least one `get-next` operation per table row. If a table row does not fit into a single message (due to message size constraints) even more operations per row are needed.

RFC 1187 describes an algorithm that speeds up the retrieval of an entire table by using multiple threads in parallel where each thread retrieves only a portion of the table. To make this work, one needs a manager which supports multiple threads and which has knowledge about the distribution of instance identifiers in the table. Note that the algorithm does not reduce the total number of request/response PDU exchanges. Instead, it is more efficient in terms of latency because several threads gather data simultaneously. The price for achieving reduced latency with multiple threads is bursty SNMP traffic, which can cause overload problems on the agent side.

If the algorithm described in RFC 1187 is not used, then each `get-next` operation must be completely finished before the next one can start. Things get worse in case packets get dropped within the network, since retransmission timers have to expire and retransmissions must succeed before the retrieval process can continue.

The situation improves with the introduction of the `get-bulk` operator. However, the response to a single `get-bulk` operation still has to fit into a single UDP packet. In theory, UDP can handle packets of nearly 64 KBytes. In practice, the maximum packet size will be much smaller. For the hundreds of kilobytes of MIB data we are considering here, even the use of `get-bulk` results in a large overall delay.

Each request/response exchange (be it `get-next` or `get-bulk`) involves at least a network round trip delay time, possibly time-out and retransmission delays, and probably also other protocol stack overhead delays (e.g. marshalling and unmarshalling of data, context switching). In summary, the overall latency of a bulk transfer is high because of the large number of PDU exchanges involved and their synchronous nature.

Problem #2: Network Overhead

Network overhead is the proportion of the network bandwidth used for management which is thus unavailable for the transport of user data. Particularly in the case of bulk transfers, which deal already with large amounts of data, it is import to keep that overhead low. All currently active SNMP frameworks (SNMPv1, SNMPv2c and SNMPv3) are inefficient in terms of the number of bytes needed to transfer MIB data over the network. We identified three causes for this inefficiency: the Basic Encoding Rules (BER), the OID naming scheme and what we will call the "`get-bulk` overshoot" problem.

BER encoding is well known to be fairly inefficient in terms of network overhead. Mitra [1] and Neufeld and Vuong [2] describe this issue in detail. At the time BER was chosen for SNMP, network overhead was not considered to be a main issue; the reason BER was selected was because it was readily available and simple to implement. Since alternative encoding rules exist nowadays, it is feasible to reduce network overhead by selecting another set of encoding rules. These new rules, however, should not increase latency too much due to additional encoding/decoding times.

If we look at the OIDs of the objects involved in a bulk transfer, we observe a high degree of redundancy. For the objects in a table, we see multiple occurrences of identical portions of OIDs: all OID prefixes up to the column number are identical, as are the instance identifier postfixes of all entries of a single table row. Because of this, redundant information is transferred, resulting in a higher network overhead than strictly needed.

The `get-bulk` operator also adds to the network overhead, since the manager, which does not know the size of the table to be retrieved, has to guess a value for the `max-repetitions` parameter. Using small values for `max-repetitions` may result in too many PDU exchanges. Using large values, however, may result in an "overshoot" effect: the agent returns data that does not belong to the table the manager is interested in. This data will be sent over the network back to the manager, just to be discarded.

Problem #3: Table Retrieval

The problems with table retrieval discussed here are holes in tables, table consistency and `get-bulk` overshoot consequences.

Retrieving table objects is more complex than retrieving other objects. This is due to the fact that the SNMP frameworks have no notion of tables, but only of conceptual tables. The difference between these two concepts is important for tables that have rows in which some columnar objects do not exist; in other words, for tables that allow their row entries to have "holes" in them. Consider the case where a manager wants to retrieve a table by performing repeated `get-next` operations. In most cases the manager uses for each row of the table a single `get-next` operation. The `get-next` PDU contains a list of OIDs, one for each column of the row; the value of these OIDs is usually taken from the response of the previous `get-next` operation. If there is a hole in the table, the `get-next` operation returns the elements of the next row, except for the column in which there is a hole. For this column the `get-next` operation returns the next available object in the MIB tree, which is the columnar object for the next table row that does have a value in that column (we will not discuss what happens if none of the remaining table rows has a value in that column). As a consequence, the manager is faced with a set of columnar objects that do not belong to the same row anymore; it has the cumbersome task of finding out what objects belong to which rows and where the holes are. In short, reconstructing the actual table, including determining where the holes in the table are and where the table ends, is a time-consuming task.

Another problem is that the manager has no guarantee that it will retrieve a table in a consistent state. This is particularly true for large tables, because the retrieval of such tables involves a large number of PDU exchanges, which take a considerable amount of time. If in the meantime some table elements are changed by the agent, the manager ends up with an inconsistent view of the table.

Finally there is a problem which we call "`get-bulk` overshoot." When `get-bulk` is used to retrieve a table, object values may be returned that do not belong to the table of interest. If, for example, a `max-repetitions` value of 50 is used, and the table contains only 10 additional elements, 40 elements will be returned that are not really needed. In this case, the agent processed information, retrieved object values from the instrumentation and used resources, just to have the manager discard the information. This can add up to quite an amount of wasted resources.

Now that we have outlined some problems with bulk MIB data transfers, we will discuss three approaches to solving them.

Approach #1: Extending SNMP

In the first approach, we seek to make small evolutionary additions to the SNMP frameworks. As a result, any changes or additions to the current protocol should be easy to implement with limited changes to existing implementations, thus protecting the investment in current implementations of the frameworks.

An Additional Transport: SNMP over TCP

Adding an SNMP over TCP transport mapping (in addition to the preferred transport mapping over UDP) is probably the most lightweight addition that can be made to the SNMP frameworks, and it already solves some of the problems outlined earlier. The immediate effect of moving from UDP to TCP is that the UDP limitation of the maximum SNMP message size of nearly 64 KBytes or less disappears. TCP has a window mechanism that allows several chunks of data to be in transit in parallel. This removes the cause for additional round trip time latencies when a table row does not fit into a single UDP message, or when the requested number of repetitions for a `get-bulk` request does not fit into a single UDP packet. As a result, overall latency will decrease and table consistency will improve. A downside is that large buffers are required on both the manager and the agent to store the large SNMP messages. This can be a serious problem for agents in embedded environments.

Several issues should be investigated:

- To prevent the need for large buffers, a scheme could be defined where many related smaller messages are sent over the same TCP connection.
- As a result of using TCP as a transport, both agents and managers get a task they did not have before: managing their TCP connections. Possible strategies are to close each TCP connection immediately after use, to keep and manage a pool of established TCP connections, or to close TCP connections after some period of inactivity.
- SNMP manager and agent implementors should have the option to decide on a per-operation basis whether to use UDP or TCP. When a manager expects to retrieve little data from a large set of

> agents, UDP is the best choice. Conversely, when large tables or MIB subtrees are expected to be retrieved from a small set of agents, TCP is a better choice.

In the early days of SNMP, when a well-known UDP port was reserved for SNMP (161), the same TCP port number was also allocated to SNMP. As a result, the reservation of a well-known port for SNMP over TCP is not an issue.

In 1994, the University of Twente had (temporarily) a prototype of SNMPv2p running over TCP. Recently, Schönwälder and Deri modified the Linux CMU SNMP library and the UCD-SNMP software to transport SNMP traffic over TCP. These experiments suggest that the extension of an existing SNMP implementation to support TCP should be relatively straightforward.

New Encoding Rules and/or Compression

An issue that affects both latency and network overhead is the way the management information is encoded for transmission over the network. We have two different ways (that can be used in conjunction) to reduce the network overhead with respect to plain BER: replacing BER by a different set of encoding rules and adding compression.

There are two different types of encoding schemes: schemes that use a definite form for the length field and schemes that use an indefinite form. Definite-form schemes require the whole of the message to be in a buffer, because they insert the length of the message in front of the message. Indefinite-form schemes do not put a length field in front of a message. Instead, they mark the end of an encoded ASN.1 element by a special byte. Hence, an indefinite-form scheme does not require the complete message to be buffered and it can encode on the fly.

We first note that all versions of SNMP mandate the use of the definite form of BER. Replacing BER by a different encoding scheme therefore requires a new protocol version and is thus a major change.

The ISO has defined several alternatives to BER. PER encoding (Packed Encoding Rules) has approximately 30% shorter encodings, at the expense of a small increase in encoding time. PER allows the use of the indefinite form, so no large encoding buffer is needed. Lightweight Encoding Rules (LER) decrease overall latency by ensuring quick encoding and decoding. However, network overhead is adversely affected, because the encodings can be much longer than those generated by BER. Distinguished Encoding Rules (DER) use the definite form only. They slightly improve encoding time over BER while having a minimal impact on network overhead compared to BER. Finally, Canonical Encoding Rules (CER) use the indefinite form like PER, but are less demanding in terms of encoding time.

We initially thought we should move from a definite-form encoding scheme to an indefinite-form encoding scheme in order to avoid large buffers. We later realized that the SNMP version 3 (SNMPv3) message header can include an authentication digest, which is computed over the whole PDU. As a result, we must buffer the entire PDU before transmitting it anyway if authentication is used. Therefore, switching to alternate encoding rules does not really prove advantageous over BER in the general case.

SNMPv3 allows to add encryption envelopes to SNMP messages. This feature can not only be useful for its intended purpose, which is encryption, but it can also be exploited to achieve data compression. By adding an encryption algorithm that in fact compresses the message, the size of the messages that are transmitted over the wire decreases. Defining compression as an encryption algorithm allows to add compression to SNMPv3 without making any changes to the protocol. However, since there is no `noAuthPriv` security level in SNMPv3, one has to use authentication in order to take advantage of compression.

Using compression relieves us of the need to abandon BER, in order to replace it with a new more efficient encoding scheme. It leaves the installed base of implemented and debugged BER encoding and decoding software in place. Any standard compression algorithm can be used like e.g. DEFLATE (RFC 1951), for which stable, debugged implementations are readily available.

Additional Protocol Operation: get-subtree

We advocate introducing a new SNMP protocol operation to be used for the retrieval of complete MIB sub-trees, since neither `get-next` nor `get-bulk` are efficient for that purpose. Note that retrieving an entire table, an entire table column or a part of a table column are all special cases of a MIB subtree. We define the `get-subtree` operation to retrieve all objects below a particular node in the MIB tree. By allowing the operation not only to retrieve a single subtree, but also to retrieve multiple subtrees with a varbind list, the operation becomes even more powerful. It can then be used to retrieve selected columns of a complete table or selected columns within a range of rows of a column.

Examples of the usage of this operation include retrieving the entire interface table (ifTable), retrieving the operational status of all interfaces in the ifTable, retrieving both the operational and the administrative statuses of all interfaces in the ifTable, and retrieving the state and remote address of all TCP connections to a particular local address/port combination. For each of these examples, the information is requested in a single protocol operation.

The amount of data returned for a single get-subtree operation can be quite large; this has two implications. First, the get-subtree operation will be most useful when used over TCP. The strict message size limitations of the UDP transport would immediately break the advantages of this new operation. Second, even when using TCP as a transport, it will generally not be feasible for agents to have memory buffers to store huge response message. Further, since it might take some time for the agent to collect the MIB data, other requests may have to wait some time before a single-threaded agent will process them. Therefore, a mechanism is needed that allows the agent to return multiple related response messages for a single get-subtree request. The TCP transport will take care of any required retransmissions and it will keep the responses in order. The TCP transport will also provide a window that allows multiple responses to be in transit concurrently.

In summary, the main advantages of the get-subtree protocol operation are:

- A table can be retrieved using a single protocol operation. As a result, latency can be minimal: a single round trip time for the protocol operation, plus what is added by the TCP transport for connection setup, segmentation, retransmissions, etc.
- The agent implementation collects from its instrumentation only the values of the requested objects; there is no get-bulk overshoot anymore.
- The absence of get-bulk overshoot also means that no network overhead is generated for objects that are not of interest anyway.
- The single operation property gives the agent the best chance of returning a consistent view of the table to the manager, much better than with a large number of separate get-next or get-bulk operations.
- It is no longer necessary to guess max-repetitions values.
- The get-subtree operation can be an extension to both SNMPv1 and SNMPv3. No new message format is needed, only a new PDU type. This qualifies get-subtree as a relatively small evolutionary step.

An agent implementation of get-subtree collects and returns each of the subtrees specified in its varbind list simultaneously, that is, row by row for a table. This ensures an efficient retrieval of table rows from the instrumentation and it minimizes the risk of getting inconsistencies within a single row.

The problem with holes in tables discussed previously still exists. The reconstruction of the conceptual table remains the task of the manager. Only the retrieval and transport over the network is greatly simplified by this new protocol operation.

Approach #2: Hybrid Solutions

In this approach we present a solution which uses a combination of SNMP and other protocols.

Bulk File MIB and FTP Client MIB

Stewart proposed to solve the bulk transfers problem by using SNMP together with the File Transfer Protocol (FTP). His proposal consists of two MIB modules. The first MIB module (CISCO-BULK-FILE-MIB) specifies how an SNMP agent stores a user-defined set of MIB data into a local file. The second MIB module (CISCO-FTP-CLIENT-MIB) can be used to upload local files to an FTP server using the FTP protocol. An SNMP agent implementing both MIB modules can be instructed to save a specified (large) amount of local MIB data into a file and upload that file to a particular FTP server. We will now describe these MIB modules briefly.

The CISCO-BULK-FILE-MIB defines three tables. The cbfDefineFileTable defines the name of the file, how it is stored and what encoding format will be used. One or more entries in the cbfDefineObjectTable are associated to a row in the cbfDefineFileTable. The entries specify what local MIB objects should be put in the file upon creation. A complete MIB table can be specified in a single entry in the cbfDefineObjectTable. A manager initiates the creation of the actual file by doing a set operation on the cbfDefineFileNow object. This results in a new entry in the cbfStatusFileTable which keeps track of the progress of the file creation.

The storage type of a bulk file can either be permanent, volatile or ephemeral, where the latter indicates that data exists only in small amounts until it is read. This storage type, when used in combination with the CISCO-FTP-CLIENT-MIB, prevents the need for a buffer large enough to hold the complete file.

There are three options for the format of the data files: BER encoded, binary and human-readable ASCII. The BER encoded format is identical to an SNMP varbind list. The binary format consists of tags and data fields. There is a tag to set a standard OID prefix, a tag for a single object, and some tags to encode tables. Tables are encoded with little OID redundancy: for each entire row only the common instance portion of all the OIDs in that row is encoded. The binary format uses a proprietary encoding scheme for the ASN.1 primitive types INTEGER, OCTET STRING and OBJECT IDENTIFIER. The ASCII format is a mechanical translation of the binary format; translation rules for tags and values to ASCII are given in the MIB specification.

The CISCO-FTP-CLIENT-MIB has a single table. An entry in the cfcRequestTable table specifies a local file that is to be uploaded to a specified FTP server, either in binary or in ASCII mode, using a specified user name and password. A manager can initiate a file transfer from the agent to an FTP server by setting the cfcRequestEntryStatus to active. The progress and result of the transfer can be monitored by reading the cfcRequestOperationState and cfcRequestResult objects. The manager can abort an ongoing transfer by setting the cfcRequestStop object.

In summary, this solution to the bulk transfer problem requires agents to implement two MIBs and the manager to configure entries in several MIB tables to initiate and control bulk transfers. This means that bulk transfers are treated totally different from normal accesses to MIB data. For this reason, security needs to be considered separately for these transfers. For example, there is no mechanism in place which authenticates or encrypts management data while in transit over the network. The hybrid solution described here also requires an FTP server on the manager side. This means that management data retrieved via a bulk transfer is processed very differently from management data retrieved via SNMP since it becomes available in a file on an FTP server.

Approach #3: Other Protocols and/or Encodings

The most important value of the SNMP management frameworks lies in the large amount of existing MIB specifications. These are not only the standard MIB specifications developed within IETF working groups, but also proprietary, vendor-specific MIB specifications. The MIB specifications are valuable because they represent detailed knowledge of what is relevant management information for a large variety of networking devices, protocols, network elements, transmission medias and so forth. This value exists regardless of the protocol used to move information around or the way the information is encoded while in transit. Therefore, both the SNMP protocol and the encoding rules can be replaced by something else. Some possibilities for such replacements and their properties are examined in this section.

MIME

From a software engineering point of view, management data is just another example of structured data. The Internet community has a standard way for transferring structured data called MIME, which is essentially a "bag and tag" scheme to transfer data. In order to transfer management data using MIME, we need to define a MIME type for it (the tag) and we need to define how data of that type is structured inside the bag. Because of the overhead it introduces, MIME is only suited for transferring bulk data. Furthermore, by means of a transfer encoding, MIME allows content to be transparently compressed while in transit. We will now discuss three options for the MIME type and its encoding.

- Option #1 is to define a new MIME tag for putting a BER-encoded SNMP message, which is normally sent over UDP, inside the MIME body. Using the multi-part feature, multiple related SNMP messages can be put into a single MIME envelope.

- Option #2 is to define a new MIME tag for putting management data encoded in ASCII into the MIME body. An ASCII representation for each SNMP protocol operation and for all SMI data types must be defined. Using an ASCII encoding scheme has the advantage that it is usually easier for programmers to read, understand and process data in a human-readable format.

- Option #3 is to represent SNMP protocol operations and SMI data types using an XML Document Type Definition (DTD). If XML gets good acceptance in industry, we will see many programmers with experience of using XML. Encoding management data in XML means that many programmers will have the knowledge and skills required to build management applications. This is a big advantage compared to the current situation where programmers need to have at least some basic knowledge about ASN.1 and BER.

Wrapping management data in MIME types has the advantage that several existing protocols can be used to move MIB data around. This includes SMTP (a store and forward protocol) or HTTP (a request/response protocol). We will look a bit closer at HTTP now.

HTTP

The well-known HTTP protocol is a good candidate for transferring MIME-encapsulated management data. We identified three reasons for that. First, the protocol was designed to transfer MIME data. Second, there currently is a clear trend in industry to embed HTTP servers for management purposes in networking equipment. So the chances of HTTP being accepted by industry as a protocol for management seem good. Third, HTTP is based on TCP, so it is suitable for the transfer of bulk data, as we outlined earlier.

There are some downsides to using HTTP for management as well; we will name three. First there is the feature richness of HTTP. HTTP has numerous options and features that are valid and useful for its intended purpose, which is to be used as a document transfer protocol in the World-Wide Web. However, for the transfer of management data, many of those features will not be useful or usable. Conforming implementations of the protocol must include all of these features. As a result, the HTTP implementations in network devices will be needlessly big and complex.

Second, since the development and standardization of HTTP will remain focused on its original purpose, future versions of the protocol might have characteristics that are unwanted for a management protocol. Also, for the same reason, it will probably be difficult to get new features that are desirable for the use as a management protocol into HTTP.

Finally, the security mechanisms proposed and used in conjunction with HTTP do not directly map to the security mechanisms defined in SNMPv3. This means that either some mappings need to be defined or that there will be different security mechanisms (authentication, privacy, access control) for accessing the same MIB data via SNMP or HTTP.

Conclusions

In this article, we looked at bulk transfers of MIB data. The current SNMP management frameworks are not very efficient for bulk transfers. The three main problems identified in this paper are latency, network overhead and table retrievals. We discussed three different approaches to speed up bulk transfers of MIB data.

We first looked at solutions within the SNMP framework. We believe that within the boundaries of the current SNMPv3 framework and with relatively little effort and small changes, the problems can be solved to a large extent. Latency can be significantly decreased by using TCP as a transport and by introducing a new `get-subtree` protocol operation. Network overhead can be decreased by compressing the payload of an SNMP message. Table retrieval can be improved by applying the new `get-subtree` operation to conceptual tables.

Second, there are possible hybrid solutions. We presented a solution proposed by Stewart. A downside to this solution might be that it treats bulk transfers as a separate, special issue, and still requires all of the normal SNMP framework and protocol stack to be in place. Furthermore, a whole new set of security problems will be the result of such an approach. Other hybrid solutions are probably also possible, but are not discussed in this article.

The third approach is to replace SNMP with another protocol. By using a protocol that runs over TCP, bulk transfer latency can remain low. By using compression on the encoded management information, network overhead can be kept low. If a mainstream technology is used for representing management information, e.g. XML, building management applications will no longer require skills specific to network management.

The first solution aims to be a small evolutionary step with respect to the current SNMPv3 management framework. It is relatively easy to implement and keeps the current implementations largely intact. This protects investments in current SNMP technology. The second solution is probably also fairly easy to implement, but has some architectural and security-related downsides that make it in our view less attractive than the first one. The third solution is not covered in as much detail as the first two. It will take quite some work to further define that solution. Because it breaks so radically with the current SNMP framework it will be more difficult to get it implemented and deployed. As such, it is intended to serve as food for thought for the long-term future of Internet management.

Acknowledgments

The ideas presented in this article emerged during a two-day meeting that took place in November 1998 in Lausanne, Switzerland. The following people were involved:

- L. Deri, University of Pisa
- J.P. Martin-Flatin, EPFL
- A. Pras, University of Twente
- J. Schönwälder, Technical University Braunschweig
- R. Sprenkels, University of Twente
- B. Wijnen, IBM T.J. Watson Research

This meeting resulted in the creation of the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF). Contact information and additional documentation can be found on the NMRG Web page at http://www.ibr.cs.tu-bs.de/projects/nmrg/.

References

[1] N. Mitra, *Efficient Encoding Rules for ASN.1-Based Protocols*, AT&T Technical Journal, 73(3):80-93, 1994.

[2] G. Neufeld, S. Vuong, *An overview of ASN.1*, Computer Networks and ISDN Systems, 23:393-415, 1992.

SNMP++: An Object Oriented Approach to Network Management Programming

Peter Erik Mellquist, Hewlett Packard Corporation

Various Simple Network Management Protocol (SNMP) Application Programmers Interfaces (APIs) exist which allow for the creation of network management applications. The majority of these APIs provide a large library of functions that require the programmer to be familiar with the inner workings of SNMP and SNMP resource management. Most of these APIs are platform specific, resulting in SNMP code specific to an operating system or network operating system platform and thus not portable.

Application development using C++ has entered the main stream and with it a rich set of reusable class libraries are now readily available. What is missing is a standard set of C++ classes for network management. An object oriented approach to SNMP network programming provides many benefits including ease of use, safety, portability and extensibility. SNMP++ offers power and flexibility that would otherwise be difficult to implement and manage.

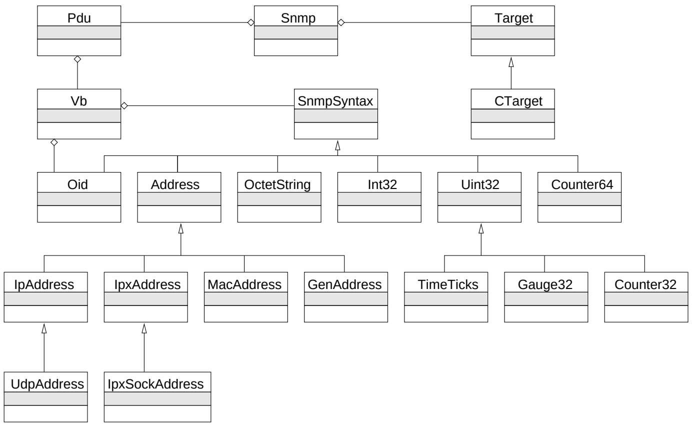
What Is SNMP++?

SNMP++ is a set of C++ classes that provide SNMP services to a network management application developer. SNMP++ is primarily focused on management application development, but many of the C++ classes have been used in the agent space as well. SNMP++ is not an additional layer or wrapper over existing SNMP engines. SNMP++ utilizes existing SNMP libraries in a few minimized areas and in doing so is efficient and portable. SNMP++ is not meant to replace other existing SNMP APIs such as WinSNMP, rather it offers power and flexibility which would otherwise be difficult to manage and implement.

SNMP++ Features

SNMP++ is based around a set of C++ classes including the Object Identifier (`Oid`) class, Variable Binding (`Vb`) class, Protocol Data Unit (`Pdu`) class, `Snmp` class and a variety of classes making work with ASN.1 and SMI types easy and object oriented.

The classes manage various SNMP structures and resources automatically when objects are instantiated and destroyed. This frees the application programmer from having to worry about de-allocating structures and resources and thus provides better protection from memory corruption and leaks. SNMP++ objects may be instantiated statically or dynamically. Static object instantiation allows destruction when the object goes out of scope. Dynamic allocation requires use of the C++ constructs `new` and `delete`. Internal to SNMP++ are various SMI structures which are protected and hidden from the public interface. All SMI structures are managed internally, the programmer does not need to define or manage SMI structures or values. For the most part, usage of 'C' pointers in SNMP++ is non existent. By hiding and managing all SMI structures and values, the SNMP++ classes are easy and safe to use. The programmer cannot corrupt what is hidden and protected from scope.



An SNMP++ application communicates with an agent through a session model. That is, an instance of the `Snmp` class maintains logical connections to specified agents. An application may have multiple `Snmp` instances, each instance communicating to the same or different agent(s). This is a powerful feature that allows a network management application to have different sessions for each management component. For example, an application may have one `Snmp` object to provide graphing statistics, another `Snmp` object to monitor traps, and a third `Snmp` object to allow SNMP MIB browsing. SNMP++ automatically handles multiple concurrent requests from different `Snmp` instances. Alternatively, a single `Snmp` instance may be used for everything.

The majority of SNMP++ is portable C++ code. Only the implementation of the `Snmp` class is different for each target operating system. If your program contains SNMP++ code, this code will port without any changes. Currently SNMP++ implementations are available for Microsoft Windows NT, Windows '95 and '98, HP UNIX, and Sun Solaris.

SNMP++ supports automatic time-out and retries. This frees the programmer from having to implement time-out or retry code. Retransmission policy is defined in the SnmpTarget class. This allows each managed target to have its own time-out / retry policy.

SNMP++ supports a blocking and an asynchronous model. The blocked mode for MS-Windows allows multiple blocked requests on separate Snmp class instances. SNMP++ also supports a non-blocking asynchronous mode for requests. Time-outs and retries are supported in both blocked and asynchronous modes.

SNMP++ has been designed with support and usage for SNMP version 1 (SNMPv1) and version 2c (SNMPv2c). All operations within the API are designed to be multi-lingual and they are not SNMP version specific. Through utilization of the SnmpTarget class, SNMP version specific operations are abstracted. SNMP++ supports all six SNMP operations (Get, GetNext, GetBulk, Set, Inform and Trap) through corresponding Snmp member functions. Each of these six functions utilizes similar parameter lists and operates in a blocked or non-blocked (asynchronous) manner. SNMP++ is designed to allow trap reception and sending on multiple transports including IP and IPX. In addition, SNMP++ allows trap reception and sending using non-standard trap IP ports and IPX socket numbers.

SNMP++ is implemented using C++ and thus allows a programmer to overload or redefine behavior which does not suite their needs. For example, if an application requires special Oid object needs, a subclass of the Oid class may be created, inheriting all the attributes and behavior the Oid base class while allowing new behavior and attributes to be added to the derived class.

An Introductory Example

Here is a simple example that illustrates the power and simplicity of SNMP++. This example obtains the MIB-2 sysDescr.0 object from the specified agent. The example shows all code needed to create a SNMP++ session, get the system description, and print it out. Retries and time-outs are managed automatically.

```
#include "snmp_pp.h"
#define SYSDESCR "1.3.6.1.2.1.1.1.0"        // OID for sysDescr.0
void get_system_descriptor()
{
  int status;
  CTarget ctarget((IpAddress) "10.4.8.5"); // SNMP++ community target
  Vb vb(SYSDESCR);                         // SNMP++ VB Object
  Pdu pdu;                                 // SNMP++ PDU

  // Construct a SNMP++ SNMP session object. Check the
  // creation status and print an error message on failure.

  Snmp snmp(status);
  if (status != SNMP_CLASS_SUCCESS) {
     cout << snmp.error_msg(status);
     return;  }

  // Add the varbind to the pdu object and invoke an SNMP get
  // operation. Print the result or an error message.

  pdu += vb;
  if ((status = snmp.get(pdu, ctarget)) != SNMP_CLASS_SUCCESS)
     cout << snmp.error_msg(status);
  else {
     pdu.get_vb(vb,0);
     cout << "System Description = "<< vb.get_printable_value(); }

};  // Thats all!
```

The actual SNMP++ calls are made up of ten lines of code. A CTarget object is created using the IP address of the agent. A variable binding (Vb) object is then created using the object identifier of the MIB object to retrieve. The Vb object is then attached to a Pdu object. An Snmp object is used to invoke a get operation. Once retrieved, the response message is printed out. All error handling code is included.

Applications Using SNMP++ Today

A significant number of commercial network management applications have been successfully deployed using SNMP++. This includes applications running standalone as well as those integrated within network management platforms. In the area of Windows based management applications, SNMP++ is tightly integrated with WinSNMP allowing sharing of trap services.

All source code for SNMP++ is freely available to any developer. This includes all source code and make files for building the libraries on MS-Windows, HP UNIX or Sun Solaris. Since the code is ANSI C++ compliant, it can also be ported to other platforms easily. Developers are free to use SNMP++ in their products without any royalties.

Future of SNMP++

SNMP++ has evolved to address a variety of needs. To date SNMP++ includes a complete API for SNMP version 1 and version 2c. In the near future, developments in the area of SNMP version 3 and agent side classes will be available. More information on SNMP++ including a complete specification, libraries and source code is available at http://rosegarden.external.hp.com/snmp++/. Discussions on SNMP++ can be mailed to the WinSNMP mailing list. To subscribe, send an electronic mail message to listserv@mailbag.intel.com with subscribe winsnmp in the body.

References

[1] Rumbaugh, James, *Object-Oriented Modeling and Design*, Prentice Hall, 1991

[2] Stroustrup, Bjarne, *The C++ Programming Language*, Second Edition, Addison Wesley, 1991

[3] Mellquist, Peter E., *SNMP++ An Object Oriented Approach to Network Management Programming Using C++*, Sunsite FTP server.

[4] Mellquist, Peter E., *SNMP++: An Object-Oriented Approach to Developing Network Management Applications*, Prentice Hall, 1997.

[5] Natale, Bob, *WinSNMP v2.0 - Evolution of an industry-standard API*, *The Simple Times* 6(1), March 1998.

SNMPv3 Support for SNMP++

Jochen Katz, University Karlsruhe

SNMP++v3 was developed within the scope of a students assignment at the University of Karlsruhe. The purpose of this work was to integrate support for SNMPv3 into SNMP++. SNMP++v3 and AGENT++v3, a C++ class library for SNMP agents, are used to build an agent which grants read/write access to objects that need to be protected from unauthorized access. Three months of time had to suffice to survey SNMPv3, extend SNMP++ and write an elaboration. The base of this implementation are the RFCs 2271-2275 and SNMP++ version 2.61. Peter Mellquist will have a look at this implementation and probably take parts of it to build an official version of SNMP++ with support for SNMPv3. For more information about SNMP++v3 and AGENT++v3, visit the Web page at http://www.fock.de/agent++/.

Requirements

This implementation had to meet several requirements:

1. Existing programs for SNMP++ must stay usable with SNMP++v3.
2. Changes to existing SNMP++ classes and functions should be kept to a minimum. This will make it easy to patch the modifications into future releases of SNMP++.
3. The details of time synchronization and engineID discovery should be hidden from the user. In particular, SNMP++v3 must re-send requests in case a Report-PDU is received which contains an unknownEngineIDs or a notInTimeWindows counter.

To support SNMPv3, parts of the dispatcher and the complete SNMPv3 message processing model (v3MP) and the user-based security model (USM) were implemented and integrated into SNMP++. The SNMP++ classes `Snmp` and `SnmpMessage` needed modifications.

Message Processing

To hide the time synchronization and engineID discovery from the user the `Snmp` class was modified. If the user requests the class to send a `Pdu` with SNMPv3, first the `Pdu` is stored for later reference and the engineID of the host specified in the target object is determined. If the engineID is unknown, the zero length engineID is used. Then the request is treated like any other SNMPv1/SNMPv2c request, i.e. it is passed to the `SnmpMessage` class, which dispatches the message to the appropriate Message Processing Model. The returned serialized message is sent over the network and the response is passed to the `SnmpMessage` class for deserialization. In case the received `Pdu` is a Report-PDU, it is checked whether it contains the unknownEngineIDs or the notInTimeWindows counter. If this is true, the whole process is repeated, i.e. the engineID is determined, the original message is serialized and sent again. Additional tests prevent an infinite loop. For asynchronous requests, this test is implemented by a new callback function that is called instead of the function specified by the user.

SNMP++ dispatches messages automatically between the network and the application. The dispatcher checks the version of incoming or outgoing messages and either calls the new functions of the v3MP or the standard functions to parse or build SNMPv1/SNMPv2c messages. The ASN.1/BER functions are called in the SNMP++ class `SnmpMessage`. The methods of this class were extended to check the version and to call the correct message processing model. The methods of the `SnmpMessage` class return and are called with a `Pdu`, the version and the community. However, the v3MP of this implementation needs and returns additional values (engineID, securityName, securityModel, securityLevel, contextEngineID and contextName). As a `Pdu` object does not contain any version specific values and since the interface of the `SnmpMessage` class should not be modified, the community string was chosen to hold those parameters. The user calls a function that writes all values except the engineID separated by a backslash into the community string. The engineID is added by the `Snmp` class. This form of encoding has to be changed to a length based encoding, as the engineIDs can contain arbitrary characters.

The main part of the v3MP module was implemented as described in the RFCs. The two SNMP ASIs to prepare an outgoing message (prepareOutgoingMessage and prepareResponseMessage) are implemented in one function that only gets the values for engineID, securityModel, securityName, securityLevel, contextEngineID, contextName and PDU and returns the serialized message. Similarly, the function to parse an incoming message gets the serialized message and returns all the values the first function gets as input. All other parameters are not needed in SNMP++ or can be determined during processing: transportDomain and transportAddress are not needed as the engineID is passed to the v3MP, messageProcessingModel is assumed to be v3MP, expectResponse and pduVersion can be determined from the Pdu and sendPduHandle is not necessary as messages are dispatched to the application using the requestID of the Pdu. The v3MP does not return a stateReference as this reference would have to be passed through the SnmpMessage and Snmp classes to the message queue class and would imply the change of several interfaces. So all stateReferences are cached inside the v3MP.

For engineID discovery the following procedure is used: The v3MP is called to build a message with a zero length engineID. The v3MP sets the securityLevel to noAuthNoPriv and deletes the variable bindings from the Pdu. Then the standard behavior for a request message is used. When the answer is processed, the engineID is automatically added to the list of known engineIDs. As this answer contains a Report-PDU with the unknownEngineIDs counter, the Snmp class will start the serialization process again.

SNMP++ uses the requestID of the Pdu to match incoming responses to outstanding requests. If SNMPv3 is used, a response possibly does not contain the requestID of the sent message (this happens if the agent can not decrypt the scopedPDU). For this reason the stateReference of each request contains the requestID and if a report message contains a wrong requestID, it is set to the saved value. For other message types the requestID is not changed as those messages have to contain the correct requestID.

Security Protocols

The USM module contains a function to generate an outgoing message and another function to process an incoming message. The standard authentication and privacy protocols (MD5, SHA-1 and DES) are implemented. An additional privacy protocol has been implemented which uses the IDEA encryption algorithm.

The security modules use the MD5 and DES routines of RSAEuro, the SHA routines of Uri Blumenthal and the IDEA routines of Tatu Ylonen.

The USM module contains two user tables, one with the user names and passwords and one with the localized keys for each used engineID. If SNMP++v3 is used in a manager, the user can add entries to the first table. Entries in the second table are automatically created if the USM is called to process or build an encrypted or authenticated message. If the user changes an entry in the first table, all appropriate entries in the other table are deleted. Both tables are deleted at program exit. As the calculation of localized keys may take several seconds and since an agent should not store passwords, the first table is not used in an agent. Users can be added at initialization time with passwords, in this case localized keys are computed with the local snmpEngineID, or through the usmUserTable of the agent. Several functions were added to the USM module to assist the user if he wants to change the keys in the usmUserTable in an agent.

Lessons Learned

Most of the time needed to implement the SNMPv3 support was spent on the USM and the v3MP. The time needed to modify SNMP++ was spent mainly on the handling of asynchronous requests and on the handling of the additional error codes of the v3MP.

This implementation was tested against the agents from UCD and MG-Soft. With both agents engineID discovery, time synchronization and exchange of noAuthNoPriv, authNoPriv (MD5 and SHA) and authPriv (MD5/DES and SHA/DES) messages worked. An agent written with AGENT++v3 and SNMP++v3 was used to test the cloning of users and the key change algorithm.

Future versions of SNMP++v3 could improve the handling of the SNMPv3 specific parameters. The chosen solution, which encodes those parameters into the community string, works but it contradicts the concept of SNMP++. According to this concept, a new target class which contains the securityName, securityModel and securityLevel and which is possibly responsible to store the engineIDs for each address, has to be defined. The context information would be stored in the Pdu or passed directly to the methods get, get_next, etc. of the Snmp class. (This is already done with the parameters nonRepeaters and maxRepetitions for a get-bulk operation). The community based solution has the advantage that it is simple to implement, but it is bad design to misuse the community string that way. Whereas the target solution fits into the concept of SNMP++, but the implementation is more complex and introduces incompatible changes in the SNMP++ API.

The functions of the USM that assist the user to change a key for one agent could be extended to do the complete key change for several agents. To improve the performance of the USM, the table that contains the localized keys could be saved at program exit and restored at initialization time.

SNMPv3 at Networld+Interop

Muriel Appelbaum, BMC Software
Rob Frye, MCI

The May 1998 Networld+Interop in Las Vegas, Nevada USA had several multi-vendor technology showcase 'Hot Spots' in the Trade Show Exhibition. The SNMPv3 Hot Spot was hosted by SNMP Research International and organized by David Reid. It highlighted the security aspects of the SNMPv3 protocol, whose standards track RFCs have Proposed Standard status. The ten organizations participating in the Hot Spot demonstrated interoperability of working prototypes and products, discussed the progress of the standard and promoted the awareness of SNMPv3. The participants were

- Advent Network Management
- Bay Networks
- BMC Software
- Cisco Systems
- Hewlett-Packard
- IBM Networking
- Liebert Corporation
- Tivoli
- SNMP Research International and the
- University of Quebec in Montreal.

According to SNMP Research's Jeff Case "The IETF bases its work on rough consensus and running code. The standards documents represent rough consensus and the [Hot Spot] demo showed running code." In the Hot Spot, products and work in progress representing six independently-developed SNMPv3 security feature implementations were shown interoperating. Some participants showed both command generator (manager) and command responder (agent) applications. All demonstrated authentication using HMAC-MD5 and privacy using CBC-DES. Most also showed HMAC-SHA authentication as well as remote configuration, but a few had not yet completed these features by show time. Participants could be readily identified by their SNMPv3 caps and "Practice Safe Sets" buttons.

Advent Network Management showed interoperability using their Java JDK 1.1-based SNMPv3 MIB Browser. Bay Networks showed both command generator and responder applications. Bay's multilingual agent for their BayStack 200 hub showed different levels of authentication and privacy based on the SNMP Research stack, working with command generators (managers) using other code bases. Bay's Optivity manager applications worked with other vendors' code bases on network hardware in the booth. BMC Software demonstrated interoperation with authentication, encryption and remote configuration features developed in C for their PATROL SNMP Toolkit and Patrol product suite. Cisco Systems demonstrated interoperability between their implementation and other code bases, using the SNMP Research-based C-language command responder capability running on their Cisco 2500 platform.

HP demonstrated OpenView Network Node Manager interoperation of secure SNMPv3 authentication, privacy and remote administration with command responders. The SNMPv3 manager is implemented with a hook that allows the SNMP Research management stack to translate SNMPv1/v2c requests into SNMPv3 before sending the request out on the wire. IBM Networking product division demonstrated authentication and privacy interoperation using an OS/390 Unix agent written in C, running in Dallas, which was remotely configurable, and the Nways Workgroup Manager for NT, written in Java. Liebert Corporation interoperated using a monitoring and control agent for their UPStation GX based on the SNMP Research stack.

SNMP Research's SNMPv3 product line demonstrated authentication and encryption as well as remote configuration, interoperating with other code bases as well as with their own code base in other vendors' products and prototypes. Tivoli demonstrated interoperation of authentication and privacy using their Java-based SNMPv3 Browser. Omar Cherkaoui and Ylian Saint-Hilaire from the University of Quebec in Montreal demonstrated interoperation using their Java-based reference implementation, including an SNMPv3 proxy, to be licensed for non-commercial use.

Not confining themselves to the Hot Spot, Hot Spot vendors also demonstrated interoperation with Epilogue Technology's SNMPv3 code on the show floor and with SNMPv1 devices elsewhere in the show.

One User's Perspective

Rob Frye said that he "was pleased to see vendors there really supporting SNMPv3." Some vendors gave off-the-record tentative dates for shipping products (details of which can't be revealed in this article), but most stayed cautiously away from date commitments. Unfortunately, most network equipment and network management vendors do not yet seem to have plans for support of SNMPv3; there is concern about whether it will be adopted quickly or suffer the lack of acceptance that SNMPv2 has. The vendors that support SNMPv3 show hope in the stability of the standards and the pace of progress of the v3 Working Group.

Of course, large carriers such as MCI are very interested in SNMP Version 3. Widespread support of "confirmed Traps" via the Inform PDU, 64-bit counters (for use on high-speed interfaces or in situations where frequent polling is not feasible), and the use of GetBulk for large table data retrieval can make an immediate difference in managing large-scale carrier-grade networks. Although these features exist in SNMPv2, the multiple versions of SNMPv2 that have led to a lack of consistent acceptance have kept these capabilities out of many systems and networks. The possibility of a secure Set mechanism to securely replace the use of (scripted) Telnet, particularly for customer service delivery, will take longer to implement than the other features, but will allow carriers such as MCI to improve upon service activation times. When SNMPv3 is widely supported, getting it into the network and management systems may be a little difficult but should pose no significant barrier. Carriers and users are used to rolling version migrations where multiple versions of software co-exist for some time. The forthcoming Coexistence and Transition RFC should help guide the way to smooth transitions between SNMPv1 (and v2) to SNMPv3.

The users and vendors look forward to future technology showcases on SNMPv3, the continued IETF Working Group efforts to finalize the standard documents (along with the various proposed enhancements being discussed), and further announcements of SNMPv3-capable products.

One Participant's Perspective

Participants were pleased to have had the opportunity to demonstrate to so many show attendees that SNMPv3 is real; that it has support among toolkit providers, ISVs and OEMs; and that the RFCs are sufficiently clear, detailed and complete to permit these implementations to interoperate. Participants explained the status and progression of the protocol; the relationship it has to SNMPv1 and SNMPv2; and their feeling that work on it is progressing well, moving steadily towards achieving Standard status. Ajay Gummadi of Bay Networks said he was "... happy that the various parties to SNMPv2 have finally cast aside their differences and are unified in supporting SNMPv3." Several participants mentioned they were enthusiastic about talking to users directly, and especially to hear first-hand about their needs.

Participants brought command responder and command generator applications, including products, prototypes and works in progress. Advent Net, IBM, Tivoli and University of Quebec showed Java implementations. As you might expect from a technology showcase and a demonstration of work in progress, several companies took advantage of the Hot Spot to identify and fix a bug or two in their code, increasing the event's overall interoperability as the show continued.

Many of the booth's visitors expressed both surprise and pleasure at seeing 10 companies with SNMPv3 security implementations and, further, interoperating code. Hot Spot participants noted excitement by some visitors and a wait-and-see attitude by others; but many attendees with a skeptical attitude indicated they now believe SNMPv3 deserves a serious look. John Seligson of Bay Networks recalled, "Many visitors asked what happened to SNMPv2. Once I explained that SNMPv3 incorporated the standardized aspects of SNMPv2 (i.e., SMIv2, new protocol operations, etc.) adding an intuitive user-based security and administrative framework they went away satisfied."

We were pleased and encouraged to see visitors representing a broad range of companies and organizations, notably the telecom industry and universities. We heard that users understand that community-based security is not sufficient. Kevin Dwinnell of Liebert said, "It is critical for customers to protect control over their network devices" and applications. Some attendees expressed the need to use SNMPv3 security for the public components of their networks, even when they use SNMPv1 or SNMPv2c for the private components. Many were gratified to see SNMPv3's simplified administration. John Seligson said, "Many visitors ... asked whether [the technology] would be appearing in products soon. ... I talked with several managers of very large networks who said that they would like to deploy as soon as possible." According to Cisco's Ram Kavasseri, "Current Cisco customers were very interested in the planned release date for SNMPv3 functionality on Cisco routers, and the availability of SNMPv3-capable management platforms ... and applications." Kavasseri added, "Response from booth visitors was extremely favorable. Major questions involved deploying of passwords across networks, and difficulty in debugging packets with the privacy mechanism enabled."

While many visitors were well-informed about the protocol and its progress, any number of visitors used the Hot Spot to gather basic information. "This was the way Interop used to be a few years ago," said Muriel Appelbaum of BMC Software, "when a show attendee could just walk in and ask for a demo or ask a detailed question and get as much technical information as they wanted."

Staffing the Hot Spot was productive and enjoyable because all the participants were helpful and cooperative. Bert Wijnen IETF Operations and Management Area Director, noted he is "very encouraged with the number of interoperating implementations and with the positive spirit [shown in this] unified presentation ... of a single technology. [This shows SNMP is] back on track and moving forward."

Please contact the respective organizations and vendors above for detailed information on their product plans and availability and take a look at the SNMPv3 web page at http://www.ibr.cs.tu-bs.de/projects/snmpv3/.

Key Vendors Support SNMPv3

David Reid, SNMP Research

SNMPv3 is writing a new chapter in the Internet Standard Management Framework story, even as SNMP continues to thrive as the cornerstone of today's enterprise management systems. Key vendors are already supporting SNMPv3 in products available today.

At the October 1998 Networld+Interop in Atlanta, SNMPv3 with Security and Administration was again the focus of a Hot Spot. Hot Spots at NetWorld+Interop focus on educating attendees about the latest in interoperable, standards-based technologies. At this event, key vendors demonstrated their implementations of the recently published SNMPv3. The demonstrations highlighted key features of the third version of the Internet-Standard Management Framework which now includes commercial-grade security and a robust administrative framework with remote configuration.

The October SNMPv3 Hot Spot in Atlanta was very similar to the highly successful SNMPv3 Hot Spot at Networld+Interop in May of 1998 in Las Vegas. Both demonstrated multiple interoperable implementations of SNMPv3, increased attendee knowledge of the capabilities, built enthusiasm for the new technology, and showed the strong vendor support for SNMPv3.

There were also a number of differences. Most notably, while the Hot Spot at Networld+Interop in May of 1998 was primarily a technology demonstration, the Hot Spot in Atlanta in October was primarily a products demonstration by the following participating companies:

- IBM demonstrated the currently shipping Nways Workgroup Manager which supports SNMPv3 today. IBM also demonstrated eNetwork Communications Server for OS/390 which will support SNMPv3 early 1999.
- SNMP Research International showed their development toolkits and end-user products for Network, System, and Application Management. All SNMP Research products support SNMPv3 today. Several other Hot Spot participants including Cisco, Bay Networks, HP, and Liebert were demonstrating products based on technology licensed from SNMP Research International.
- InterWorking Labs demonstrated their latest SNMP test suite, SilverCreek 6.0, which now includes over 100 tests for SNMPv3. The SNMP test suite is designed to insure interoperability and compliance to the standard. SilverCreek 6.0, with support for SNMPv3, is now available.
- Bay Networks, a Nortel Networks business, will be shipping SNMPv3 technology in a select number of Workgroup Switching products soon. Bay Networks also plans to include SNMPv3 as a core component of future release versions of Optivity NMS and related applications.
- AdventNet's SNMPv3 Java product, currently in beta testing, is expected to be shipping by the time this article is published.
- Hewlett-Packard provides SNMPv3 support today for HP OpenView Network Node Manager through a partnership with SNMP Research International.
- Cisco demonstrated SNMPv3 in their 2500 and 4500 routers. Cisco has incorporated SNMPv3 into release 12.0(3)T of IOS. This release is expected to start shipping to customers in the Spring of 1999.
- Liebert Corporation demonstrated SNMPv3 in their uninteruptable power supplies. Liebert plans to start shipping SNMPv3 in products in 1999.
- The University of Quebec in Montreal has a public domain implementation of SNMPv3 available now for download from their web site.

In addition to the Hot Spot participants, many other vendors are also working on SNMPv3 products. More information about SNMPv3, including links to the above listed companies and technical information about SNMPv3, is available on the SNMPv3 Hot Spot web page at http://www.snmp.com/v3hotspot/.

Questions Answered

David T. Perkins, SNMPinfo

The SNMPv3 framework and document set has seen increasing attention over the last few months. This has resulted in questions about what will happen to SNMPv1. This month's column will answer some of those questions.

Will SNMPv1 become obsolete when SNMPv3 is advanced by the IETF?

First, let's review the status of IETF documents. IETF documents are published in RFC series. Each document is given a number and a status. Most, but not all RFCs, are on the "standards track." A specification enters the standards track with a label of *Proposed* standard. After meeting IETF specified requirements it can be advanced and be given the *Draft* standard status. And after it meets all IETF requirements, it is advanced to become an IETF *Standard* (which is also called *Full Standard* status). If a specification is replaced by a newer version, the older one is given a status of *Obsolete*. If a specification is replaced by newer technology, the older technology specifications have their status changed to *Historical*. The details of the IETF standards process are found in RFC 2026.

The IETF standards process classifies documents as a Technical Specification (TS) or an Applicability Specification (AS). A TS is "any description of a protocol, service, procedure, convention, or format." An AS describes "how, and under what circumstances, one or more TSs may be applied to support a particular Internet capability." An AS specifies a requirement level to each TS to which it refers. The levels are:

- Required - implementation of the TS is required for minimal conformance with the AS
- Recommended - implementation is not required, but is desired
- Elective - implementation is optional
- Limited Use - the TS is for use in limited or unique circumstances, such as when the TS is *Experimental* and not standards-track
- Not Recommended - the TS is not for general use, such as when it has limited functionality or is historic

Conceptually, ASs and TSs are separate documents. In practice, a standards-track document may be a combination of an AS and one or more referenced TSs. The latest version of the *Internet Official Protocol Standards* RFC (as of this writing RFC 2400) specifies the status and requirement level of each RFC.

Now, let's review what is defined by each version of SNMP. The SNMPv1 management framework includes documents that define the SNMPv1 management protocol (RFC 1157), the structure of management information (SMIv1) (RFC 1155, RFC 1212, and RFC 1215), and an initial set of managed objects (RFC 1213) and events (RFC 1215). The SMIv1 specifies the base data types for managed objects. It also defines a language for defining managed objects, events, refinements to the base data types, and OID values. Finally, SMIv1 contains several administrative assignments of OID values.

There are two additional frameworks for SNMP, which are SNMPv2 and SNMPv3. Both of these frameworks define similar, but "incompatible on the wire" versions of the SNMP protocol. An improved version of the SMI, called SMIv2, is defined in the SNMPv2 framework that is also used by the SNMPv3 framework. Neither the SNMPv2 nor the SNMPv3 frameworks replace the initial set of objects and events defined in the SNMPv1 framework. However, each version defines additional objects used to manage the SNMP protocol. The SNMPv3 framework contains a large number of objects that can be used to remotely configure the administrative aspects of SNMP entities, which include those supporting SNMPv1, SNMPv2c, and SNMPv3.

Independently from the frameworks, the initial set of objects and events defined in RFC 1213 and RFC 1215 were split into separate documents.

So, to answer the question "What will happen to SNMPv1?," we need to break the question into three parts, which are:

1. What will happen to the SNMPv1 protocol?
2. What will happen to the MIB module language defined in SMIv1?
3. What will happen to the MIB modules?

Here is what has occurred, and the best guess as to what will happen.

What will happen to the SNMPv1 protocol?
The SNMPv1 protocol is currently supported by many devices. The SNMPv3 framework supports the notion of multi-lingual devices. Thus, my best guess is that the SNMPv1 protocol (which currently has a "Recommended" requirement level) will live on for a long time and its definition, RFC 1157, will not be "retired" and made *Historic* soon. (However, there will be some political maneuvering for it to be made *Historic* or for its requirement level to be changed to "Elective" or even "Limited Use". The market will be what determines the time table for advancing SNMPv3 to *[Full] Standard* status and the change of status for SNMPv1.)

What will happen to the SMIv1 language?
SMIv2 (which is currently at the "Elective" requirement level) offers much improvement over SMIv1 (which is currently at the "Recommended" requirement level). SMIv2 has been approved for advancement to *[Full] Standard* status, but has not been published as RFCs at the time this article was written. Even before this

upgrade in status, all MIBs defined in new RFCs were required to be written in the SMIv2 format. As RFCs containing MIBs in the SMIv1 format are updated, the MIBs are converted to the SMIv2 format. However, this conversion could not be done for documents to be advanced to *[Full] Standard* status because SMIv2 was not yet at *[Full] Standard* status. Now that SMIv2 has advanced, all MIBs in RFCs will be converted to the SMIv2 format when they are updated. Currently, most of the major vendors of SNMP management systems that contain MIB compilers support SMIv2. Thus, the best guess is that the SMIv1 documents will be retired to *Historic* status (with requirement level "Not Recommended") now that SMIv2 documents have been advanced to *[Full] Standard* status and given the requirement level of "Recommended."

What will happen to the MIB modules?

The initial set of objects defined in RFC 1213 and RFC 1215 have been replaced by RFCs that define individual groups found in RFC 1213 and RFC 1215. When all of these replacement documents are advanced to *[Full] Standard* status, then RFC 1213 and RFC 1215 will be retired to *Historic* status with requirement level of "Not Recommended." The replacement RFCs currently have a requirement level of "Elective," which will be changed to "Recommended" when the replacement RFCs are advanced. The replacement RFCs are:

- RFC 1907 - contains the `system` and `snmp` groups
- RFC 2233 - contains the `interfaces` group
- RFC 2011 - contains the `ip` group (except for the IP routing table)
- RFC 2096 - contains a replacement for the IP routing table
- RFC 2012 - contains the `tcp` group
- RFC 2013 - contains the `udp` group
- The `at` group of RFC 1213 is not replaced, since it is *deprecated*.
- The `egp` group of RFC 1213 is not replaced since it is obsolete because EGP is *Historic*.

Is the MIB module language a proper subset of ASN.1 and can ASN.1 tools be used on SNMP MIB modules?

The MIB module language is not a proper subset of ASN.1. It has never been and will never be. There are two versions of the MIB module language, defined in SMIv1 and SMIv2. Both use elements of ASN.1, but are not ASN.1 subsets. ASN.1 works well in defining the formats of messages and is used to define the format of SNMP messages. ASN.1 tools can be used on the definitions of SNMP messages to generate code to encode and decode SNMP messages. However, none of the leading vendors of SNMP toolkits (or freely available SNMP toolkits) use ASN.1 tools in order to process MIB modules. There have been many stories about developers new to SNMP trying to use ASN.1 tools to create management applications and agents. The outcomes have been disaster. The SNMP MIB module language is far removed from the "normal" usage of ASN.1, and, thus, ASN.1 tools are of practically no value in processing SNMP MIB specifications.

The latest update of the SMI specifications, which should be published soon, clarify the differences between ASN.1 and the MIB module language. However, these SMI specifications still require the reader to have a copy of the 1998 version of ASN.1 handy for reference.

How do you include a sequence within another sequence?

This is an old and frequently asked question, with the label "table in a table." The answer to the question is that it is not possible to directly have such a construct! However, you can achieve the same result by defining two tables in your SNMP MIB module.

Say, you have in the C language a struct definition like the following:

```
struct myStruct {
  int a;
  int b[10];
  int c;
  } myTab[20];
```

To turn this into SNMP MIB definitions, you would need two tables:

```
myfirstTable OBJECT-TYPE
  SYNTAX SEQUENCE OF MyFirstEntry
    ...

myFirstEntry OBJECT-TYPE
  SYNTAX MyFirstEntry
    ...
  INDEX { i1 }
    ...

MyFirstEntry ::= SEQUENCE {
  i1 Integer32,
  a  Integer32,
```

```
   c  Integer32 }

<< definitions for objects i1, a, and c >>

mySecondTable OBJECT-TYPE
  SYNTAX SEQUENCE OF MySecondEntry
   ...

mySecondEntry OBJECT-TYPE
  SYNTAX MySecondEntry
   ...
  INDEX { i1, i2 } -- i1 is from the first, i2
                   -- is from the second table
   ...

 MySecondEntry ::= SEQUENCE {
   i2 Integer32,
   b  Integer32 }

<< definitions for objects i2 and b >>
```

Editor's Comment

Jürgen Schönwälder, TU Braunschweig
Aiko Pras, University of Twente

A year has passed between the last issue of *The Simple Times* and the issue you are reading right now. There are a number of reasons for this delay. Many of them have to do with recent activities within the IETF.

SMIv2 approved as Standard

In June 1998, a "design team" was chartered to advance the SMIv2 (RFC 1902, RFC 1903, RFC 1904) from Draft Standard to [Full] Standard. The advancement to [Full] Standard is necessary in order to allow MIB specifications to advance to [Full] Standard status. The design team worked through a list of 75 issues related to the current SMIv2 specifications. The changes proposed by the design team were reviewed several times and the IESG finally approved the SMIv2 revisions as [Full] Standard in January 1999. Publication of the revised RFCs can be expected in the coming months.

SNMPv3 approved as a Draft Standard

The SNMPv3 specifications, which were published in January 1998 as Proposed Standards, have been revised during the last months. The revised specifications have been approved as Draft Standards by the IESG in February 1999. The publication of the RFCs can be expected any time soon. The Draft Standards status indicates a strong belief that the SNMPv3 specifications are mature and will be useful. This means that it is reasonable for vendors to deploy implementations of SNMPv3 into disruption sensitive environments. More information can be found on the SNMPv3 Web page at http://www.ibr.cs.tu-bs.de/projects/snmpv3/.

IRTF Research Groups

Two research groups have been formed within the Internet Research Task Force (IRTF) in order to address some of the long term management problems. The Services Management Research Group (NSM) is chartered to investigate new architectures, information models, and supporting protocols to enable the convergence of network and system management into a common service management framework.

The Network Management Research Group (NMRG) will work on solutions for network management problems that are not yet considered well understood enough for engineering work within the IETF. The initial focus will be on higher-layer management services that interface with the current Internet management framework. This includes communication services between management systems, which may belong to different management domains, as well as customer-oriented management services.

We can expect to hear more about these research groups and the work they are doing in the future. This issue of *The Simple Times* already includes an article about bulk transfers of MIB data. It is the result of an ad-hoc meeting which led to the formation of the NMRG.

Operations and Management Area News

The "Operations and Management Area" (OPS) of the IETF got a new area director. Randy Bush (Verio) was selected to take over the position previously held by Harald Alvestrand (Maxware). He will now supervise the work within the OPS area together with the second area director Bert Wijnen (IBM Research). Harald Alvestrand was selected as a new member of the Internet Architecture Board (IAB).

The OPS Web server (http://www.ops.ietf.org/) provides guidelines for authors of IETF MIB modules. It also has a Web page which allows to track the progression of OPS related Internet-Drafts through the IESG.

Finally, there are two new public OPS mailing lists: The ops-area@ops.ietf.org mailing list is intended for general discussions relevant to the OPS area. The mibs@ops.ietf.org mailing list is for discussions related to MIB development. To subscribe, send a message to the corresponding `-request` address with `subscribe` in the body.

Standards Summary

Please consult the latest version of *Internet Official Protocol Standards*. As of this writing, the latest version is RFC 2400.

SMIv1 Data Definition Language

Full Standards:

- RFC 1155 - Structure of Management Information
- RFC 1212 - Concise MIB Definitions

Informational:

- RFC 1215 - A Convention for Defining Traps

SMIv2 Data Definition Language

Draft Standards:

- RFC 1902 - Structure of Management Information
- RFC 1903 - Textual Conventions
- RFC 1904 - Conformance Statements

SNMPv1 Protocol

Full Standards:

- RFC 1157 - Simple Network Management Protocol

Proposed Standards:

- RFC 1418 - SNMP over OSI
- RFC 1419 - SNMP over AppleTalk
- RFC 1420 - SNMP over IPX

SNMPv2 Protocol

Draft Standards:

- RFC 1905 - Protocol Operations for SNMPv2
- RFC 1906 - Transport Mappings for SNMPv2
- RFC 1907 - MIB for SNMPv2
- RFC 1908 - Coexistence between SNMPv1 and SNMPv2

Experimental:

- RFC 1901 - Community-based SNMPv2
- RFC 1909 - Administrative Infrastructure
- RFC 1910 - User-based Security Model

SNMPv3 Protocol

Draft Standards:

- RFC 1905 - Protocol Operations for SNMPv2
- RFC 1906 - Transport Mappings for SNMPv2
- RFC 1907 - MIB for SNMPv2

Proposed Standards:

- RFC 2271 - Architecture for Describing SNMP Management Frameworks
- RFC 2272 - Message Processing and Dispatching
- RFC 2273 - SNMPv3 Applications
- RFC 2274 - User-based Security Model
- RFC 2275 - View-based Access Control Model

SNMP Agent Extensibility

Proposed Standards:

- RFC 2257 - AgentX Protocol Version 1

SMIv1 MIB Modules

Full Standards:

- RFC 1213 - Management Information Base II
- RFC 1643 - Ethernet-Like Interface Types MIB

Draft Standards:

- RFC 1493 - Bridge MIB
- RFC 1559 - DECnet phase IV MIB
- RFC 1757 - Remote Network Monitoring MIB

Proposed Standards:

- RFC 1285 - FDDI Interface Type (SMT 6.2) MIB
- RFC 1381 - X.25 LAPB MIB
- RFC 1382 - X.25 Packet Layer MIB
- RFC 1414 - Identification MIB
- RFC 1461 - X.25 Multiprotocol Interconnect MIB
- RFC 1471 - PPP Link Control Protocol MIB
- RFC 1472 - PPP Security Protocols MIB
- RFC 1473 - PPP IP NCP MIB

- RFC 1474 - PPP Bridge NCP MIB
- RFC 1512 - FDDI Interface Type (SMT 7.3) MIB
- RFC 1513 - RMON Token Ring Extensions MIB
- RFC 1514 - Host Resources MIB
- RFC 1515 - IEEE 802.3 MAU MIB
- RFC 1525 - Source Routing Bridge MIB
- RFC 1742 - AppleTalk MIB

SMIv2 MIB Modules

Draft Standards:

- 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 Type MIB
- RFC 1724 - RIP version 2 MIB
- RFC 1748 - IEEE 802.5 Interface Type MIB
- RFC 1850 - OSPF version 2 MIB
- RFC 1907 - SNMPv2 MIB
- RFC 2115 - Frame Relay DTE Interface Type MIB

Proposed Standards:

- RFC 1567 - X.500 Directory Monitoring MIB
- RFC 1604 - Frame Relay Service MIB
- RFC 1611 - DNS Server MIB
- RFC 1612 - DNS Resolver MIB
- RFC 1628 - Uninterruptible Power Supply MIB
- RFC 1666 - SNA NAU MIB
- RFC 1696 - Modem MIB
- RFC 1697 - RDBMS MIB
- RFC 1747 - SNA Data Link Control MIB
- RFC 1749 - 802.5 Station Source Routing MIB
- RFC 1759 - Printer MIB
- RFC 2006 - Internet Protocol Mobility MIB
- RFC 2011 - Internet Protocol MIB
- RFC 2012 - Transmission Control Protocol MIB
- RFC 2013 - User Datagram Protocol MIB
- RFC 2020 - IEEE 802.12 Interfaces MIB
- RFC 2021 - RMON Version 2 MIB
- RFC 2024 - Data Link Switching MIB
- RFC 2037 - Entity MIB
- RFC 2051 - APPC MIB
- RFC 2074 - RMON Protocol Identifier
- RFC 2096 - IP Forwarding Table MIB
- RFC 2108 - IEEE 802.3 Repeater MIB
- RFC 2127 - ISDN MIB
- RFC 2128 - Dial Control MIB
- RFC 2206 - Resource Reservation Protocol MIB
- RFC 2213 - Integrated Services MIB
- RFC 2214 - Guaranteed Service MIB
- RFC 2232 - Dependent LU Requester MIB
- RFC 2233 - Interfaces Group MIB
- RFC 2238 - High Performance Routing MIB
- RFC 2239 - IEEE 802.3 MAU MIB
- RFC 2248 - Network Services Monitoring MIB
- RFC 2249 - Mail Monitoring MIB
- RFC 2266 - IEEE 802.12 Repeater MIB
- RFC 2271 - SNMP Framework MIB
- RFC 2272 - SNMPv3 MPD MIB
- RFC 2273 - SNMP Applications MIB
- RFC 2274 - SNMPv3 USM MIB
- RFC 2275 - SNMP VACM MIB
- RFC 2287 - System-Level Application Mgmt MIB
- RFC 2320 - Classical IP and ARP over ATM MIB
- RFC 2358 - Ethernet-Like Interface Types MIB
- RFC 2366 - Multicast over UNI 3.0/3.1 / ATM MIB

- RFC 2452 - IPv6 UDP MIB
- RFC 2454 - IPv6 TCP MIB
- RFC 2455 - APPN MIB
- RFC 2456 - APPN Trap MIB
- RFC 2457 - APPN Extended Border Node MIB
- RFC 2465 - IPv6 Textual Conventions and MIB
- RFC 2466 - ICMPv6 MIB
- RFC 2493 - 15 Minute Performance History TCs
- RFC 2494 - DS0, DS0 Bundle Interface Type MIB
- RFC 2495 - DS1, E1, DS2, E2 Interface Type MIB
- RFC 2496 - DS3/E3 Interface Type MIB
- RFC 2512 - Accounting MIB for ATM Networks
- RFC 2513 - Accounting Control MIB
- RFC 2514 - ATM Textual Conventions and OIDs
- RFC 2515 - ATM MIB
- RFC 2558 - SONET/SDH Interface Type MIB

IANA Maintained MIB Modules

- Interface Type Textual Convention (IANAifType)
  ftp://ftp.iana.org/mib/ianaiftype.mib

Related Documents

Informational:

- RFC 1270 - SNMP Communication Services
- RFC 1321 - MD5 Message-Digest Algorithm
- RFC 1470 - Network Management Tool Catalog
- RFC 2039 - Applicability of Standard MIBs to WWW Server Management
- RFC 2089 - Mapping SNMPv2 onto SNMPv1 within a bi-lingual SNMP agent

Experimental:

- RFC 1187 - Bulk Table Retrieval with the SNMP
- RFC 1224 - Techniques for Managing Asynchronously Generated Alerts
- RFC 1238 - CLNS MIB
- RFC 1592 - SNMP Distributed Program Interface
- RFC 1792 - TCP/IPX Connection MIB Specification
- RFC 2064 - Traffic Flow Measurement: Meter MIB

Recent Publications

Building Network Management Tools with Tcl/Tk

- Authors:
  Dave Zeltserman `<davez9@gte.net>`
  Gerard Puoplo `<puoplo@shore.net>`
- Publisher: Prentice Hall
  `http://www.prenhall.com/`
- ISBN: 0-13-080727-3
- Available: April, 1998

This book shows how to build custom network management tools using the scripting language Tcl/Tk. The source code of several example applications for response time monitoring, network discovery, IP path tracing, web-based status monitoring and RMON2 configuration is explained in detail.

RMON: Remote Monitoring of SNMP-Managed LANs

- Author: Dave Perkins `<dperkins@dsperkins.com>`
- Publisher: Prentice Hall
  `http://www.prenhall.com/`
- ISBN: 0-13-096163-9
- Available: September, 1998

This book translates the SNMP and RMON terminology into business terms so you can communicate with your network management product vendors. In addition, the book provides a close-up review of the RMON standards and walks you through the objects found in the RMON MIB modules. The book contains many diagrams and figures to illustrate the essential elements of the RMON standards and the key SNMP concepts required to use RMON.

SNMP-based ATM Network Management

- Author: H. Pan
- Publisher: Artech House
  `http://www.artech-house.com/`
- ISBN: 0-89-006983-2
- Available: September, 1998

This book explains the fundamentals of the ATM network and the SNMP protocol. It overviews the details of the Physical and ATM Layer, LANE, PNNI, and shows how different standard MIBs and proprietary MIBs are used together to manage an ATM Switch.

SNMP, SNMPv2, SNMPv3, and RMON 1 and 2

- Author: William Stallings <ws@shore.net>
- Publisher: Addison-Wesley
  http://www.aw.com/
- ISBN: 0-201-48534-6
- Available: December, 1998

A comprehensive treatment of SNMP-based standards, including a description of the protocols, MIBs, and practical issues. Covers SNMPv1, SNMPv2c and SNMPv3, the original RMON1, and the current version of RMON2.

TMN Telecommunications Management Network

- Author: Divakara Udupa
- Publisher: McGraw-Hill
  http://www.mcgraw-hill.com/
- ISBN: 0-07-065815-3
- Available: January, 1999

This book is a comprehensive study of TMN architectures, functions, capabilities and requirements. It also contains a chapter on SNMP management.

A Practical Guide to SNMPv3 and Network Management

- Author: Dave Zeltserman <davez9@gte.net>
- Publisher: Prentice Hall
  http://www.prenhall.com/
- ISBN: 0-13-021453-1
- Available: May, 1999

A guide to SNMPv3 from the viewpoint of a management application writer. This book describes in detail how the remote configuration capabilities of SNMPv3 can be used to automate SNMPv3 administration.

Calendar and Announcements

IETF Meetings:

- 44th Meeting of the IETF
  March 15-19, 1999, Minneapolis, MN, USA
- 45th Meeting of the IETF
  July 12-16, 1999, Oslo, Norway
- 46th Meeting of the IETF
  November 8-12, 1999, Washington, DC, USA

Conferences and Workshops:

- Integrated Network Management '99
  May 22-28, 1999, Boston, MA, USA
- Distributed Systems Operations & Management '99
  October 11-13, 1999, Zurich, Switzerland
- Network Operations and Management Symposium 2000
  April 10-14, 2000, Honolulu, Hawaii, USA

Exhibitions and Trade Shows:

- NetWorld + Interop Singapore
  April 5-9, 1999, Singapore, Singapore
- NetWorld + Interop London
  April 19-21, 1999, London, UK
- NetWorld + Interop Las Vegas
  May 10-14, 1999, Las Vegas, USA
- NetWorld + Interop Tokio
  May 31- June 4, 1999, Tokio, Japan
- NetWorld + Interop Toronto
  July 14-16, 1999, Canada
- NetWorld + Interop Paris
  September 14-17, 1999, Paris, France
- NetWorld + Interop Atlanta
  September 13-17, 1999, Atlanta, USA
- NetWorld + Interop Sao Paulo
  November 9-12, 1999, Sao Paulo, Brazil
- NetWorld + Interop Sydney
  November 15-19, 1999, Sydney, Australia

Publication Information

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 version 1.0. 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 HTML, PDF and PostScript. New issues are announced via an electronic mailing list. Send electronic mail to

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

with

```
subscribe simple-times
```

in the body if you want to subscribe to this list. Back issues are available via *The Simple Times* Web server:

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