Network Working Group
Request for Comments: 1155
Obsoletes: RFC 1065

M. Rose
Performance Systems International
K. McCloghrie
Hughes LAN Systems
May 1990

Structure and Identification of Management Information for TCP/IP-based Internets

Table of Contents

1. Status of this Memo

This RFC is a re-release of RFC 1065, with a changed "Status of this Memo", plus a few minor typographical corrections. The technical

content of the document is unchanged from RFC 1065.

This memo provides the common definitions for the structure and identification of management information for TCP/IP-based internets. In particular, together with its companion memos which describe the management information base along with the network management protocol, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular, the Internet.

This memo specifies a Standard Protocol for the Internet community. Its status is "Recommended". TCP/IP implementations in the Internet which are network manageable are expected to adopt and implement this specification.

The Internet Activities Board recommends that all IP and TCP implementations be network manageable. This implies implementation of the Internet MIB (RFC-1156) and at least one of the two recommended management protocols SNMP (RFC-1157) or CMOT (RFC-1095). It should be noted that, at this time, SNMP is a full Internet standard and CMOT is a draft standard. See also the Host and Gateway Requirements RFCs for more specific information on the applicability of this standard.

Please refer to the latest edition of the "IAB Official Protocol Standards" RFC for current information on the state and status of standard Internet protocols.

Distribution of this memo is unlimited.

2. Introduction

This memo describes the common structures and identification scheme for the definition of management information used in managing TCP/IP-based internets. Included are descriptions of an object information model for network management along with a set of generic types used to describe management information. Formal descriptions of the structure are given using Abstract Syntax Notation One (ASN.1) [1].

This memo is largely concerned with organizational concerns and administrative policy: it neither specifies the objects which are managed, nor the protocols used to manage those objects. These concerns are addressed by two companion memos: one describing the Management Information Base (MIB) [2], and the other describing the Simple Network Management Protocol (SNMP) [3].

This memo is based in part on the work of the Internet Engineering

Task Force, particularly the working note titled "Structure and Identification of Management Information for the Internet" [4]. This memo uses a skeletal structure derived from that note, but differs in one very significant way: that note focuses entirely on the use of OSI-style network management. As such, it is not suitable for use with SNMP.

This memo attempts to achieve two goals: simplicity and extensibility. Both are motivated by a common concern: although the management of TCP/IP-based internets has been a topic of study for some time, the authors do not feel that the depth and breadth of such understanding is complete. More bluntly, we feel that previous experiences, while giving the community insight, are hardly conclusive. By fostering a simple SMI, the minimal number of constraints are imposed on future potential approaches; further, by fostering an extensible SMI, the maximal number of potential approaches are available for experimentation.

It is believed that this memo and its two companions comply with the guidelines set forth in RFC 1052, "IAB Recommendations for the Development of Internet Network Management Standards" [5] and RFC 1109, "Report of the Second Ad Hoc Network Management Review Group" [6]. In particular, we feel that this memo, along with the memo describing the management information base, provide a solid basis for network management of the Internet.

3. Structure and Identification of Management Information

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using Abstract Syntax Notation One (ASN.1) [1].

Each type of object (termed an object type) has a name, a syntax, and an encoding. The name is represented uniquely as an OBJECT IDENTIFIER. An OBJECT IDENTIFIER is an administratively assigned name. The administrative policies used for assigning names are discussed later in this memo.

The syntax for an object type defines the abstract data structure corresponding to that object type. For example, the structure of a given object type might be an INTEGER or OCTET STRING. Although in general, we should permit any ASN.1 construct to be available for use in defining the syntax of an object type, this memo purposely restricts the ASN.1 constructs which may be used. These restrictions are made solely for the sake of simplicity.

The encoding of an object type is simply how instances of that object type are represented using the object's type syntax. Implicitly tied to the notion of an object's syntax and encoding is how the object is represented when being transmitted on the network. This memo specifies the use of the basic encoding rules of ASN.1 [7].

It is beyond the scope of this memo to define either the MIB used for network management or the network management protocol. As mentioned earlier, these tasks are left to companion memos. This memo attempts to minimize the restrictions placed upon its companions so as to maximize generality. However, in some cases, restrictions have been made (e.g., the syntax which may be used when defining object types in the MIB) in order to encourage a particular style of management. Future editions of this memo may remove these restrictions.

3.1. Names

Names are used to identify managed objects. This memo specifies names which are hierarchical in nature. The OBJECT IDENTIFIER concept is used to model this notion. An OBJECT IDENTIFIER can be used for purposes other than naming managed object types; for example, each international standard has an OBJECT IDENTIFIER assigned to it for the purposes of identification. In short, OBJECT IDENTIFIERs are a means for identifying some object, regardless of the semantics associated with the object (e.g., a network object, a standards document, etc.)

An OBJECT IDENTIFIER is a sequence of integers which traverse a

global tree. The tree consists of a root connected to a number of labeled nodes via edges. Each node may, in turn, have children of its own which are labeled. In this case, we may term the node a subtree. This process may continue to an arbitrary level of depth. Central to the notion of the OBJECT IDENTIFIER is the understanding that administrative control of the meanings assigned to the nodes may be delegated as one traverses the tree. A label is a pairing of a brief textual description and an integer.

The root node itself is unlabeled, but has at least three children directly under it: one node is administered by the International Organization for Standardization, with label iso(1); another is administrated by the International Telegraph and Telephone Consultative Committee, with label ccitt(0); and the third is jointly administered by the ISO and the CCITT, joint-iso-ccitt(2).

Under the iso(1) node, the ISO has designated one subtree for use by other (inter)national organizations, org(3). Of the children nodes present, two have been assigned to the U.S. National Institutes of Standards and Technology. One of these subtrees has been transferred by the NIST to the U.S. Department of Defense, dod(6).

As of this writing, the DoD has not indicated how it will manage its subtree of OBJECT IDENTIFIERs. This memo assumes that DoD will allocate a node to the Internet community, to be administered by the Internet Activities Board (IAB) as follows:

```
internet    OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
```

That is, the Internet subtree of OBJECT IDENTIFIERs starts with the prefix:

```
1.3.6.1.
```

This memo, as a standard approved by the IAB, now specifies the policy under which this subtree of OBJECT IDENTIFIERs is administered. Initially, four nodes are present:

```
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

3.1.1. Directory

The directory(1) subtree is reserved for use with a future memo that discusses how the OSI Directory may be used in the Internet.

3.1.2. Mgmt

The mgmt(2) subtree is used to identify objects which are defined in IAB-approved documents. Administration of the mgmt(2) subtree is delegated by the IAB to the Internet Assigned Numbers Authority for the Internet. As RFCs which define new versions of the Internet-standard Management Information Base are approved, they are assigned an OBJECT IDENTIFIER by the Internet Assigned Numbers Authority for identifying the objects defined by that memo.

For example, the RFC which defines the initial Internet standard MIB would be assigned management document number 1. This RFC would use the OBJECT IDENTIFIER

```
{ mgmt 1 }
```

or

```
1.3.6.1.2.1
```

in defining the Internet-standard MIB.

The generation of new versions of the Internet-standard MIB is a rigorous process. Section 5 of this memo describes the rules used when a new version is defined.

3.1.3. Experimental

The experimental(3) subtree is used to identify objects used in Internet experiments. Administration of the experimental(3) subtree is delegated by the IAB to the Internet Assigned Numbers Authority of the Internet.

For example, an experimenter might received number 17, and would have available the OBJECT IDENTIFIER

```
{ experimental 17 }
```

or

```
1.3.6.1.3.17
```

for use.

As a part of the assignment process, the Internet Assigned Numbers Authority may make requirements as to how that subtree is used.

3.1.4. Private

The private(4) subtree is used to identify objects defined unilaterally. Administration of the private(4) subtree is delegated by the IAB to the Internet Assigned Numbers Authority for the Internet. Initially, this subtree has at least one child:

```
enterprises   OBJECT IDENTIFIER ::= { private 1 }
```

The enterprises(1) subtree is used, among other things, to permit parties providing networking subsystems to register models of their products.

Upon receiving a subtree, the enterprise may, for example, define new MIB objects in this subtree. In addition, it is strongly recommended that the enterprise will also register its networking subsystems under this subtree, in order to provide an unambiguous identification mechanism for use in management protocols. For example, if the "Flintstones, Inc." enterprise produced networking subsystems, then they could request a node under the enterprises subtree from the Internet Assigned Numbers Authority. Such a node might be numbered:

```
1.3.6.1.4.1.42
```

The "Flintstones, Inc." enterprise might then register their "Fred Router" under the name of:

```
1.3.6.1.4.1.42.1.1
```

3.2. Syntax

Syntax is used to define the structure corresponding to object types. ASN.1 constructs are used to define this structure, although the full generality of ASN.1 is not permitted.

The ASN.1 type ObjectSyntax defines the different syntaxes which may be used in defining an object type.

3.2.1. Primitive Types

Only the ASN.1 primitive types INTEGER, OCTET STRING, OBJECT IDENTIFIER, and NULL are permitted. These are sometimes referred to as non-aggregate types.

3.2.1.1. Guidelines for Enumerated INTEGERs

If an enumerated INTEGER is listed as an object type, then a named-number having the value 0 shall not be present in the list of

enumerations. Use of this value is prohibited.

3.2.2. Constructor Types

The ASN.1 constructor type SEQUENCE is permitted, providing that it is used to generate either lists or tables.

For lists, the syntax takes the form:

```
SEQUENCE { <type1>, ..., <typeN> }
```

where each <type> resolves to one of the ASN.1 primitive types listed above. Further, these ASN.1 types are always present (the DEFAULT and OPTIONAL clauses do not appear in the SEQUENCE definition).

For tables, the syntax takes the form:

```
SEQUENCE OF <entry>
```

where <entry> resolves to a list constructor.

Lists and tables are sometimes referred to as aggregate types.

3.2.3. Defined Types

In addition, new application-wide types may be defined, so long as they resolve into an IMPLICITly defined ASN.1 primitive type, list, table, or some other application-wide type. Initially, few application-wide types are defined. Future memos will no doubt define others once a consensus is reached.

3.2.3.1. NetworkAddress

This CHOICE represents an address from one of possibly several protocol families. Currently, only one protocol family, the Internet family, is present in this CHOICE.

3.2.3.2. IpAddress

This application-wide type represents a 32-bit internet address. It is represented as an OCTET STRING of length 4, in network byte-order.

When this ASN.1 type is encoded using the ASN.1 basic encoding rules, only the primitive encoding form shall be used.

3.2.3.3. Counter

This application-wide type represents a non-negative integer which

monotonically increases until it reaches a maximum value, when it wraps around and starts increasing again from zero. This memo specifies a maximum value of 2^32-1 (4294967295 decimal) for counters.

3.2.3.4. Gauge

This application-wide type represents a non-negative integer, which may increase or decrease, but which latches at a maximum value. This memo specifies a maximum value of 2^32-1 (4294967295 decimal) for gauges.

3.2.3.5. TimeTicks

This application-wide type represents a non-negative integer which counts the time in hundredths of a second since some epoch. When object types are defined in the MIB which use this ASN.1 type, the description of the object type identifies the reference epoch.

3.2.3.6. Opaque

This application-wide type supports the capability to pass arbitrary ASN.1 syntax. A value is encoded using the ASN.1 basic rules into a string of octets. This, in turn, is encoded as an OCTET STRING, in effect "double-wrapping" the original ASN.1 value.

Note that a conforming implementation need only be able to accept and recognize opaquely-encoded data. It need not be able to unwrap the data and then interpret its contents.

Further note that by use of the ASN.1 EXTERNAL type, encodings other than ASN.1 may be used in opaquely-encoded data.

3.3. Encodings

Once an instance of an object type has been identified, its value may be transmitted by applying the basic encoding rules of ASN.1 to the syntax for the object type.

4. Managed Objects

Although it is not the purpose of this memo to define objects in the MIB, this memo specifies a format to be used by other memos which define these objects.

An object type definition consists of five fields:

OBJECT:

A textual name, termed the OBJECT DESCRIPTOR, for the object type, along with its corresponding OBJECT IDENTIFIER.

Syntax:
The abstract syntax for the object type. This must resolve to an instance of the ASN.1 type ObjectSyntax (defined below).

Definition:
A textual description of the semantics of the object type. Implementations should ensure that their instance of the object fulfills this definition since this MIB is intended for use in multi-vendor environments. As such it is vital that objects have consistent meaning across all machines.

Access:
One of read-only, read-write, write-only, or not-accessible.

Status:
One of mandatory, optional, or obsolete.

Future memos may also specify other fields for the objects which they define.

4.1. Guidelines for Object Names

No object type in the Internet-Standard MIB shall use a sub-identifier of 0 in its name. This value is reserved for use with future extensions.

Each OBJECT DESCRIPTOR corresponding to an object type in the internet-standard MIB shall be a unique, but mnemonic, printable string. This promotes a common language for humans to use when discussing the MIB and also facilitates simple table mappings for user interfaces.

4.2. Object Types and Instances

An object type is a definition of a kind of managed object; it is

declarative in nature. In contrast, an object instance is an instantiation of an object type which has been bound to a value. For example, the notion of an entry in a routing table might be defined in the MIB. Such a notion corresponds to an object type; individual entries in a particular routing table which exist at some time are object instances of that object type.

A collection of object types is defined in the MIB. Each such subject type is uniquely named by its OBJECT IDENTIFIER and also has a textual name, which is its OBJECT DESCRIPTOR. The means whereby object instances are referenced is not defined in the MIB. Reference to object instances is achieved by a protocol-specific mechanism: it is the responsibility of each management protocol adhering to the SMI to define this mechanism.

An object type may be defined in the MIB such that an instance of that object type represents an aggregation of information also represented by instances of some number of "subordinate" object types. For example, suppose the following object types are defined in the MIB:

```
OBJECT:
-------
   atIndex { atEntry 1 }

Syntax:
   INTEGER

Definition:
   The interface number for the physical address.

Access:
   read-write.

Status:
   mandatory.


OBJECT:
-------
   atPhysAddress { atEntry 2 }

Syntax:
   OCTET STRING

Definition:
   The media-dependent physical address.
```

```
   Access:
      read-write.

   Status:
      mandatory.


   OBJECT:
   -------
      atNetAddress { atEntry 3 }

   Syntax:
      NetworkAddress

   Definition:
      The network address corresponding to the media-dependent physical
      address.

   Access:
      read-write.

   Status:
      mandatory.
```

Then, a fourth object type might also be defined in the MIB:

```
   OBJECT:
   -------
      atEntry { atTable 1 }

   Syntax:

      AtEntry ::= SEQUENCE {
            atIndex
            INTEGER,
            atPhysAddress
            OCTET STRING,
            atNetAddress
            NetworkAddress
            }

   Definition:
      An entry in the address translation table.

   Access:
      read-write.
```

Status:
mandatory.

Each instance of this object type comprises information represented by instances of the former three object types. An object type defined in this way is called a list.

Similarly, tables can be formed by aggregations of a list type. For example, a fifth object type might also be defined in the MIB:

OBJECT:
------
atTable { at 1 }

Syntax:
SEQUENCE OF AtEntry

Definition:
The address translation table.

Access:
read-write.

Status:
mandatory.

such that each instance of the atTable object comprises information represented by the set of atEntry object types that collectively constitute a given atTable object instance, that is, a given address translation table.

Consider how one might refer to a simple object within a table. Continuing with the previous example, one might name the object type

{ atPhysAddress }

and specify, using a protocol-specific mechanism, the object instance

{ atNetAddress } = { internet "10.0.0.52" }

This pairing of object type and object instance would refer to all instances of atPhysAddress which are part of any entry in some address translation table for which the associated atNetAddress value is { internet "10.0.0.52" }.

To continue with this example, consider how one might refer to an aggregate object (list) within a table. Naming the object type

```
{ atEntry }
```

and specifying, using a protocol-specific mechanism, the object instance

```
{ atNetAddress } = { internet "10.0.0.52" }
```

refers to all instances of entries in the table for which the associated atNetAddress value is { internet "10.0.0.52" }.

Each management protocol must provide a mechanism for accessing simple (non-aggregate) object types. Each management protocol specifies whether or not it supports access to aggregate object types. Further, the protocol must specify which instances are "returned" when an object type/instance pairing refers to more than one instance of a type.

To afford support for a variety of management protocols, all information by which instances of a given object type may be usefully distinguished, one from another, is represented by instances of object types defined in the MIB.

4.3. Macros for Managed Objects

In order to facilitate the use of tools for processing the definition of the MIB, the OBJECT-TYPE macro may be used. This macro permits the key aspects of an object type to be represented in a formal way.

```
OBJECT-TYPE MACRO ::=
BEGIN
    TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                      "ACCESS" Access
                      "STATUS" Status
    VALUE NOTATION ::= value (VALUE ObjectName)

    Access ::= "read-only"
                    | "read-write"
                    | "write-only"
                    | "not-accessible"
    Status ::= "mandatory"
                    | "optional"
                    | "obsolete"
    END
```

Given the object types defined earlier, we might imagine the following definitions being present in the MIB:

```
                atIndex OBJECT-TYPE
```

```
                SYNTAX  INTEGER
                ACCESS  read-write
                STATUS  mandatory
                ::= { atEntry 1 }

        atPhysAddress OBJECT-TYPE
                SYNTAX  OCTET STRING
                ACCESS  read-write
                STATUS  mandatory
                ::= { atEntry 2 }

        atNetAddress OBJECT-TYPE
                SYNTAX  NetworkAddress
                ACCESS  read-write
                STATUS  mandatory
                ::= { atEntry 3 }

        atEntry OBJECT-TYPE
                SYNTAX  AtEntry
                ACCESS  read-write
                STATUS  mandatory
                ::= { atTable 1 }

        atTable OBJECT-TYPE
                SYNTAX  SEQUENCE OF AtEntry
                ACCESS  read-write
                STATUS  mandatory
                ::= { at 1 }

        AtEntry ::= SEQUENCE {
            atIndex
                INTEGER,
            atPhysAddress
                OCTET STRING,
            atNetAddress
                NetworkAddress
        }
```

The first five definitions describe object types, relating, for example, the OBJECT DESCRIPTOR atIndex to the OBJECT IDENTIFIER { atEntry 1 }. In addition, the syntax of this object is defined (INTEGER) along with the access permitted (read-write) and status (mandatory). The sixth definition describes an ASN.1 type called AtEntry.

5. Extensions to the MIB

Every Internet-standard MIB document obsoletes all previous such documents. The portion of a name, termed the tail, following the OBJECT IDENTIFIER

```
{ mgmt version-number }
```

used to name objects shall remain unchanged between versions. New versions may:

(1) declare old object types obsolete (if necessary), but not delete their names;

(2) augment the definition of an object type corresponding to a list by appending non-aggregate object types to the object types in the list; or,

(3) define entirely new object types.

New versions may not:

(1) change the semantics of any previously defined object without changing the name of that object.

These rules are important because they admit easier support for multiple versions of the Internet-standard MIB. In particular, the semantics associated with the tail of a name remain constant throughout different versions of the MIB. Because multiple versions of the MIB may thus coincide in "tail-space," implementations supporting multiple versions of the MIB can be vastly simplified.

However, as a consequence, a management agent might return an instance corresponding to a superset of the expected object type. Following the principle of robustness, in this exceptional case, a manager should ignore any additional information beyond the definition of the expected object type. However, the robustness principle requires that one exercise care with respect to control actions: if an instance does not have the same syntax as its expected object type, then those control actions must fail. In both the monitoring and control cases, the name of an object returned by an operation must be identical to the name requested by an operation.

6. Definitions

```
            RFC1155-SMI DEFINITIONS ::= BEGIN

            EXPORTS -- EVERYTHING
                    internet, directory, mgmt,
                    experimental, private, enterprises,
                    OBJECT-TYPE, ObjectName, ObjectSyntax, SimpleSyntax,
                    ApplicationSyntax, NetworkAddress, IpAddress,
                    Counter, Gauge, TimeTicks, Opaque;

             -- the path to the root

             internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }

             directory     OBJECT IDENTIFIER ::= { internet 1 }

             mgmt          OBJECT IDENTIFIER ::= { internet 2 }

             experimental  OBJECT IDENTIFIER ::= { internet 3 }

             private       OBJECT IDENTIFIER ::= { internet 4 }
             enterprises   OBJECT IDENTIFIER ::= { private 1 }


             -- definition of object types

             OBJECT-TYPE MACRO ::=
             BEGIN
                 TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                                   "ACCESS" Access
                                   "STATUS" Status
                 VALUE NOTATION ::= value (VALUE ObjectName)

                 Access ::= "read-only"
                                 | "read-write"
                                 | "write-only"
                                 | "not-accessible"
                 Status ::= "mandatory"
                                 | "optional"
                                 | "obsolete"
             END

                -- names of objects in the MIB

                ObjectName ::=
                    OBJECT IDENTIFIER
```

```
-- syntax of objects in the MIB

ObjectSyntax ::=
    CHOICE {
        simple
            SimpleSyntax,

-- note that simple SEQUENCEs are not directly
-- mentioned here to keep things simple (i.e.,
-- prevent mis-use).  However, application-wide
-- types which are IMPLICITly encoded simple
-- SEQUENCEs may appear in the following CHOICE

        application-wide
            ApplicationSyntax
    }

   SimpleSyntax ::=
       CHOICE {
           number
               INTEGER,

           string
               OCTET STRING,

           object
               OBJECT IDENTIFIER,

           empty
               NULL
       }

   ApplicationSyntax ::=
       CHOICE {
           address
               NetworkAddress,

           counter
               Counter,

           gauge
               Gauge,

           ticks
               TimeTicks,

           arbitrary
               Opaque
```

```
-- other application-wide types, as they are
-- defined, will be added here
    }


-- application-wide types

NetworkAddress ::=
    CHOICE {
        internet
            IpAddress
    }

IpAddress ::=
    [APPLICATION 0]          -- in network-byte order
        IMPLICIT OCTET STRING (SIZE (4))

Counter ::=
    [APPLICATION 1]
        IMPLICIT INTEGER (0..4294967295)

Gauge ::=
    [APPLICATION 2]
        IMPLICIT INTEGER (0..4294967295)

TimeTicks ::=
    [APPLICATION 3]
        IMPLICIT INTEGER (0..4294967295)

Opaque ::=
    [APPLICATION 4]          -- arbitrary ASN.1 value,
        IMPLICIT OCTET STRING   --   "double-wrapped"

END
```

7. Acknowledgements

This memo was influenced by three sets of contributors to earlier drafts:

First, Lee Labarre of the MITRE Corporation, who as author of the NETMAN SMI [4], presented the basic roadmap for the SMI.

Second, several individuals who provided valuable comments on this memo prior to its initial distribution:

James R. Davin, Proteon
Mark S. Fedor, NYSERNet
Craig Partridge, BBN Laboratories
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Wengyik Yeong, NYSERNet

Third, the IETF MIB working group:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
James R. Davin, Proteon
Mark S. Fedor, NYSERNet
Robb Foster, BBN
Phill Gross, The MITRE Corporation
Bent Torp Jensen, Convergent Technology
Lee Labarre, The MITRE Corporation
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Dave Mackie, 3Com/Bridge
Craig Partridge, BBN (chair)
Jim Robertson, 3Com/Bridge
Marshall T. Rose, The Wollongong Group
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Lou Steinberg, IBM
Dean Throop, Data General
Unni Warrier, Unisys

8. References

[1] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[2] McCloghrie K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Performance Systems International and Hughes LAN Systems, May 1990.

[3] Case, J., M. Fedor, M. Schoffstall, and J. Davin, The Simple Network Management Protocol", RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science, May 1990.

[4] LaBarre, L., "Structure and Identification of Management Information for the Internet", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, April 1988.

[5] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[6] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[7] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International Organization for Standardization, International Standard 8825, December 1987.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Marshall T. Rose
PSI, Inc.
PSI California Office
P.O. Box 391776
Mountain View, CA 94039

Phone: (415) 961-3380

EMail: mrose@PSI.COM

Keith McCloghrie
The Wollongong Group
1129 San Antonio Road
Palo Alto, CA 04303

Phone: (415) 962-7160

EMail: sytek!kzm@HPLABS.HP.COM