Conformance Statements
for version 2 of the
Simple Network Management Protocol (SNMPv2)

Status of this Memo

This RFC specifies an IAB standards track protocol for the
Internet community, and requests discussion and suggestions
for improvements. Please refer to the current edition of the
"IAB Official Protocol Standards" for the standardization
state and status of this protocol. Distribution of this memo
is unlimited.

Table of Contents

1 Introduction ........................................... 2
1.1 A Note on Terminology .................................. 2
2 Definitions ........................................... 3
3.1 The OBJECT-GROUP macro ............................... 3
3.2 The MODULE-COMPLIANCE macro ......................... 4
3.3 The AGENT-CAPABILITIES macro ........................ 7
3 Mapping of the OBJECT-GROUP macro .................... 10
3.1 Mapping of the OBJECTS clause ....................... 10
3.2 Mapping of the STATUS clause ....................... 10
3.3 Mapping of the DESCRIPTION clause .................. 10
3.4 Mapping of the REFERENCE clause .................. 11
3.5 Mapping of the OBJECT-GROUP value ................ 11
3.6 Usage Example ...................................... 12
4 Mapping of the MODULE-COMPLIANCE macro ............. 13
4.1 Mapping of the STATUS clause ....................... 13
4.2 Mapping of the DESCRIPTION clause ................. 13
4.3 Mapping of the REFERENCE clause ................... 13
4.4 Mapping of the MODULE clause ...................... 13
4.4.1 Mapping of the MANDATORY-GROUPS clause ....... 14
4.4.2 Mapping of the GROUP clause .................... 14
4.4.3 Mapping of the OBJECT clause ................... 14
1. Introduction

A network management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines both authentication and authorization policies.

Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled through access to their management information.

Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written using a subset of OSI’s Abstract Syntax Notation One (ASN.1) [1], termed the Structure of Management Information (SMI) [2].

It may be useful to define the acceptable lower-bounds of implementation, along with the actual level of implementation achieved. It is the purpose of this document to define the notation used for these purposes.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, is termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).
2. Definitions

SNMPv2-CONF DEFINITIONS ::= BEGIN

-- definitions for conformance groups

OBJECT-GROUP MACRO ::= BEGIN

    TYPE NOTATION ::= ObjectsPart
                    "STATUS" Status
                    "DESCRIPTION" Text
                    ReferPart

    VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)

ObjectsPart ::= "OBJECTS" "{" Objects "}"

Objects ::= Object
          | Objects "," Object

Object ::= value(Name ObjectName)

Status ::= "current"
          | "obsolete"

ReferPart ::= "REFERENCE" Text
            | empty

-- uses the NVT ASCII character set

Text ::= """" string """"
-- definitions for compliance statements

MODULE-COMPLIANCE MACRO ::= BEGIN
  TYPE NOTATION ::= "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  ModulePart
  VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)
  Status ::= "current" | "obsolete"
  ReferPart ::= "REFERENCE" Text | empty
  ModulePart ::= Modules | empty
  Modules ::= Module | Modules Module
  Module ::= -- name of module -- "MODULE" ModuleName
            MandatoryPart
            CompliancePart
  ModuleName ::= modulereference ModuleIdentifier
                -- must not be empty unless contained
                -- in MIB Module
                | empty
  ModuleIdentifier ::= value(ModuleID OBJECT IDENTIFIER) | empty
  MandatoryPart ::= "MANDATORY-GROUPS" "{" Groups "}" | empty
Groups ::=    
  Group 
  | Groups ";" Group  
Group ::=    
  value(Group OBJECT IDENTIFIER)  
CompliancePart ::=    
  Compliances 
  | empty  
Compliances ::=    
  Compliance 
  | Compliances Compliance  
Compliance ::=    
  ComplianceGroup 
  | Object  
ComplianceGroup ::=    
  "GROUP" value(Name OBJECT IDENTIFIER) 
  "DESCRIPTION" Text  
Object ::=    
  "OBJECT" value(Name ObjectName) 
  SyntaxPart 
  WriteSyntaxPart 
  AccessPart 
  "DESCRIPTION" Text  
-- must be a refinement for object’s SYNTAX clause  
SyntaxPart ::=    
  "SYNTAX" type(SYNTAX) 
  | empty  
-- must be a refinement for object’s SYNTAX clause  
WriteSyntaxPart ::=    
  "WRITE-SYNTAX" type(WriteSYNTAX) 
  | empty  
AccessPart ::=    
  "MIN-ACCESS" Access 
  | empty  
Access ::=    
  "not-accessible" 
  | "read-only" 
  | "read-write"
"read-create"

-- uses the NVT ASCII character set

Text ::= **** string ****

END
-- definitions for capabilities statements

AGENT-CAPABILITIES MACRO ::= BEGIN

TYPE NOTATION ::= "PRODUCT-RELEASE" Text
                 "STATUS" Status
                 "DESCRIPTION" Text
                 ReferPart
                 ModulePart


Status ::= "current"
         "obsolete"

ReferPart ::= "REFERENCE" Text
            empty

ModulePart ::= Modules
             empty

Modules ::= Module
          Modules Module

Module ::= -- name of module -- "SUPPORTS" ModuleName
          "INCLUDES" "{" Groups "}" VariationPart

ModuleName ::= identifier ModuleIdentifier

ModuleIdentifier ::= value(ModuleID OBJECT IDENTIFIER)
                    empty

Groups ::= Group
         Groups "," Group

Group ::=
value(Name OBJECT IDENTIFIER)

VariationPart ::=  
Variations  
| empty  
Variations ::=  
Variation  
| Variations Variation  

Variation ::=  
"VARIATION" value(Name ObjectName)  
SyntaxPart  
WriteSyntaxPart  
AccessPart  
CreationPart  
DefValPart  
"DESCRIPTION" Text  

-- must be a refinement for object’s SYNTAX clause  
SyntaxPart ::=  
"SYNTAX" type(SYNTAX)  
| empty  

-- must be a refinement for object’s SYNTAX clause  
WriteSyntaxPart ::=  
"WRITE-SYNTAX" type(WriteSYNTAX)  
| empty  

AccessPart ::=  
"ACCESS" Access  
| empty  

Access ::=  
"not-implemented"  
| "read-only"  
| "read-write"  
| "read-create"  
-- following is for backward-compatibility only  
| "write-only"  

CreationPart ::=  
"CREATION-REQUIRES" "{ Cells "}"  
| empty  

Cells ::=  


Cell
    | Cells "," Cell

Cell ::= value(Cell ObjectName)

DefValPart ::= "DEFVAL" "{" value(Defval ObjectSyntax) "}"
             | empty

-- uses the NVT ASCII character set
Text ::= "****" string "****"
3. Mapping of the OBJECT-GROUP macro

For conformance purposes, it is useful to define a collection of related managed objects. The OBJECT-GROUP macro is used to define each such collection of related objects. It should be noted that the expansion of the OBJECT-GROUP macro is something which conceptually happens during implementation and not during run-time.

To "implement" an object, a SNMPv2 entity acting in an agent role must return a reasonably accurate value for management protocol retrieval operations; similarly, if the object is writable, then in response to a management protocol set operation, a SNMPv2 entity must accordingly be able to reasonably influence the underlying managed entity. If a SNMPv2 entity acting in an agent role can not implement an object, the management protocol provides for the SNMPv2 entity to return an exception or error, e.g, noSuchObject [6]. Under no circumstances shall a SNMPv2 entity return a value for objects which it does not implement -- it must always return the appropriate exception or error, as described in the protocol specification [6].

3.1. Mapping of the OBJECTS clause

The OBJECTS clause which must be present, is used to name each object contained in the conformance group. Each of the named objects must be defined in the same information module as the OBJECT-GROUP macro appears, and must have a MAX-ACCESS clause value of "read-only", "read-write", or "read-create".

3.2. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory.

3.3. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual definition of that group, along with a description of
any relations to other groups. Note that generic compliance requirements should not be stated in this clause. However, implementation relationships between this group and other groups may be defined in this clause.

3.4. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a group defined in some other information module. This is useful when de-ossifying a MIB module produced by some other organization.

3.5. Mapping of the OBJECT-GROUP value

The value of an invocation of the OBJECT-GROUP macro is the name of the group, which is an OBJECT IDENTIFIER, an administratively assigned name.
3.6. Usage Example

Consider how the system group from MIB-II [3] might be described:

```
systemGroup OBJECT-GROUP
    OBJECTS { sysDescr, sysObjectID, sysUpTime,
               sysContact, sysName, sysLocation,
               sysServices }
    STATUS current
    DESCRIPTION
        "The system group defines objects which are common
to all managed systems."
 ::= { mibIIGroups 1 }
```

According to this invocation, the conformance group named

```
{ mibIIGroups 1 }
```

contains 7 objects.
4. Mapping of the MODULE-COMPLIANCE macro

The MODULE-COMPLIANCE macro is used to convey a minimum set of requirements with respect to implementation of one or more MIB modules. It should be noted that the expansion of the MODULE-COMPLIANCE macro is something which conceptually happens during implementation and not during run-time.

A requirement on all "standard" MIB modules is that a corresponding MODULE-COMPLIANCE specification is also defined, either in the same information module or in a companion information module.

4.1. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that that object is obsolete, but that an implementor may wish to support that object to foster interoperability with older implementations.

4.2. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual definition of this compliance statement and should embody any information which would otherwise be communicated in any ASN.1 commentary annotations associated with the statement.

4.3. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a compliance statement defined in some other information module.

4.4. Mapping of the MODULE clause

The MODULE clause, which must be present, is repeatedly used to name each MIB module for which compliance requirements are
being specified. Each MIB module is named by its module name, and optionally, by its associated OBJECT IDENTIFIER as well. The module name can be omitted when the MODULE-COMPLIANCE invocation occurs inside a MIB module, to refer to the encompassing MIB module.

4.4.1. Mapping of the MANDATORY-GROUPS clause

The MANDATORY-GROUPS clause, which need not be present, names the one or more groups within the correspondent MIB module which are unconditionally mandatory for implementation. If a SNMPv2 entity acting in an agent role claims compliance to the MIB module, then it must implement each and every object within each conformance group listed. That is, if a SNMPv2 entity returns a noSuchObject exception in response to a management protocol get operation [5] for any object within any mandatory conformance group for every MIB view, then that SNMPv2 entity is not a conformant implementation of the MIB module.

4.4.2. Mapping of the GROUP clause

The GROUP clause which need not be present, is repeatedly used to name each MIB group which is conditionally mandatory or unconditionally optional for compliance to the MIB module. A MIB group named in a GROUP clause must be absent from the correspondent MANDATORY-GROUPS clause.

Conditionally mandatory groups include those which are mandatory only if a particular protocol is implemented, or only if another group is implemented. A GROUP clause’s DESCRIPTION specifies the conditions under which the group is conditionally mandatory.

A MIB group which is named in neither a MANDATORY-GROUPS clause nor a GROUP clause, is unconditionally optional for compliance to the MIB module.

4.4.3. Mapping of the OBJECT clause

The OBJECT clause which need not be present, is repeatedly used to name each MIB object for which compliance has a
refined requirement with respect to the MIB module definition. The MIB object must be present in one of the conformance groups named in the correspondent MANDATORY-GROUPS clause or GROUP clauses.

4.4.3.1. Mapping of the SYNTAX clause

The SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent OBJECT clause. Note that if this clause and a WRITE-SYNTAX clause are both present, then this clause only applies when instances of the object named in the correspondent OBJECT clause are read.

Consult Section 10 of [2] for more information on refined syntax.

4.4.3.2. Mapping of the WRITE-SYNTAX clause

The WRITE-SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent OBJECT clause when instances of that object are written.

Consult Section 10 of [2] for more information on refined syntax.

4.4.3.3. Mapping of the MIN-ACCESS clause

The MIN-ACCESS clause, which need not be present, is used to define the minimal level of access for the object named in the correspondent OBJECT clause. If this clause is absent, the minimal level of access is the same as the maximal level specified in the correspondent invocation of the OBJECT-TYPE macro. If present, this clause must not specify a greater level of access than is specified in the correspondent invocation of the OBJECT-TYPE macro.

The level of access for certain types of objects is fixed according to their syntax definition. These types are: conceptual tables and rows, auxiliary objects, and objects with the syntax of Counter32, Counter64, or certain types of Case, McCloghrie, Rose & Waldbusser
textual conventions (e.g., RowStatus [6]). A MIN-ACCESS clause should not be present for such objects.

An implementation is compliant if the level of access it provides is greater or equal to the minimal level in the MODULE-COMPLIANCE macro and less or equal to the maximal level in the OBJECT-TYPE macro.

4.4.3.4. Mapping of the DESCRIPTION clause

The DESCRIPTION clause must be present for each use of the GROUP or OBJECT clause. For an OBJECT clause, it contains a textual description of the refined compliance requirement. For a GROUP clause, it contains a textual description of the conditions under which the group is conditionally mandatory or unconditionally optional.

4.5. Mapping of the MODULE-COMPLIANCE value

The value of an invocation of the MODULE-COMPLIANCE macro is an OBJECT IDENTIFIER. As such, this value may be authoritatively used when referring to the compliance statement embodied by that invocation of the macro.
4.6. Usage Example

Consider how a compliance statement might be included at the end of the MIB-II document [3], assuming that conformance groups were defined therein:

mibIICompliances
OBJECT IDENTIFIER ::= { mibIIConformance 1 }
mibIIGroups    OBJECT IDENTIFIER ::= { mibIIConformance 2 }

mibIICompliance MODULE-COMPLIANCE
  STATUS  current
  DESCRIPTION
    "The compliance statement for SNMPv2 entities residing on systems which implement the Internet suite of protocols."
  MODULE  -- compliance to the containing MIB module
    MANDATORY-GROUPS   { systemGroup, snmpGroup }

GROUP       interfacesGroup
  DESCRIPTION
    "The interfaces group is mandatory for systems with network interfaces."

GROUP       ipGroup
  DESCRIPTION
    "The ip group is mandatory for systems which implement IP."

GROUP       icmpGroup
  DESCRIPTION
    "The icmp group is mandatory for systems which implement ICMP."

GROUP       tcpGroup
  DESCRIPTION
    "The tcp group is mandatory for systems which implement TCP."
  OBJECT      tcpConnState
    MIN-ACCESS  read-only
    DESCRIPTION
      "A compliant system need not allow write-access to this object."

GROUP       udpGroup
DESCRIPTION
"The udp group is mandatory for systems which implement UDP."

GROUP       egpGroup
DESCRIPTION
"The egp group is mandatory for systems which implement EGP."

 ::= { mibIICompliances 1 }

According to this invocation, to claim alignment with the compliance statement named

 { mibIICompliances 1 }

a system must implement RFC1213’s systemGroup and snmpGroup conformance groups. If the system implements any network interfaces, then RFC1213’s interfacesGroup conformance group must be implemented. Further, if the system implements any of the IP, ICMP, TCP, UDP, or EGP protocols, then the correspondent conformance group in RFC1213 must be implemented, if compliance is to be claimed. Finally, although RFC1213 specifies that it makes "protocol sense" for the tcpConnState object to be writable, this specification allows the system to permit only read-only access and still claim compliance.
5. Mapping of the AGENT-CAPABILITIES macro

The AGENT-CAPABILITIES macro is used to convey the capabilities present in a SNMPv2 entity acting in an agent role. It should be noted that the expansion of the AGENT-CAPABILITIES macro is something which conceptually happens during implementation and not during run-time.

When a MIB module is written, it is divided into units of conformance termed groups. If a SNMPv2 entity acting in an agent role claims to implement a group, then it must implement each and every object within that group. Of course, for whatever reason, a SNMPv2 entity might implement only a subset of the groups within a MIB module. In addition, the definition of some MIB objects leave some aspects of the definition to the discretion of an implementor.

Practical experience has demonstrated a need for concisely describing the capabilities of an agent with respect to one or more MIB modules. The AGENT-CAPABILITIES macro allows an agent implementor to describe the precise level of support which an agent claims in regards to a MIB group, and to bind that description to the value of sysObjectID [3] associated with the agent, or to the value of an instance of the snmpORID object in the snmpORTable [4]. In particular, some objects may have restricted or augmented syntax or access-levels.

If the AGENT-CAPABILITIES invocation is given to a management-station implementor, then that implementor can build management applications which optimize themselves when communicating with a particular agent. For example, the management-station can maintain a database of these invocations. When a management-station interacts with an agent, it retrieves the agent’s sysObjectID [3]. Based on this, it consults the database. If an entry is found, then the management application can optimize its behavior accordingly.

Note that this binding to sysObjectID may not always suffice to define all MIB objects to which an agent can provide access. In particular, this situation occurs where the agent dynamically learns of the objects it supports. In these cases, the snmpORID column of snmpORTable [4] contains information which should be used in addition to sysObjectID.
Note that the AGENT-CAPABILITIES macro specifies refinements or variations with respect to OBJECT-TYPE macros in MIB modules, NOT with respect to MODULE-COMPLIANCE macros in compliance statements.

5.1. Mapping of the PRODUCT-RELEASE clause

The PRODUCT-RELEASE clause, which must be present, contains a textual description of the product release which includes this agent.

5.2. Mapping of the STATUS clause

The STATUS clause, which must be present, indicates whether this definition is current or historic.

The values "current", and "obsolete" are self-explanatory. The "deprecated" value indicates that that object is obsolete, but that an implementor may wish to support that object to foster interoperability with older implementations.

5.3. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present, contains a textual description of this agent.

5.4. Mapping of the REFERENCE clause

The REFERENCE clause, which need not be present, contains a textual cross-reference to a capability statement defined in some other information module.

5.5. Mapping of the SUPPORTS clause

The SUPPORTS clause, which need not be present, is repeatedly used to name each MIB module for which the agent claims a complete or partial implementation. Each MIB module is named by its module name, and optionally, by its associated OBJECT IDENTIFIER as well.
5.5.1. Mapping of the INCLUDES clause

The INCLUDES clause, which must be present for each use of the SUPPORTS clause, is used to name each MIB group associated with the SUPPORT clause, which the agent claims to implement.

5.5.2. Mapping of the VARIATION clause

The VARIATION clause, which need not be present, is repeatedly used to name each MIB object which the agent implements in some variant or refined fashion with respect to the correspondent invocation of the OBJECT-TYPE macro.

Note that the variation concept is meant for generic implementation restrictions, e.g., if the variation for an object depends on the values of other objects, then this should be noted in the appropriate DESCRIPTION clause.

5.5.2.1. Mapping of the SYNTAX clause

The SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent VARIATION clause. Note that if this clause and a WRITE-SYNTAX clause are both present, then this clause only applies when instances of the object named in the correspondent VARIATION clause are read.

Consult Section 10 of [2] for more information on refined syntax.

5.5.2.2. Mapping of the WRITE-SYNTAX clause

The WRITE-SYNTAX clause, which need not be present, is used to provide a refined SYNTAX for the object named in the correspondent VARIATION clause when instances of that object are written.

Consult Section 10 of [2] for more information on refined syntax.
5.5.2.3. Mapping of the ACCESS clause

The ACCESS clause, which need not be present, is used to indicate the agent provides less than the maximal level of access to the object named in the correspondent VARIATION clause.

The value "not-implemented" indicates the agent does not implement the object, and in the ordering of possible values is equivalent to "not-accessible".

The value "write-only" is provided solely for backward compatibility, and shall not be used for newly-defined object types. In the ordering of possible values, "write-only" is less than "not-accessible".

5.5.2.4. Mapping of the CREATION-REQUIRES clause

The CREATION-REQUIRES clause, which need not be present, is used to name the columnar objects of a conceptual row to which values must be explicitly assigned, by a management protocol set operation, before the agent will allow the instance of the status column of that row to be set to 'active'. (Consult the definition of RowStatus [6].)

If the conceptual row does not have a status column (i.e., the objects corresponding to the conceptual table were defined using the mechanisms in [7,8]), then the CREATION-REQUIRES clause, which need not be present, is used to name the columnar objects of a conceptual row to which values must be explicitly assigned, by a management protocol set operation, before the agent will create new instances of objects in that row.

This clause must not present unless the object named in the correspondent VARIATION clause is a conceptual row, i.e., has a syntax which resolves to a SEQUENCE containing columnar objects. The objects named in the value of this clause usually will refer to columnar objects in that row. However, objects unrelated to the conceptual row may also be specified.

All objects which are named in the CREATION-REQUIRES clause for a conceptual row, and which are columnar objects of that row, must have an access level of "read-create".
5.5.2.5. Mapping of the DEFVAL clause

The DEFVAL clause, which need not be present, is used to provide a refined DEFVAL value for the object named in the correspondent VARIATION clause. The semantics of this value are identical to those of the OBJECT-TYPE macro’s DEFVAL clause.

5.5.2.6. Mapping of the DESCRIPTION clause

The DESCRIPTION clause, which must be present for each use of the VARIATION clause, contains a textual description of the variant or refined implementation.

5.6. Mapping of the AGENT-CAPABILITIES value

The value of an invocation of the AGENT-CAPABILITIES macro is an OBJECT IDENTIFIER, which names the value of sysObjectID [3] or snmpORID [4] for which this capabilities statement is valid.
5.7. Usage Example

Consider how a capabilities statement for an agent might be described:

eampleAgent AGENT-CAPABILITIES
PRODUCT-RELEASE "ACME Agent release 1.1 for 4BSD"
STATUS current
DESCRIPTION "ACME agent for 4BSD"

SUPPORTS RFC1213-MIB
INCLUDES { systemGroup, interfacesGroup, atGroup, ipGroup, icmpGroup, tcpGroup, udpGroup, snmpGroup }

VARIATION ifAdminStatus
SYNTAX INTEGER { up(1), down(2) }
DESCRIPTION "Unable to set test mode on 4BSD"

VARIATION ifOperStatus
SYNTAX INTEGER { up(1), down(2) }
DESCRIPTION "Information limited on 4BSD"

VARIATION atEntry
CREATION-REQUIRES { atPhysAddress }
DESCRIPTION "Address mappings on 4BSD require both protocol and media addresses"

VARIATION ipDefaultTTL
SYNTAX INTEGER (255..255)
DESCRIPTION "Hard-wired on 4BSD"

VARIATION ipInAddrErrors
ACCESS not-implemented
DESCRIPTION "Information not available on 4BSD"

VARIATION ipRouteType
SYNTAX INTEGER { direct(3), indirect(4) }
WRITE-SYNTAX INTEGER { invalid(2), direct(3), indirect(4) }
DESCRIPTION "Information limited on 4BSD"

VARIATION tcpConnState
ACCESS read-only
DESCRIPTION "Unable to set this on 4BSD"
SUPPORTS EVAL-MIB
INCLUDES { functionsGroup, expressionsGroup }
VARIATION exprEntry
CREATION-REQUIRES { evalString }
DESCRIPTION "Conceptual row creation supported"

::= { acmeAgents 1 }

According to this invocation, an agent with a sysObjectID (or snmpORID) value of

{ acmeAgents 1 }

supports two MIB modules.

From MIB-II, all conformance groups except the egpGroup conformance group are supported. However, the object ipInAddrErrors is not implemented, whilst the objects

ifAdminStatus
ifOperStatus
ipDefaultTTL
ipRouteType

have a restricted syntax, and the object
tcpConnState

is available only for reading. Note that in the case of the object ipRouteType the set of values which may be read is different than the set of values which may be written. Finally, when creating a new instance in the atTable, the set-request must create an instance of atPhysAddress.

From the EVAL-MIB, all the objects contained in the functionsGroup and expressionsGroup conformance groups are supported, without variation. In addition, creation of new instances in the expr table is supported.
6. Extending an Information Module

As experience is gained with a published information module, it may be desirable to revise that information module.

Section 10 of [2] defines the rules for extending an information module. The remainder of this section defines how conformance groups, compliance statements, and capabilities statements may be extended.

6.1. Conformance Groups

If any non-editorial change is made to any clause of an object group then the OBJECT IDENTIFIER value associated with that object group must also be changed, along with its associated descriptor.

6.2. Compliance Definitions

If any non-editorial change is made to any clause of a compliance definition, then the OBJECT IDENTIFIER value associated with that compliance definition must also be changed, along with its associated descriptor.

6.3. Capabilities Definitions

If any non-editorial change is made to any clause of a capabilities definition, then the OBJECT IDENTIFIER value associated with that capabilities definition must also be changed, along with its associated descriptor.
7. Acknowledgements

The section on compliance statements is based, in part, on a conversation with James R. Davin in December, 1990.

The section on capabilities statements is based, in part, on RFC 1303.

Finally, the comments of the SNMP version 2 working group are gratefully acknowledged:

Beth Adams, Network Management Forum
Steve Alexander, INTERACTIVE Systems Corporation
David Arneson, Cabletron Systems
Toshiya Asaba
Fred Baker, ACC
Jim Barnes, Xylogics, Inc.
Brian Bataille
Andy Bierman, SynOptics Communications, Inc.
Uri Blumenthal, IBM Corporation
Fred Bohle, Interlink
Jack Brown
Theodore Brunner, Bellcore
Stephen F. Bush, GE Information Services
Jeffrey D. Case, University of Tennessee, Knoxville
John Chang, IBM Corporation
Szusin Chen, Sun Microsystems
Robert Ching
Chris Chiotasso, Ungermann-Bass
Bobby A. Clay, NASA/Boeing
John Cooke, Chipcom
Tracy Cox, Bellcore
Juan Cruz, Datability, Inc.
David Cullerot, Cabletron Systems
Cathy Cunningham, Microcom
James R. (Chuck) Davin, Bellcore
Michael Davis, Clearpoint
Mike Davison, FiberCom
Cynthia DellaTorre, MITRE
Taso N. Devetzis, Bellcore
Manual Diaz, DAVID Systems, Inc.
Jon Dreyer, Sun Microsystems
David Engel, Optical Data Systems
Mike Erlinger, Lexcel
Roger Fajman, NIH
Daniel Fauvarque, Sun Microsystems
Karen Frisa, CMU
Shari Galitzer, MITRE
Shawn Gallagher, Digital Equipment Corporation
Richard Graveman, Bellcore
Maria Greene, Xyplex, Inc.
Michel Guittet, Apple
Robert Gutierrez, NASA
Bill Hagerty, Cabletron Systems
Gary W. Haney, Martin Marietta Energy Systems
Patrick Hanil, Nokia Telecommunications
Matt Hecht, SNMP Research, Inc.
Edward A. Heiner, Jr., Synertec Inc.
Susan E. Hicks, Martin Marietta Energy Systems
Geral Holzhauer, Apple
John Hopprich, DAVID Systems, Inc.
Jeff Hughes, Hewlett-Packard
Robin Iddon, Axon Networks, Inc.
David Itusak
Kevin M. Jackson, Concord Communications, Inc.
Ole J. Jacobsen, Interop Company
Ronald Jacoby, Silicon Graphics, Inc.
Satish Joshi, SynOptics Communications, Inc.
Frank Kastenholz, FTP Software
Mark Kepke, Hewlett-Packard
Ken Key, SNMP Research, Inc.
Zbiginew Kielczewski, Eicon
Jongyeol Kim
Andrew Knutsen, The Santa Cruz Operation
Michael L. Kornegay, VisiSoft
Deirdre C. Kostik, Bellcore
Cheryl Krupczak, Georgia Tech
Mark S. Lewis, Telebit
David Lin
David Lindemulder, AT&T/NCR
Ben Lisowski, Sprint
David Liu, Bell-Northern Research
John Lunny, The Wollongong Group
Robert C. Lushbaugh Martin, Marietta Energy Systems
Michael Luufer, BBN
Carl Madison, Star-Tek, Inc.
Keith McCloghrie, Hughes LAN Systems
Evan McGinnis, 3Com Corporation
Bill McKenzie, IBM Corporation
Donna McMaster, SynOptics Communications, Inc.
Warren Vik, INTERACTIVE Systems Corporation
Yannis Viniotis
Steven L. Waldbusser, Carnegie Mellon University
Timothy M. Walden, ACC
Alice Wang, Sun Microsystems
James Watt, Newbridge
Luanne Waul, Timeplex
Donald E. Westlake III, Digital Equipment Corporation
Gerry White
Bert Wijnen, IBM Corporation
Peter Wilson, 3Com Corporation
Steven Wong, Digital Equipment Corporation
Randy Worzella, IBM Corporation
Daniel Woycke, MITRE
Honda Wu
Jeff Yarnell, Protools
Chris Young, Cabletron
Kiho Yum, 3Com Corporation
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).

"Structure of Management Information for version 2 of the
Simple Network Management Protocol (SNMPv2)", RFC 1442,
SNMP Research, Inc., Hughes LAN Systems, Dover Beach

Base for Network Management of TCP/IP-based internets:

"Management Information Base for version 2 of the Simple
Network Management Protocol (SNMPv2)", RFC 1450, SNMP
Research, Inc., Hughes LAN Systems, Dover Beach

"Protocol Operations for version 2 of the Simple Network
Management Protocol (SNMPv2)", RFC 1448, SNMP Research,
Inc., Hughes LAN Systems, Dover Beach Consulting, Inc.,
Carnegie Mellon University, April 1993.

"Textual Conventions for version 2 of the Simple Network
Management Protocol (SNMPv2)", RFC 1443, SNMP Research,
Inc., Hughes LAN Systems, Dover Beach Consulting, Inc.,
Carnegie Mellon University, April 1993.

[7] Rose, M., and McCloghrie, K., "Structure and
Identification of Management Information for TCP/IP-based

9. Security Considerations

Security issues are not discussed in this memo.

10. Authors’ Addresses

Jeffrey D. Case
SNMP Research, Inc.
3001 Kimberlin Heights Rd.
Knoxville, TN  37920-9716
US

Phone: +1 615 573 1434
Email: case@snmp.com

Keith McCloghrie
Hughes LAN Systems
1225 Charleston Road
Mountain View, CA  94043
US

Phone: +1 415 966 7934
Email: kzm@hls.com

Marshall T. Rose
Dover Beach Consulting, Inc.
420 Whisman Court
Mountain View, CA  94043-2186
US

Phone: +1 415 968 1052
Email: mrose@dbc.mtview.ca.us

Steven Waldbusser
Carnegie Mellon University
4910 Forbes Ave
Pittsburgh, PA  15213
US

Phone: +1 412 268 6628
Email: waldbusser@cmu.edu