Status of this Memo

This memo defines the second version of the Management Information Base (MIB-II) for use with network management protocols in TCP/IP-based internets. 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. Abstract ............................................. 2
2. Introduction ........................................ 2
3. Changes from RFC 1156 ............................. 3
  3.1 Deprecated Objects ............................... 3
  3.2 Display Strings ................................... 4
  3.3 Physical Addresses ............................... 4
  3.4 The System Group ................................ 5
  3.5 The Interfaces Group ............................... 5
  3.6 The Address Translation Group ................. 6
  3.7 The IP Group ...................................... 6
  3.8 The ICMP Group .................................... 7
  3.9 The TCP Group .................................... 7
  3.10 The UDP Group ................................... 7
  3.11 The EGP Group ................................... 7
  3.12 The Transmission Group ......................... 8
  3.13 The SNMP Group .................................. 8
  3.14 Changes from RFC 1158 ......................... 9
4. Objects .............................................. 10
  4.1 Format of Definitions ............................ 10
5. Overview ............................................. 10
6. Definitions .......................................... 12
  6.1 Textual Conventions ............................ 12
  6.2 Groups in MIB-II ................................ 13
  6.3 The System Group ................................ 13
1. Abstract

This memo defines the second version of the Management Information Base (MIB-II) for use with network management protocols in TCP/IP-based internets. In particular, together with its companion memos which describe the structure of management information (RFC 1155) along with the network management protocol (RFC 1157) for TCP/IP-based internets, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular the Internet community.

2. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], a two-prong strategy for network management of TCP/IP-based internets was undertaken. In the short-term, the Simple Network Management Protocol (SNMP) was to be used to manage nodes in the Internet community. In the long-term, the use of the OSI network management framework was to be examined. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI) [2], and RFC 1066, which defined the Management Information Base (MIB) [3]. Both of these documents were designed so as to be compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this, portions of the Internet community became network manageable in a timely fashion.

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [4], the requirements of the SNMP and the OSI
network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network management framework, the SNMP, to respond to new operational needs in the Internet community by producing this document.

As such, the current network management framework for TCP/IP-based internets consists of: Structure and Identification of Management Information for TCP/IP-based internets, RFC 1155 [12], which describes how managed objects contained in the MIB are defined; Management Information Base for Network Management of TCP/IP-based internets: MIB-II, this memo, which describes the managed objects contained in the MIB (and supercedes RFC 1156 [13]); and, the Simple Network Management Protocol, RFC 1098 [5], which defines the protocol used to manage these objects.

3. Changes from RFC 1156

Features of this MIB include:

(1) incremental additions to reflect new operational requirements;

(2) upwards compatibility with the SMI/MIB and the SNMP;

(3) improved support for multi-protocol entities; and,

(4) textual clean-up of the MIB to improve clarity and readability.

The objects defined in MIB-II have the OBJECT IDENTIFIER prefix:

\[ \text{mib-2} \quad \text{OBJECT IDENTIFIER ::= \{ mgmt 1 } \]

which is identical to the prefix used in MIB-I.

3.1. Deprecated Objects

In order to better prepare implementors for future changes in the MIB, a new term "deprecated" may be used when describing an object. A deprecated object in the MIB is one which must be supported, but one which will most likely be removed from the next version of the MIB (e.g., MIB-III).

MIB-II marks one object as being deprecated:

\[ \text{atTable} \]
As a result of deprecating the atTable object, the entire Address Translation group is deprecated.

Note that no functionality is lost with the deprecation of these objects: new objects providing equivalent or superior functionality are defined in MIB-II.

3.2. Display Strings

In the past, there have been misinterpretations of the MIB as to when a string of octets should contain printable characters, meant to be displayed to a human. As a textual convention in the MIB, the datatype

\[
\text{DisplayString ::= OCTET STRING}
\]

is introduced. A DisplayString is restricted to the NVT ASCII character set, as defined in pages 10-11 of [6].

The following objects are now defined in terms of DisplayString:

- sysDescr
- ifDescr

It should be noted that this change has no effect on either the syntax nor semantics of these objects. The use of the DisplayString notation is merely an artifact of the explanatory method used in MIB-II and future MIBs.

Further it should be noted that any object defined in terms of OCTET STRING may contain arbitrary binary data, in which each octet may take any value from 0 to 255 (decimal).

3.3. Physical Addresses

As a further, textual convention in the MIB, the datatype

\[
\text{PhysAddress ::= OCTET STRING}
\]

is introduced to represent media- or physical-level addresses.

The following objects are now defined in terms of PhysAddress:

- ifPhysAddress
- atPhysAddress
- ipNetToMediaPhysAddress
It should be noted that this change has no effect on either the syntax nor semantics of these objects. The use of the PhysAddress notation is merely an artifact of the explanatory method used in MIB-II and future MIBs.

3.4. The System Group

Four new objects are added to this group:

- sysContact
- sysName
- sysLocation
- sysServices

These provide contact, administrative, location, and service information regarding the managed node.

3.5. The Interfaces Group

The definition of the ifNumber object was incorrect, as it required all interfaces to support IP. (For example, devices without IP, such as MAC-layer bridges, could not be managed if this definition was strictly followed.) The description of the ifNumber object is changed accordingly.

The ifTable object was mistakenly marked as read-write, it has been (correctly) re-designated as not-accessible. In addition, several new values have been added to the ifType column in the ifTable object:

- ppp(23)
- softwareLoopback(24)
- eon(25)
- ethernet-3Mbit(26)
- nsip(27)
- slip(28)
- ultra(29)
- ds3(30)
- sip(31)
- frame-relay(32)

Finally, a new column has been added to the ifTable object:

- ifSpecific

which provides information about information specific to the media being used to realize the interface.
3.6. The Address Translation Group

In MIB-I this group contained a table which permitted mappings from network addresses (e.g., IP addresses) to physical addresses (e.g., MAC addresses). Experience has shown that efficient implementations of this table make two assumptions: a single network protocol environment, and mappings occur only from network address to physical address.

The need to support multi-protocol nodes (e.g., those with both the IP and CLNP active), and the need to support the inverse mapping (e.g., for ES-IS), have invalidated both of these assumptions. As such, the atTable object is declared deprecated.

In order to meet both the multi-protocol and inverse mapping requirements, MIB-II and its successors will allocate up to two address translation tables inside each network protocol group. That is, the IP group will contain one address translation table, for going from IP addresses to physical addresses. Similarly, when a document defining MIB objects for the CLNP is produced (e.g., [7]), it will contain two tables, for mappings in both directions, as this is required for full functionality.

It should be noted that the choice of two tables (one for each direction of mapping) provides for ease of implementation in many cases, and does not introduce undue burden on implementations which realize the address translation abstraction through a single internal table.

3.7. The IP Group

The access attribute of the variable ipForwarding has been changed from read-only to read-write.

In addition, there is a new column to the ipAddrTable object,

    ipAdEntReasmMaxSize

which keeps track of the largest IP datagram that can be re-assembled on a particular interface.

The descriptor of the ipRoutingTable object has been changed to ipRouteTable for consistency with the other IP routing objects. There are also three new columns in the ipRouteTable object,

    ipRouteMask
    ipRouteMetric5
    ipRouteInfo
the first is used for IP routing subsystems that support arbitrary subnet masks, and the latter two are IP routing protocol-specific.

Two new objects are added to the IP group:

- ipNetToMediaTable
- ipRoutingDiscards

the first is the address translation table for the IP group (providing identical functionality to the now deprecated atTable in the address translation group), and the latter provides information when routes are lost due to a lack of buffer space.

3.8. The ICMP Group

There are no changes to this group.

3.9. The TCP Group

Two new variables are added:

- tcpInErrs
- tcpOutRsts

which keep track of the number of incoming TCP segments in error and the number of resets generated by a TCP.

3.10. The UDP Group

A new table:

- udpTable

is added.

3.11. The EGP Group

Experience has indicated a need for additional objects that are useful in EGP monitoring. In addition to making several additions to the egpNeighborTable object, i.e.,

- egpNeighAs
- egpNeighInMsgs
- egpNeighInErrs
- egpNeighOutMsgs
- egpNeighOutErrs
- egpNeighInErrMsgs
- egpNeighOutErrMsgs
egpNeighStateUps
egpNeighStateDowns
egpNeighIntervalHello
egpNeighIntervalPoll
egpNeighMode
egpNeighEventTrigger

a new variable is added:

egpAs

which gives the autonomous system associated with this EGP entity.

3.12. The Transmission Group

MIB-I was lacking in that it did not distinguish between different types of transmission media. A new group, the Transmission group, is allocated for this purpose:

transmission OBJECT IDENTIFIER ::= { mib-2 10 }

When Internet-standard definitions for managing transmission media are defined, the transmission group is used to provide a prefix for the names of those objects.

Typically, such definitions reside in the experimental portion of the MIB until they are "proven", then as a part of the Internet standardization process, the definitions are accordingly elevated and a new object identifier, under the transmission group is defined. By convention, the name assigned is:

type OBJECT IDENTIFIER ::= { transmission number }

where "type" is the symbolic value used for the media in the ifType column of the ifTable object, and "number" is the actual integer value corresponding to the symbol.

3.13. The SNMP Group

The application-oriented working groups of the IETF have been tasked to be receptive towards defining MIB variables specific to their respective applications.

For the SNMP, it is useful to have statistical information. A new group, the SNMP group, is allocated for this purpose:

snmp OBJECT IDENTIFIER ::= { mib-2 11 }
3.14. Changes from RFC 1158

Features of this MIB include:

(1) The managed objects in this document have been defined using the conventions defined in the Internet-standard SMI, as amended by the extensions specified in [14]. It must be emphasized that definitions made using these extensions are semantically identically to those in RFC 1158.

(2) The PhysAddress textual convention has been introduced to represent media addresses.

(3) The ACCESS clause of sysLocation is now read-write.

(4) The definition of sysServices has been clarified.

(5) New ifType values (29-32) have been defined. In addition, the textual-descriptor for the DS1 and E1 interface types has been corrected.

(6) The definition of ipForwarding has been clarified.

(7) The definition of ipRouteType has been clarified.

(8) The ipRouteMetric5 and ipRouteInfo objects have been defined.

(9) The ACCESS clause of tcpConnState is now read-write, to support deletion of the TCB associated with a TCP connection. The definition of this object has been clarified to explain this usage.

(10) The definition of egpNeighEventTrigger has been clarified.

(11) The definition of several of the variables in the new snmp group have been clarified. In addition, the snmpInBadTypes and snmpOutReadOnlys objects are no longer present. (However, the object identifiers associated with those objects are reserved to prevent future use.)

(12) The definition of snmpInReadOnlys has been clarified.

(13) The textual descriptor of the snmpEnableAuthTraps has been changed to snmpEnableAuthenTraps, and the definition has been clarified.
(14) The ipRoutingDiscards object was added.

(15) The optional use of an implementation-dependent, small positive integer was disallowed when identifying instances of the IP address and routing tables.

4. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type’s syntax. Implicitly tied to the notion of an object type’s syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1, subject to the additional requirements imposed by the SNMP.

4.1. Format of Definitions

Section 6 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [14].

5. Overview

Consistent with the IAB directive to produce simple, workable systems in the short-term, the list of managed objects defined here, has been derived by taking only those elements which are considered essential.

This approach of taking only the essential objects is NOT restrictive, since the SMI defined in the companion memo provides...
three extensibility mechanisms: one, the addition of new standard objects through the definitions of new versions of the MIB; two, the addition of widely-available but non-standard objects through the experimental subtree; and three, the addition of private objects through the enterprises subtree. Such additional objects can not only be used for vendor-specific elements, but also for experimentation as required to further the knowledge of which other objects are essential.

The design of MIB-II is heavily influenced by the first extensibility mechanism. Several new variables have been added based on operational experience and need. Based on this, the criteria for including an object in MIB-II are remarkably similar to the MIB-I criteria:

1. An object needed to be essential for either fault or configuration management.

2. Only weak control objects were permitted (by weak, it is meant that tampering with them can do only limited damage). This criterion reflects the fact that the current management protocols are not sufficiently secure to do more powerful control operations.

3. Evidence of current use and utility was required.

4. In MIB-I, an attempt was made to limit the number of objects to about 100 to make it easier for vendors to fully instrument their software. In MIB-II, this limit was raised given the wide technological base now implementing MIB-I.

5. To avoid redundant variables, it was required that no object be included that can be derived from others in the MIB.

6. Implementation specific objects (e.g., for BSD UNIX) were excluded.

7. It was agreed to avoid heavily instrumenting critical sections of code. The general guideline was one counter per critical section per layer.

MIB-II, like its predecessor, the Internet-standard MIB, contains only essential elements. There is no need to allow individual objects to be optional. Rather, the objects are arranged into the following groups:
These groups are the basic unit of conformance: This method is as follows: if the semantics of a group is applicable to an implementation, then it must implement all objects in that group. For example, an implementation must implement the EGP group if and only if it implements the EGP.

There are two reasons for defining these groups: to provide a means of assigning object identifiers; and, to provide a method for implementations of managed agents to know which objects they must implement.

6. Definitions

```
RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
  mgmt, NetworkAddress, IpAddress, Counter, Gauge,
  TimeTicks
  FROM RFC1155-SMI

OBJECT-TYPE
  FROM RFC-1212;

-- This MIB module uses the extended OBJECT-TYPE macro as defined in [14];

-- MIB-II (same prefix as MIB-I)

mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }

-- textual conventions

DisplayString ::= OCTET STRING
-- This data type is used to model textual information taken
-- from the NVT ASCII character set. By convention, objects
-- with this syntax are declared as having
```
-- SIZE (0..255)

PhysAddress ::= OCTET STRING
-- This data type is used to model media addresses. For many
-- types of media, this will be in a binary representation.
-- For example, an ethernet address would be represented as
-- a string of 6 octets.

-- groups in MIB-II

system OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces OBJECT IDENTIFIER ::= { mib-2 2 }
at OBJECT IDENTIFIER ::= { mib-2 3 }
ip OBJECT IDENTIFIER ::= { mib-2 4 }
icmp OBJECT IDENTIFIER ::= { mib-2 5 }
tcp OBJECT IDENTIFIER ::= { mib-2 6 }
udp OBJECT IDENTIFIER ::= { mib-2 7 }
egp OBJECT IDENTIFIER ::= { mib-2 8 }
-- historical (some say hysterical)
cmot OBJECT IDENTIFIER ::= { mib-2 9 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }

snmp OBJECT IDENTIFIER ::= { mib-2 11 }

-- the System group

-- Implementation of the System group is mandatory for all
-- systems. If an agent is not configured to have a value
-- for any of these variables, a string of length 0 is
-- returned.

sysDescr OBJECT-TYPE
  SYNTAX  DisplayString (SIZE (0..255))
  ACCESS  read-only
  STATUS  mandatory
DESCRIPTION

"A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters."
::= { system 1 }

sysObjectID OBJECT-TYPE
SYNTAX  OBJECT IDENTIFIER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION

"The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining 'what kind of box' is being managed. For example, if vendor 'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its 'Fred Router'."
::= { system 2 }

sysUpTime OBJECT-TYPE
SYNTAX  TimeTicks
ACCESS  read-only
STATUS  mandatory
DESCRIPTION

"The time (in hundredths of a second) since the network management portion of the system was last re-initialized."
::= { system 3 }

sysContact OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..255))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION

"The textual identification of the contact person for this managed node, together with information on how to contact this person."
::= { system 4 }

sysName OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..255))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
 "An administratively-assigned name for this
 managed node. By convention, this is the node’s
 fully-qualified domain name."
 ::= { system 5 }

sysLocation OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..255))
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
 "The physical location of this node (e.g.,
 ‘telephone closet, 3rd floor’)."
 ::= { system 6 }

sysServices OBJECT-TYPE
SYNTAX  INTEGER (0..127)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
 "A value which indicates the set of services that
 this entity primarily offers.

The value is a sum. This sum initially takes the
value zero, Then, for each layer, L, in the range
1 through 7, that this node performs transactions
for, 2 raised to (L - 1) is added to the sum. For
example, a node which performs primarily routing
functions would have a value of 4 (2^(3-1)). In
contrast, a node which is a host offering
application services would have a value of 72
(2^(4-1) + 2^(7-1)). Note that in the context of
the Internet suite of protocols, values should be
calculated accordingly:

layer   functionality
 1    physical (e.g., repeaters)
 2    datalink/subnetwork (e.g., bridges)
 3    internet (e.g., IP gateways)
 4    end-to-end (e.g., IP hosts)
 7    applications (e.g., mail relays)

For systems including OSI protocols, layers 5 and
6 may also be counted."
 ::= { system 7 }
the Interfaces group

Implementation of the Interfaces group is mandatory for all systems.

ifNumber OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of network interfaces (regardless of their current state) present on this system."
::= { interfaces 1 }

-- the Interfaces table

The Interfaces table contains information on the entity’s interfaces. Each interface is thought of as being attached to a ‘subnetwork’. Note that this term should not be confused with ‘subnet’ which refers to an addressing partitioning scheme used in the Internet suite of protocols.

ifTable OBJECT-TYPE
SYNTAX SEQUENCE OF IfEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of interface entries. The number of entries is given by the value of ifNumber."
::= { interfaces 2 }

ifEntry OBJECT-TYPE
SYNTAX IfEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "An interface entry containing objects at the subnetwork layer and below for a particular interface."
INDEX { ifIndex }
::= { ifTable 1 }

IfEntry ::= SEQUENCE {
   ifIndex INTEGER,
ifIndex OBJECT-TYPE
SYNTAX      INTEGER
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
   "A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization."

 ::= { ifEntry 1 }

ifDescr OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
   "A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface."

 ::= { ifEntry 2 }

ifType OBJECT-TYPE
   SYNTAX  INTEGER {
      other(1),           -- none of the following
      regular1822(2),
      hdh1822(3),
      ddn-x25(4),
      rfc877-x25(5),
      ethernet-csmacd(6),
      iso88023-csmacd(7),
      iso88024-tokenBus(8),
      iso88025-tokenRing(9),
      iso88026-man(10),
      starLan(11),
      proteon-10Mbit(12),
      proteon-80Mbit(13),
      hyperchannel(14),
      fddi(15),
      lapb(16),
      sdic(17),
      dsl(18),           -- T-1
      el(19),           -- european equiv. of T-1
      basicISDN(20),
      primaryISDN(21),     -- proprietary serial
      propPointToPointSerial(22),
      ppp(23),
      softwareLoopback(24),
      eon(25),           -- CLNP over IP [11]
      ethernet-3Mbit(26),
nsip(27), -- XNS over IP
slip(28), -- generic SLIP
ultra(29), -- ULTRA technologies
ds3(30), -- T-3
sip(31), -- SMDS
frame-relay(32)

ACCESS read-only
STATUS mandatory
DESCRIPTION
"The type of interface, distinguished according to
the physical/link protocol(s) immediately 'below'
the network layer in the protocol stack."
::= { ifEntry 3 }

ifMtu OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The size of the largest datagram which can be
sent/received on the interface, specified in
octets. For interfaces that are used for
transmitting network datagrams, this is the size
of the largest network datagram that can be sent
on the interface."
::= { ifEntry 4 }

ifSpeed OBJECT-TYPE
SYNTAX Gauge
ACCESS read-only
STATUS mandatory
DESCRIPTION
"An estimate of the interface's current bandwidth
in bits per second. For interfaces which do not
vary in bandwidth or for those where no accurate
estimation can be made, this object should contain
the nominal bandwidth."
::= { ifEntry 5 }

ifPhysAddress OBJECT-TYPE
SYNTAX PhysAddress
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The interface’s address at the protocol layer
immediately ‘below’ the network layer in the
protocol stack. For interfaces which do not have
such an address (e.g., a serial line), this object should contain an octet string of zero length."

::= { ifEntry 6 }

ifAdminStatus OBJECT-TYPE
SYNTAX  INTEGER {
    up(1),       -- ready to pass packets
down(2),
testing(3)   -- in some test mode
}
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"The desired state of the interface. The testing(3) state indicates that no operational packets can be passed."

::= { ifEntry 7 }

ifOperStatus OBJECT-TYPE
SYNTAX  INTEGER {
    up(1),       -- ready to pass packets
down(2),
testing(3)   -- in some test mode
}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed."

::= { ifEntry 8 }

ifLastChange OBJECT-TYPE
SYNTAX  TimeTicks
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value."

::= { ifEntry 9 }

ifInOctets OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS mandatory
DESCRIPTION
"The total number of octets received on the interface, including framing characters."
::= { ifEntry 10 }

ifInUcastPkts OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of subnetwork-unicast packets delivered to a higher-layer protocol."
::= { ifEntry 11 }

ifInNUcastPkts OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of non-unicast (i.e., subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol."
::= { ifEntry 12 }

ifInDiscards OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space."
::= { ifEntry 13 }

ifInErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol."
::= { ifEntry 14 }
ifInUnknownProtos OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of packets received via the interface
which were discarded because of an unknown or
unsupported protocol."
 ::= { ifEntry 15 }

ifOutOctets OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of octets transmitted out of the
interface, including framing characters."
 ::= { ifEntry 16 }

ifOutUcastPkt OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of packets that higher-level
protocols requested be transmitted to a
subnetwork-unicast address, including those that
were discarded or not sent."
 ::= { ifEntry 17 }

ifOutNUcastPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of packets that higher-level
protocols requested be transmitted to a non-
unicast (i.e., a subnetwork-broadcast or
subnetwork-multicast) address, including those
that were discarded or not sent."
 ::= { ifEntry 18 }

ifOutDiscards OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of outbound packets which were chosen

to be discarded even though no errors had been
detected to prevent their being transmitted. One
possible reason for discarding such a packet could
be to free up buffer space."

::= { ifEntry 19 }

ifOutErrors OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of outbound packets that could not be
transmitted because of errors."
 ::= { ifEntry 20 }

ifOutQLen OBJECT-TYPE
SYNTAX Gauge
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The length of the output packet queue (in
packets)."
 ::= { ifEntry 21 }

ifSpecific OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A reference to MIB definitions specific to the
particular media being used to realize the
interface. For example, if the interface is
realized by an ethernet, then the value of this
object refers to a document defining objects
specific to ethernet. If this information is not
present, its value should be set to the OBJECT
IDENTIFIER { 0 0 }, which is a syntactically valid
object identifier, and any conformant
implementation of ASN.1 and BER must be able to
generate and recognize this value."
 ::= { ifEntry 22 }

-- the Address Translation group

-- Implementation of the Address Translation group is
-- mandatory for all systems. Note however that this group
-- is deprecated by MIB-II. That is, it is being included
-- solely for compatibility with MIB-I nodes, and will most
-- likely be excluded from MIB-III nodes. From MIB-II and
-- onwards, each network protocol group contains its own
-- address translation tables.

-- The Address Translation group contains one table which is
-- the union across all interfaces of the translation tables
-- for converting a NetworkAddress (e.g., an IP address) into
-- a subnetwork-specific address. For lack of a better term,
-- this document refers to such a subnetwork-specific address
-- as a ‘physical’ address.

-- Examples of such translation tables are: for broadcast
-- media where ARP is in use, the translation table is
-- equivalent to the ARP cache; or, on an X.25 network where
-- non-algorithmic translation to X.121 addresses is
-- required, the translation table contains the
-- NetworkAddress to X.121 address equivalences.

atTable OBJECT-TYPE
SYNTAX  SEQUENCE OF AtEntry
ACCESS  not-accessible
STATUS  deprecated
DESCRIPTION
"The Address Translation tables contain the
NetworkAddress to ‘physical’ address equivalences.
Some interfaces do not use translation tables for
determining address equivalences (e.g., DDN-X.25
has an algorithmic method); if all interfaces are
of this type, then the Address Translation table
is empty, i.e., has zero entries."
::= { at 1 }

atEntry OBJECT-TYPE
SYNTAX  AtEntry
ACCESS  not-accessible
STATUS  deprecated
DESCRIPTION
"Each entry contains one NetworkAddress to
‘physical’ address equivalence."
INDEX  { atIfIndex,
atNetAddress }
::= { atTable 1 }

AtEntry ::= SEQUENCE {
atIfIndex
   INTEGER,  

SNMP Working Group [Page 24]
atIfIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  deprecated
DESCRIPTION
"The interface on which this entry’s equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex."
::= { atEntry 1 }

atPhysAddress OBJECT-TYPE
SYNTAX  PhysAddress
ACCESS  read-write
STATUS  deprecated
DESCRIPTION
"The media-dependent ‘physical’ address.
Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively dissassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object."
::= { atEntry 2 }

atNetAddress OBJECT-TYPE
SYNTAX  NetworkAddress
ACCESS  read-write
STATUS  deprecated
DESCRIPTION
"The NetworkAddress (e.g., the IP address) corresponding to the media-dependent ‘physical’ address."
::= { atEntry 3 }

-- the IP group

-- Implementation of the IP group is mandatory for all
-- systems.

ipForwarding OBJECT-TYPE
SYNTAX INTEGER {
  forwarding(1), -- acting as a gateway
  not-forwarding(2) -- NOT acting as a gateway
}
ACCESS read-write
STATUS mandatory
DESCRIPTION
  "The indication of whether this entity is acting
  as an IP gateway in respect to the forwarding of
  datagrams received by, but not addressed to, this
  entity.  IP gateways forward datagrams.  IP hosts
do not (except those source-routed via the host).

  Note that for some managed nodes, this object may
take on only a subset of the values possible.
  Accordingly, it is appropriate for an agent to
return a 'badValue' response if a management
station attempts to change this object to an
inappropriate value."
::= { ip 1 }

ipDefaultTTL OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
  "The default value inserted into the Time-To-Live
  field of the IP header of datagrams originated at
  this entity, whenever a TTL value is not supplied
  by the transport layer protocol."
::= { ip 2 }

ipInReceives OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The total number of input datagrams received from
  interfaces, including those received in error."
::= { ip 3 }

ipInHdrErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, etc."
 ::= { ip 4 }

ipInAddrErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of input datagrams discarded because the IP address in their IP header’s destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address."
 ::= { ip 5 }

ipForwDatagrams OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source-Route option processing was successful."
 ::= { ip 6 }

ipInUnknownProtos OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol."
::= { ip 7 }

ipInDiscards OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (e.g., for lack of buffer space). Note that this counter does not include any datagrams discarded while awaiting re-assembly."
::= { ip 8 }

ipInDelivers OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of input datagrams successfully delivered to IP user-protocols (including ICMP)."
::= { ip 9 }

ipOutRequests OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams."
::= { ip 10 }

ipOutDiscards OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of output IP datagrams for which no
problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in ipForwardDatagrams if any such packets met this (discretionary) discard criterion."

::= { ip 11 }

ipOutNoRoutes OBJECT-TYPE
  SYNTAX  Counter
  ACCESS  read-only
  STATUS  mandatory
  DESCRIPTION
    "The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwardDatagrams which meet this 'no-route' criterion. Note that this includes any datagrams which a host cannot route because all of its default gateways are down."

::= { ip 12 }

ipReasmTimeout OBJECT-TYPE
  SYNTAX  INTEGER
  ACCESS  read-only
  STATUS  mandatory
  DESCRIPTION
    "The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity."

::= { ip 13 }

ipReasmReqs OBJECT-TYPE
  SYNTAX  Counter
  ACCESS  read-only
  STATUS  mandatory
  DESCRIPTION
    "The number of IP fragments received which needed to be reassembled at this entity."

::= { ip 14 }

ipReasmOKs OBJECT-TYPE
  SYNTAX  Counter
  ACCESS  read-only
  STATUS  mandatory
  DESCRIPTION
    "The number of IP datagrams successfully re-assembled."
::= { ip 15 }

ipReasmFails OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of failures detected by the IP re-
   assembly algorithm (for whatever reason: timed
   out, errors, etc).  Note that this is not
   necessarily a count of discarded IP fragments
   since some algorithms (notably the algorithm in
   RFC 815) can lose track of the number of fragments
   by combining them as they are received."
::= { ip 16 }

ipFragOKs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of IP datagrams that have been
   successfully fragmented at this entity."
::= { ip 17 }

ipFragFails OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of IP datagrams that have been
   discarded because they needed to be fragmented at
   this entity but could not be, e.g., because their
   Don’t Fragment flag was set."
::= { ip 18 }

ipFragCreates OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of IP datagram fragments that have
   been generated as a result of fragmentation at
   this entity."
-- the IP address table

-- The IP address table contains this entity's IP addressing information.

ipAddrTable OBJECT-TYPE
SYNTAX  SEQUENCE OF IpAddrEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
   "The table of addressing information relevant to this entity's IP addresses."
::= { ip 20 }

ipAddrEntry OBJECT-TYPE
SYNTAX  IpAddrEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
   "The addressing information for one of this entity's IP addresses."
INDEX  { ipAdEntAddr }
::= { ipAddrTable 1 }

IpAddrEntry ::=  
SEQUENCE { 
   ipAdEntAddr
      IpAddress,
   ipAdEntIfIndex
      INTEGER,
   ipAdEntNetMask
      IpAddress,
   ipAdEntBcastAddr
      INTEGER,
   ipAdEntReasmMaxSize
      INTEGER (0..65535)
}

ipAdEntAddr OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The IP address to which this entry's addressing information pertains."
::= { ipAddrEntry 1 }
ipAdEntIfIndex OBJECT-TYPE
SYNTAX     INTEGER
ACCESS     read-only
STATUS     mandatory
DESCRIPTION
  "The index value which uniquely identifies the
  interface to which this entry is applicable. The
  interface identified by a particular value of this
  index is the same interface as identified by the
  same value of ifIndex."
 ::= { ipAddrEntry 2 }

ipAdEntNetMask OBJECT-TYPE
SYNTAX     IpAddress
ACCESS     read-only
STATUS     mandatory
DESCRIPTION
  "The subnet mask associated with the IP address of
  this entry. The value of the mask is an IP
  address with all the network bits set to 1 and all
  the hosts bits set to 0."
 ::= { ipAddrEntry 3 }

ipAdEntBcastAddr OBJECT-TYPE
SYNTAX     INTEGER
ACCESS     read-only
STATUS     mandatory
DESCRIPTION
  "The value of the least-significant bit in the IP
  broadcast address used for sending datagrams on
  the (logical) interface associated with the IP
  address of this entry. For example, when the
  Internet standard all-ones broadcast address is
  used, the value will be 1. This value applies to
  both the subnet and network broadcasts addresses
  used by the entity on this (logical) interface."
 ::= { ipAddrEntry 4 }

ipAdEntReasmMaxSize OBJECT-TYPE
SYNTAX     INTEGER (0..65535)
ACCESS     read-only
STATUS     mandatory
DESCRIPTION
  "The size of the largest IP datagram which this
  entity can re-assemble from incoming IP fragmented
  datagrams received on this interface."
 ::= { ipAddrEntry 5 }
-- the IP routing table

-- The IP routing table contains an entry for each route
-- presently known to this entity.

ipRouteTable OBJECT-TYPE
  SYNTAX  SEQUENCE OF IpRouteEntry
  ACCESS  not-accessible
  STATUS  mandatory
  DESCRIPTION
    "This entity’s IP Routing table."
  ::= { ip 21 }

ipRouteEntry OBJECT-TYPE
  SYNTAX  IpRouteEntry
  ACCESS  not-accessible
  STATUS  mandatory
  DESCRIPTION
    "A route to a particular destination."
  INDEX   { ipRouteDest }
  ::= { ipRouteTable 1 }

IpRouteEntry ::= SEQUENCE {
  ipRouteDest
    IpAddress,
  ipRouteIfIndex
    INTEGER,
  ipRouteMetric1
    INTEGER,
  ipRouteMetric2
    INTEGER,
  ipRouteMetric3
    INTEGER,
  ipRouteMetric4
    INTEGER,
  ipRouteNextHop
    IpAddress,
  ipRouteType
    INTEGER,
  ipRouteProto
    INTEGER,
  ipRouteAge
    INTEGER,
  ipRouteMask
    IpAddress,
  ipRouteMetric5
    INTEGER,}
ipRouteInfo
   OBJECT IDENTIFIER

   

   ipRouteDest OBJECT-TYPE
   SYNTAX    IpAddress
   ACCESS    read-write
   STATUS    mandatory
   DESCRIPTION
      "The destination IP address of this route. An
      entry with a value of 0.0.0.0 is considered a
default route. Multiple routes to a single
destination can appear in the table, but access to
such multiple entries is dependent on the table-
access mechanisms defined by the network
management protocol in use."
 ::= { ipRouteEntry 1 }

ipRouteIfIndex OBJECT-TYPE
   SYNTAX    INTEGER
   ACCESS    read-write
   STATUS    mandatory
   DESCRIPTION
      "The index value which uniquely identifies the
local interface through which the next hop of this
route should be reached. The interface identified
by a particular value of this index is the same
interface as identified by the same value of
ifIndex."
 ::= { ipRouteEntry 2 }

ipRouteMetric1 OBJECT-TYPE
   SYNTAX    INTEGER
   ACCESS    read-write
   STATUS    mandatory
   DESCRIPTION
      "The primary routing metric for this route. The
semantics of this metric are determined by the
routing-protocol specified in the route’s
ipRouteProto value. If this metric is not used,
its value should be set to -1."
 ::= { ipRouteEntry 3 }

ipRouteMetric2 OBJECT-TYPE
   SYNTAX    INTEGER
   ACCESS    read-write
   STATUS    mandatory
   DESCRIPTION
      

SNMP Working Group                                             [Page 34]
"An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route’s ipRouteProto value. If this metric is not used, its value should be set to -1."

::= { ipRouteEntry 4 }

ipRouteMetric3 OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
 "An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route’s ipRouteProto value. If this metric is not used, its value should be set to -1."

::= { ipRouteEntry 5 }

ipRouteMetric4 OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
 "An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route’s ipRouteProto value. If this metric is not used, its value should be set to -1."

::= { ipRouteEntry 6 }

ipRouteNextHop OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
 "The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent’s IP address on that interface.)"

::= { ipRouteEntry 7 }

ipRouteType OBJECT-TYPE
SYNTAX  INTEGER {
   other(1), -- none of the following
   invalid(2), -- an invalidated route
}
The type of route. Note that the values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture.

Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively dissassociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

::= { ipRouteEntry 8 }

ipRouteProto OBJECT-TYPE
SYNTAX INTEGER {
    other(1), -- none of the following
    local(2), -- entries
    netmgmt(3), -- set via a network
    icmp(4), -- obtained via ICMP,
    egp(5), -- e.g., Redirect
    ggp(6), -- the remaining values are
    -- all gateway routing
    -- protocols
}
hello(7),
rip(8),
is-is(9),
es-is(10),
ciscoIgrp(11),
bbnSpfIgp(12),
ospf(13),
bgp(14)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The routing mechanism via which this route was
learned. Inclusion of values for gateway routing
protocols is not intended to imply that hosts
should support those protocols."
::= { ipRouteEntry 9 }

ipRouteAge OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The number of seconds since this route was last
updated or otherwise determined to be correct.
Note that no semantics of ‘too old’ can be implied
except through knowledge of the routing protocol
by which the route was learned."
::= { ipRouteEntry 10 }

ipRouteMask OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
DESCRIPTION
"Indicate the mask to be logical-ANDed with the
destination address before being compared to the
value in the ipRouteDest field. For those systems
that do not support arbitrary subnet masks, an
agent constructs the value of the ipRouteMask by
determining whether the value of the correspondent
ipRouteDest field belong to a class-A, B, or C
network, and then using one of:

<table>
<thead>
<tr>
<th>Mask</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.0.0.0</td>
<td>class-A</td>
</tr>
<tr>
<td>255.255.0.0</td>
<td>class-B</td>
</tr>
<tr>
<td>255.255.255.0</td>
<td>class-C</td>
</tr>
</tbody>
</table>
If the value of the ipRouteDest is 0.0.0.0 (a
default route), then the mask value is also
0.0.0.0. It should be noted that all IP routing
subsystems implicitly use this mechanism.

::= { ipRouteEntry 11 }

ipRouteMetric5 OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"An alternate routing metric for this route. The
semantics of this metric are determined by the
routing-protocol specified in the route’s
ipRouteProto value. If this metric is not used,
its value should be set to -1."

::= { ipRouteEntry 12 }

ipRouteInfo OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A reference to MIB definitions specific to the
particular routing protocol which is responsible
for this route, as determined by the value
specified in the route’s ipRouteProto value. If
this information is not present, its value should
be set to the OBJECT IDENTIFIER { 0 0 }, which is
a syntactically valid object identifier, and any
conformant implementation of ASN.1 and BER must be
able to generate and recognize this value."

::= { ipRouteEntry 13 }

-- the IP Address Translation table

-- The IP address translation table contain the IpAddress to
-- 'physical' address equivalences. Some interfaces do not
-- use translation tables for determining address
-- equivalences (e.g., DDN-X.25 has an algorithmic method);
-- if all interfaces are of this type, then the Address
-- Translation table is empty, i.e., has zero entries.

ipNetToMediaTable OBJECT-TYPE
SYNTAX SEQUENCE OF IpNetToMediaEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"The IP Address Translation table used for mapping
from IP addresses to physical addresses."
::= { ip 22 }

ipNetToMediaEntry OBJECT-TYPE
SYNTAX IpNetToMediaEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"Each entry contains one IpAddress to ‘physical’
address equivalence."
INDEX  { ipNetToMediaIfIndex,
           ipNetToMediaNetAddress }
::= { ipNetToMediaTable 1 }

IpNetToMediaEntry ::= 
SEQUENCE {
   ipNetToMediaIfIndex
      INTEGER,
   ipNetToMediaPhysAddress
      PhysAddress,
   ipNetToMediaNetAddress
      IpAddress,
   ipNetToMediaType
      INTEGER
}

ipNetToMediaIfIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The interface on which this entry’s equivalence
is effective. The interface identified by a
particular value of this index is the same
interface as identified by the same value of
ifIndex."
::= { ipNetToMediaEntry 1 }

ipNetToMediaPhysAddress OBJECT-TYPE
SYNTAX PhysAddress
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The media-dependent ‘physical’ address."
::= { ipNetToMediaEntry 2 }
ipNetToMediaNetAddress OBJECT-TYPE  
SYNTAX  IpAddress  
ACCESS  read-write  
STATUS  mandatory  
DESCRIPTION  
"The IpAddress corresponding to the media- 
dependent 'physical' address."
::= { ipNetToMediaEntry 3 }

ipNetToMediaType OBJECT-TYPE  
SYNTAX  INTEGER {  
other(1),        -- none of the following  
invalid(2),      -- an invalidated mapping  
dynamic(3),      
static(4)  
}  
ACCESS  read-write  
STATUS  mandatory  
DESCRIPTION  
"The type of mapping.  

Setting this object to the value invalid(2) has 
the effect of invalidating the corresponding entry 
in the ipNetToMediaTable. That is, it effectively 
dissociates the interface identified with said 
entry from the mapping identified with said entry. 
It is an implementation-specific matter as to 
whether the agent removes an invalidated entry 
from the table. Accordingly, management stations 
must be prepared to receive tabular information 
from agents that corresponds to entries not 
currently in use. Proper interpretation of such 
entries requires examination of the relevant 
ipNetToMediaType object."
::= { ipNetToMediaEntry 4 }

-- additional IP objects

ipRoutingDiscards OBJECT-TYPE  
SYNTAX  Counter  
ACCESS  read-only  
STATUS  mandatory  
DESCRIPTION  
"The number of routing entries which were chosen 
to be discarded even though they are valid. One 
possible reason for discarding such an entry could 
be to free-up buffer space for other routing
entries.
::= { ip 23 }

-- the ICMP group

-- Implementation of the ICMP group is mandatory for all
-- systems.

icmpInMsgs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of ICMP messages which the
entity received. Note that this counter includes
all those counted by icmpInErrors."
::= { icmp 1 }

icmpInErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP messages which the entity
received but determined as having ICMP-specific
errors (bad ICMP checksums, bad length, etc.)."
::= { icmp 2 }

icmpInDestUnreachs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Destination Unreachable
messages received."
::= { icmp 3 }

icmpInTimeExcds OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Time Exceeded messages
received."
::= { icmp 4 }
icmpInParmProbs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The number of ICMP Parameter Problem messages received."
 ::= { icmp 5 }

icmpInSrcQuenchs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The number of ICMP Source Quench messages received."
 ::= { icmp 6 }

icmpInRedirects OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The number of ICMP Redirect messages received."
 ::= { icmp 7 }

icmpInEchos OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The number of ICMP Echo (request) messages received."
 ::= { icmp 8 }

icmpInEchoReps OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The number of ICMP Echo Reply messages received."
 ::= { icmp 9 }

icmpInTimestamps OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of ICMP Timestamp (request) messages received."
::= { icmp 10 }

icmpInTimestampReps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Timestamp Reply messages received."
::= { icmp 11 }

icmpInAddrMasks OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Address Mask Request messages received."
::= { icmp 12 }

icmpInAddrMaskReps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Address Mask Reply messages received."
::= { icmp 13 }

icmpOutMsgs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors."
::= { icmp 14 }

icmpOutErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP messages which this entity did not send due to problems discovered within ICMP
such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter's value."

::= { icmp 15 }

icmpOutDestUnreachs OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The number of ICMP Destination Unreachable messages sent."
::= { icmp 16 }

icmpOutTimeExcds OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The number of ICMP Time Exceeded messages sent."
::= { icmp 17 }

icmpOutParmProbs OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The number of ICMP Parameter Problem messages sent."
::= { icmp 18 }

icmpOutSrcQuenches OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The number of ICMP Source Quench messages sent."
::= { icmp 19 }

icmpOutRedirects OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The number of ICMP Redirect messages sent. For a

SNMP Working Group [Page 44]
host, this object will always be zero, since hosts
do not send redirects."

::= { icmp 20 }

icmpOutEchos OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Echo (request) messages sent."
::= { icmp 21 }

icmpOutEchoReps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Echo Reply messages sent."
::= { icmp 22 }

icmpOutTimestamps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Timestamp (request) messages
sent."
::= { icmp 23 }

icmpOutTimestampReps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Timestamp Reply messages
sent."
::= { icmp 24 }

icmpOutAddrMasks OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Address Mask Request messages
sent."
::= { icmp 25 }
icmpOutAddrMaskReps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of ICMP Address Mask Reply messages sent."
 ::= { icmp 26 }

-- the TCP group

-- Implementation of the TCP group is mandatory for all systems that implement the TCP.

-- Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection -- in question.

tcpRtoAlgorithm OBJECT-TYPE
SYNTAX  INTEGER {
   other(1),   -- none of the following
   constant(2), -- a constant rto
   rsre(3),     -- MIL-STD-1778, Appendix B
   vanj(4)      -- Van Jacobson’s algorithm [10]
}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The algorithm used to determine the timeout value used for retransmitting unacknowledged octets."
 ::= { tcp 1 }

tcpRtoMin OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793."

::= { tcp 2 }

tcpRtoMax OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793."
::= { tcp 3 }

tcpMaxConn OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1."
::= { tcp 4 }

tcpActiveOpens OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state."
::= { tcp 5 }

tcpPassiveOpens OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state."
::= { tcp 6 }
tcpAttemptFails OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of times TCP connections have made a
direct transition to the CLOSED state from either
the SYN-SENT state or the SYN-RCVD state, plus the
number of times TCP connections have made a direct
transition to the LISTEN state from the SYN-RCVD
state."
 ::= { tcp 7 }

tcpEstabResets OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of times TCP connections have made a
direct transition to the CLOSED state from either
the ESTABLISHED state or the CLOSE-WAIT state."
 ::= { tcp 8 }

tcpCurrEstab OBJECT-TYPE
SYNTAX  Gauge
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of TCP connections for which the
current state is either ESTABLISHED or CLOSE-
WAIT."
 ::= { tcp 9 }

tcpInSegs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of segments received, including
those received in error. This count includes
segments received on currently established
connections."
 ::= { tcp 10 }

tcpOutSegs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of segments sent, including
those on current connections but excluding those
containing only retransmitted octets."
 ::= { tcp 11 }

tcpRetransSegs OBJECT-TYPE
SYNTAX   Counter
ACCESS   read-only
STATUS   mandatory
DESCRIPTION
"The total number of segments retransmitted - that
is, the number of TCP segments transmitted
containing one or more previously transmitted
octets."
 ::= { tcp 12 }

-- the TCP Connection table

-- The TCP connection table contains information about this
-- entity’s existing TCP connections.

tcpConnTable OBJECT-TYPE
SYNTAX   SEQUENCE OF TcpConnEntry
ACCESS   not-accessible
STATUS   mandatory
DESCRIPTION
"A table containing TCP connection-specific
information."
 ::= { tcp 13 }

tcpConnEntry OBJECT-TYPE
SYNTAX   TcpConnEntry
ACCESS   not-accessible
STATUS   mandatory
DESCRIPTION
"Information about a particular current TCP
connection. An object of this type is transient,
in that it ceases to exist when (or soon after)
the connection makes the transition to the CLOSED
state."
INDEX   { tcpConnLocalAddress,
tcpConnLocalPort,
tcpConnRemAddress,
tcpConnRemPort }
 ::= { tcpConnTable 1 }
TcpConnEntry ::= 
   SEQUENCE {
      tcpConnState
         INTEGER,
      tcpConnLocalAddress
         IpAddress,
      tcpConnLocalPort
         INTEGER (0..65535),
      tcpConnRemAddress
         IpAddress,
      tcpConnRemPort
         INTEGER (0..65535)
   }

(tcpConnState OBJECT-TYPE
SYNTAX INTEGER {
   closed(1),
   listen(2),
   synSent(3),
   synReceived(4),
   established(5),
   finWait1(6),
   finWait2(7),
   closeWait(8),
   lastAck(9),
   closing(10),
   timeWait(11),
   deleteTCB(12)
}
ACCESS read-write
STATUS mandatory
DESCRIPTION "The state of this TCP connection.

The only value which may be set by a management
station is deleteTCB(12). Accordingly, it is
appropriate for an agent to return a 'badValue'
response if a management station attempts to set
this object to any other value.

If a management station sets this object to the
value deleteTCB(12), then this has the effect of
deleting the TCB (as defined in RFC 793) of the
corresponding connection on the managed node,
resulting in immediate termination of the
connection.

As an implementation-specific option, a RST
segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).

::= { tcpConnEntry 1 }

tcpConnLocalAddress OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used."

::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The local port number for this TCP connection."

::= { tcpConnEntry 3 }

tcpConnRemAddress OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The remote IP address for this TCP connection."

::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE
SYNTAX  INTEGER (0..65535)
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The remote port number for this TCP connection."

::= { tcpConnEntry 5 }

-- additional TCP objects

tcpInErrs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The total number of segments received in error
   (e.g., bad TCP checksums)."
 ::= { tcp 14 }

tcpOutRsts OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of TCP segments sent containing the
   RST flag."
 ::= { tcp 15 }

-- the UDP group

-- Implementation of the UDP group is mandatory for all
-- systems which implement the UDP.

udpInDatagrams OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The total number of UDP datagrams delivered to
   UDP users."
 ::= { udp 1 }

udpNoPorts OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The total number of received UDP datagrams for
   which there was no application at the destination
   port."
 ::= { udp 2 }

udpInErrors OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
   "The number of received UDP datagrams that could
   not be delivered for reasons other than the lack
   of an application at the destination port."
 ::= { udp 3 }
udpOutDatagrams OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of UDP datagrams sent from this entity."
 ::= { udp 4 }

-- the UDP Listener table

-- The UDP listener table contains information about this entity's UDP end-points on which a local application is currently accepting datagrams.

udpTable OBJECT-TYPE
SYNTAX  SEQUENCE OF UdpEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
"A table containing UDP listener information."
 ::= { udp 5 }

udpEntry OBJECT-TYPE
SYNTAX  UdpEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
"Information about a particular current UDP listener."
INDEX  { udpLocalAddress, udpLocalPort }
 ::= { udpTable 1 }

UdpEntry ::= SEQUENCE {
    udpLocalAddress
        IpAddress,
    udpLocalPort
        INTEGER (0..65535)
}

udpLocalAddress OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The local IP address for this UDP listener. In
the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used.

::= { udpEntry 1 }

udpLocalPort OBJECT-TYPE
SYNTAX INTEGER (0..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "The local port number for this UDP listener."
 ::= { udpEntry 2 }

-- the EGP group

-- Implementation of the EGP group is mandatory for all systems which implement the EGP.

egpInMsgs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of EGP messages received without error."
 ::= { egp 1 }

egpInErrors OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of EGP messages received that proved to be in error."
 ::= { egp 2 }

egpOutMsgs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION "The total number of locally generated EGP messages."
 ::= { egp 3 }

egpOutErrors OBJECT-TYPE
SYNTAX Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of locally generated EGP messages not
sent due to resource limitations within an EGP
entity."
 ::= { egp 4 }

-- the EGP Neighbor table

-- The EGP neighbor table contains information about this
-- entity’s EGP neighbors.

egpNeighTable OBJECT-TYPE
SYNTAX  SEQUENCE OF EgpNeighEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
"The EGP neighbor table."
 ::= { egp 5 }

egpNeighEntry OBJECT-TYPE
SYNTAX  EgpNeighEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION
"Information about this entity’s relationship with
a particular EGP neighbor."
INDEX  { egpNeighAddr }
 ::= { egpNeighTable 1 }

EgpNeighEntry ::= SEQUENCE {
  egpNeighState
    INTEGER,
  egpNeighAddr
    IpAddress,
  egpNeighAs
    INTEGER,
  egpNeighInMsgs
    Counter,
  egpNeighOutMsgs
    Counter,
  egpNeighErrs
    Counter,
  egpNeighOutErrs
    Counter,
egpNeighInErrMsgs
Counter,
egpNeighOutErrMsgs
Counter,
egpNeighStateUps
Counter,
egpNeighStateDowns
Counter,
egpNeighIntervalHello
INTEGER,
egpNeighIntervalPoll
INTEGER,
egpNeighMode
INTEGER,
egpNeighEventTrigger
INTEGER
}

egpNeighState OBJECT-TYPE
SYNTAX  INTEGER {
  idle(1),
  acquisition(2),
  down(3),
  up(4),
  cease(5)
}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The EGP state of the local system with respect to
this entry’s EGP neighbor. Each EGP state is
represented by a value that is one greater than
the numerical value associated with said state in
RFC 904."
 ::= { egpNeighEntry 1 }

egpNeighAddr OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The IP address of this entry’s EGP neighbor."
 ::= { egpNeighEntry 2 }

egpNeighAs OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The autonomous system of this EGP peer. Zero
should be specified if the autonomous system
number of the neighbor is not yet known."
::= { egpNeighEntry 3 }

egpNeighInMsgs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of EGP messages received without error
from this EGP peer."
::= { egpNeighEntry 4 }

egpNeighInErrs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of EGP messages received from this EGP
peer that proved to be in error (e.g., bad EGP
checksum)."
::= { egpNeighEntry 5 }

egpNeighOutMsgs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of locally generated EGP messages to
this EGP peer."
::= { egpNeighEntry 6 }

egpNeighOutErrs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of locally generated EGP messages not
sent to this EGP peer due to resource limitations
within an EGP entity."
::= { egpNeighEntry 7 }

egpNeighInErrMsgs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of EGP-defined error messages received from this EGP peer."
::= { egpNeighEntry 8 }

egpNeighOutErrMsgs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of EGP-defined error messages sent to this EGP peer."
::= { egpNeighEntry 9 }

egpNeighStateUps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of EGP state transitions to the UP state with this EGP peer."
::= { egpNeighEntry 10 }

egpNeighStateDowns OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The number of EGP state transitions from the UP state to any other state with this EGP peer."
::= { egpNeighEntry 11 }

egpNeighIntervalHello OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The interval between EGP Hello command retransmissions (in hundredths of a second). This represents the t1 timer as defined in RFC 904."
::= { egpNeighEntry 12 }

egpNeighIntervalPoll OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The interval between EGP poll command
retransmissions (in hundredths of a second). This represents the t3 timer as defined in RFC 904."

```::= { egpNeighEntry 13 }
```

egpNeighMode OBJECT-TYPE
SYNTAX  INTEGER { active(1), passive(2) }
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The polling mode of this EGP entity, either passive or active."
```::= { egpNeighEntry 14 }
```

egpNeighEventTrigger OBJECT-TYPE
SYNTAX  INTEGER { start(1), stop(2) }
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
"A control variable used to trigger operator-initiated Start and Stop events. When read, this variable always returns the most recent value that egpNeighEventTrigger was set to. If it has not been set since the last initialization of the network management subsystem on the node, it returns a value of ‘stop’.

When set, this variable causes a Start or Stop event on the specified neighbor, as specified on pages 8-10 of RFC 904. Briefly, a Start event causes an Idle peer to begin neighbor acquisition and a non-Idle peer to reinitiate neighbor acquisition. A stop event causes a non-Idle peer to return to the Idle state until a Start event occurs, either via egpNeighEventTrigger or otherwise."
```::= { egpNeighEntry 15 }
```

-- additional EGP objects

egpAs OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The autonomous system number of this EGP entity."
```::= { egp 6 }
```
-- the Transmission group

-- Based on the transmission media underlying each interface
-- on a system, the corresponding portion of the Transmission
-- group is mandatory for that system.

-- When Internet-standard definitions for managing
-- transmission media are defined, the transmission group is
-- used to provide a prefix for the names of those objects.

-- Typically, such definitions reside in the experimental
-- portion of the MIB until they are "proven", then as a
-- part of the Internet standardization process, the
-- definitions are accordingly elevated and a new object
-- identifier, under the transmission group is defined. By
-- convention, the name assigned is:
--
-- type OBJECT IDENTIFIER ::= { transmission number }
--
-- where "type" is the symbolic value used for the media in
-- the ifType column of the ifTable object, and "number" is
-- the actual integer value corresponding to the symbol.

-- the SNMP group

-- Implementation of the SNMP group is mandatory for all
-- systems which support an SNMP protocol entity. Some of
-- the objects defined below will be zero-valued in those
-- SNMP implementations that are optimized to support only
-- those functions specific to either a management agent or
-- a management station. In particular, it should be
-- observed that the objects below refer to an SNMP entity,
-- and there may be several SNMP entities residing on a
-- managed node (e.g., if the node is hosting acting as
-- a management station).

snmpInPkts OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
"The total number of Messages delivered to the
SNMP entity from the transport service."
 ::= { snmp 1 }

snmpOutPkts OBJECT-TYPE
SYNTAX      Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Messages which were passed from the SNMP protocol entity to the transport service."
::= { snmp 2 }

snmpInBadVersions OBJECT-TYPE
SYNTAX  Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version."
::= { snmp 3 }

snmpInBadCommunityNames OBJECT-TYPE
SYNTAX  Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Messages delivered to the SNMP protocol entity which used a SNMP community name not known to said entity."
::= { snmp 4 }

snmpInBadCommunityUses OBJECT-TYPE
SYNTAX  Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the Message."
::= { snmp 5 }

snmpInASNParseErrs OBJECT-TYPE
SYNTAX  Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages."
::= { snmp 6 }
-- { snmp 7 } is not used

snmpInTooBigs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
delivered to the SNMP protocol entity and for
which the value of the error-status field is
'tooBig'."
::= { snmp 8 }

snmpInNoSuchNames OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
delivered to the SNMP protocol entity and for
which the value of the error-status field is
'noSuchName'."
::= { snmp 9 }

snmpInBadValues OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
delivered to the SNMP protocol entity and for
which the value of the error-status field is
'badValue'."
::= { snmp 10 }

snmpInReadOnlys OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number valid SNMP PDUs which were
delivered to the SNMP protocol entity and for
which the value of the error-status field is
'readOnly'. It should be noted that it is a
protocol error to generate an SNMP PDU which
contains the value 'readOnly' in the error-status
field, as such this object is provided as a means
of detecting incorrect implementations of the
snmpInGenErrs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is ‘genErr’." :
::= { snmp 12 }

snmpInTotalReqVars OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs."
::= { snmp 13 }

snmpInTotalSetVars OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs."
::= { snmp 14 }

snmpInGetRequests OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNMP protocol entity."
::= { snmp 15 }

snmpInGetNexts OBJECT-TYPE
SYNTAX Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP Get-Next PDUs which have
been accepted and processed by the SNMP protocol
entity."
::= { snmp 16 }

snmpInSetRequests OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP Set-Request PDUs which
have been accepted and processed by the SNMP
protocol entity."
::= { snmp 17 }

snmpInGetResponses OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP Get-Response PDUs which
have been accepted and processed by the SNMP
protocol entity."
::= { snmp 18 }

snmpInTraps OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP Trap PDUs which have
been accepted and processed by the SNMP protocol
entity."
::= { snmp 19 }

snmpOutTooBigs OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
generated by the SNMP protocol entity and for
which the value of the error-status field is
'tooBig.'"
::= { snmp 20 }
snmpOutNoSuchNames OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
generated by the SNMP protocol entity and for
which the value of the error-status is
'noSuchName'.'"
::= { snmp 21 }

snmpOutBadValues OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
generated by the SNMP protocol entity and for
which the value of the error-status field is
'badValue'.'"
::= { snmp 22 }

-- { snmp 23 } is not used

snmpOutGenErrs OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP PDUs which were
generated by the SNMP protocol entity and for
which the value of the error-status field is
'genErr'.'"
::= { snmp 24 }

snmpOutGetRequests OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Get-Request PDUs which
have been generated by the SNMP protocol entity."
::= { snmp 25 }

snmpOutGetNexts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Get-Next PDUs which have
been generated by the SNMP protocol entity."
::= { snmp 26 }

snmpOutSetRequests OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Set-Request PDUs which
have been generated by the SNMP protocol entity."
::= { snmp 27 }

snmpOutGetResponses OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Get-Response PDUs which
have been generated by the SNMP protocol entity."
::= { snmp 28 }

snmpOutTraps OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The total number of SNMP Trap PDUs which have
been generated by the SNMP protocol entity."
::= { snmp 29 }

snmpEnableAuthenTraps OBJECT-TYPE
SYNTAX INTEGER { enabled(1), disabled(2) }
ACCESS read-write
STATUS mandatory
DESCRIPTION
"Indicates whether the SNMP agent process is
permitted to generate authentication-failure traps. The value of this object overrides any
configuration information; as such, it provides a
means whereby all authentication-failure traps may
be disabled.

Note that it is strongly recommended that this
object be stored in non-volatile memory so that it
remains constant between re-initializations of the
network management system."
7. Acknowledgements

This document was produced by the SNMP Working Group:

Anne Ambler, Spider
Karl Auerbach, Sun
Fred Baker, ACC
David Bridgham, Epilogue Technology
Ken Brinkerhoff
Ron Broersma, NOSC
Brian Brown, Synoptics
Jack Brown, US Army
Theodore Brunner, Bellcore
Jeff Buffum, HP
Jeffrey Buffum, HP
John Burress, Wellfleet
Jeffrey D. Case, University of Tennessee at Knoxville
Chris Chiptasso, Spartacus
Paul Ciarfella, DEC
Bob Collet
John Cook, Chipcom
Tracy Cox, Bellcore
James R. Davin, MIT-LCS
Eric Decker, cisco
Kurt Dobbins, Cabletron
Nadya El-Afandi, Network Systems
Gary Ellis, HP
Fred Engle
Mike Erlinger
Mark S. Fedor, PSI
Richard Fox, Synoptics
Karen Frisa, CMU
Stan Froyd, ACC
Chris Gunner, DEC
Fred Harris, University of Tennessee at Knoxville
Ken Hibbard, Xylogics
Ole Jacobsen, Interop
Ken Jones
Satish Joshi, Synoptics
Frank Kastenholz, Racal-Interlan
Shimshon Kaufman, Spartacus
Ken Key, University of Tennessee at Knoxville
Jim Kinder, Fibercom
Alex Koifman, BBN
Christopher Kolb, PSI
Cheryl Krupczak, NCR
Paul Langille, DEC
Martin Lee Schoffstall, PSI
Peter Lin, Vitalink
John Lunny, TWG
Carl Malamud
Gary Malkin, FTP Software, Inc.
Randy Mayhew, University of Tennessee at Knoxville
Keith McClorehie, Hughes LAN Systems
Donna McMaster, David Systems
Lynn Monsanto, Sun
Dave Perkins, 3COM
Jim Reinstedler, Ungerman Bass
Anil Rijsinghani, DEC
Kathy Rinehart, Arnold AFB
Kary Robertson
Marshall T. Rose, PSI (chair)
L. Michael Sabo, NCSC
Jon Saperia, DEC
Greg Satz, cisco
Martin Schoffstall, PSI
John Seligson
Steve Sherry, Xyplex
Fei Shu, NEC
Sam Sjogren, TGV
Mark Sleeper, Sparta
Lance Sprung
Mike St.Johns
Bob Stewart, Xyplex
Emil Sturniold
Kaj Tesink, Bellcore
Geoff Thompson, Synoptics
Dean Throop, Data General
Bill Townsend, Xylogics
Maurice Turcotte, Racal-Milgo
Kannan Varadhou
Sudhanshu Verma, HP
Bill Versteeg, Network Research Corporation
Warren Vik, Interactive Systems
David Waitzman, BBN
Steve Waldbusser, CMU
Dan Wintringhan
David Wood
Wengyik Yeong, PSI
Jeff Young, Cray Research
In addition, the comments of the following individuals are also acknowledged:

Craig A. Finseth, Minnesota Supercomputer Center, Inc.
Jeffrey C. Honig, Cornell University Theory Center
Philip R. Karn, Bellcore

8. References


9. Security Considerations

Security issues are not discussed in this memo.

10. Authors’ Addresses

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

Phone: (415) 966-7934
EMail: kzm@hls.com

Marshall T. Rose
Performance Systems International
5201 Great America Parkway
Suite 3106
Santa Clara, CA  95054

Phone: +1 408 562 6222
EMail: mrose@psi.com
X.500:  rose, psi, us