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

                          Table of Contents

1. Status of this Memo

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

This memo specifies a draft standard for the Internet community. TCP/IP implementations in the Internet which are network manageable are expected to adopt and implement this specification.

Distribution of this memo is unlimited.

2. IAB POLICY STATEMENT

This MIB specification is the first edition of an evolving document defining variables needed for monitoring and control of various components of the Internet. Not all groups of defined variables are mandatory for all Internet components.

For example, the EGP group is mandatory for gateways using EGP but not for hosts which should not be running EGP. Similarly, the TCP group is mandatory for hosts running TCP but not for gateways which aren't running it. What IS mandatory, however, is that all variables of a group be supported if any element of the group is supported.

It is expected that additional MIB groups and variables will be defined over time to accommodate the monitoring and control needs of new or changing components of the Internet. The MIB working group will continue to refine this specification and projects a revision incorporating new requirements in early 1989.

3. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], the Internet Activities Board has directed the Internet Engineering Task Force (IETF) to create two new working groups in the area of network management. One group is charged with the further specification and definition of elements to be included in the Management Information Base. The other is charged with defining the modifications to the Simple Network Management Protocol (SNMP) to accommodate the short-term needs of the network vendor and operator communities. The long-term needs of the Internet community are to be met using the ISO CMIS/CMIP [2,3] framework as a basis. An existing IETF working group, the "NETMAN" group, is already engaged in defining the use of CMIS/CMIP in a TCP/IP network, and will continue with responsibility for addressing the longer-term requirements.

The output of the MIB working group is to be provided to both the SNMP working group and the NETMAN group, so as to ensure compatibility of monitored items for both network management frameworks.

The MIB working group has produced this memo and a companion. The

companion memo [4] defines a Structure for Management Information
(SMI) for use by the managed objects contained in the MIB. This memo
defines the list of managed objects.

The IAB also urged the working groups to be "extremely sensitive to
the need to keep SNMP simple," and recommends that the MIB working
group take as its starting inputs the MIB definitions found in the
High-Level Entity Management Systems (HEMS) RFC 1024 [5], the initial
SNMP specification [6], and the CMIS/CMIP memos [7,8].

Thus, the list of managed objects defined here, has been derived by
taking only those elements which are considered essential. Since
such elements are essential, there is no need to allow the
implementation of individual objects, to be optional. Rather, all
compliant implementations will contain all applicable (see below)
objects defined in this memo.

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
multilateral 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 primary criterion for being considered essential was for an
object to be contained in all of the above referenced MIB
definitions. A few other objects have been included, but only if the
MIB working group believed they are truly essential. The detailed
list of criteria against which potential inclusions in this (initial)
MIB were considered, was:

   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) An attempt was made to limit the number of objects to
      about 100 to make it easier for vendors to fully

instrument their software.

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.

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 Abstract Syntax Notation One (ASN.1) [9].

   The mechanisms used for describing these objects are specified in the
   companion memo. 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 companion memo 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. This memo
   specifies the use of the basic encoding rules of ASN.1 [10].

4.1. Object Groups

   Since this list of managed objects contains only the essential
   elements, there is no need to allow individual objects to be
   optional. Rather, the objects are arranged into the following
   groups:

                    - System
                    - Interfaces
                    - Address Translation
                    - IP
                    - ICMP
                    - TCP
                    - UDP
                    - EGP

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

4.2. Format of Definitions

   The next section contains the specification of all object types
   contained in the MIB. Following the conventions of the companion
   memo, the object types are defined using the following fields:

        OBJECT:

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

        Syntax:
             The abstract syntax for the object type, presented using
             ASN.1. This must resolve to an instance of the ASN.1
             type ObjectSyntax defined in the SMI.

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

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

        Status:
             One of mandatory, optional, or obsolete.

5. Object Definitions

```
RFC1066-MIB { iso org(3) dod(6) internet(1) mgmt(2) 1 }

DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, OBJECT-TYPE, NetworkAddress, IpAddress,
        Counter, Gauge, TimeTicks
          FROM RFC1065-SMI;

mib        OBJECT IDENTIFIER ::= { mgmt 1 }

system     OBJECT IDENTIFIER ::= { mib 1 }
interfaces OBJECT IDENTIFIER ::= { mib 2 }
at         OBJECT IDENTIFIER ::= { mib 3 }
ip         OBJECT IDENTIFIER ::= { mib 4 }
icmp       OBJECT IDENTIFIER ::= { mib 5 }
tcp        OBJECT IDENTIFIER ::= { mib 6 }
udp        OBJECT IDENTIFIER ::= { mib 7 }
egp        OBJECT IDENTIFIER ::= { mib 8 }

END
```

5.1. The System Group

        Implementation of the System group is mandatory for all
        systems.

        OBJECT:
        ----
            sysDescr { system 1 }

        Syntax:
            OCTET STRING

        Definition:
            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.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            sysObjectID { system 2 }

        Syntax:
            OBJECT IDENTIFIER

        Definition:
            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 1.3.6.1.4.1.42, it could assign the identifier
            1.3.6.1.4.1.42.1.1 to its "Fred Router".

        Access:
            read-only.

        Status:
            mandatory.

OBJECT:
----
    sysUpTime { system 3 }

Syntax:
    TimeTicks

Definition:
    The time (in hundredths of a second) since the network
    management portion of the system was last re-initialized.

Access:
    read-only.

Status:
    mandatory.

5.2. The Interfaces Group

        Implementation of the Interfaces group is mandatory for all
        systems.

        OBJECT:
        ----
            ifNumber { interfaces 1 }

        Syntax:
            INTEGER

        Definition:
            The number of network interfaces (regardless of their
            current state) on which this system can send/receive IP
            datagrams.

        Access:
            read-only.

        Status:
            mandatory.

5.2.1. The Interfaces Table

        OBJECT:
        ----
            ifTable { interfaces 2 }

        Syntax:
            SEQUENCE OF IfEntry

        Definition:
            A list of interface entries. The number of entries is
            given by the value of ifNumber.

        Access:
            read-write.

        Status:
            mandatory.

        OBJECT:
        ----
            ifEntry { ifTable 1 }

        Syntax:
            IfEntry ::= SEQUENCE {

```
                    ifIndex
                        INTEGER,
                    ifDescr
                        OCTET STRING,
                    ifType
                        INTEGER,
                    ifMtu
                        INTEGER,
                    ifSpeed
                        Gauge,
                    ifPhysAddress
                        OCTET STRING,
                    ifAdminStatus
                        INTEGER,
                    ifOperStatus
                        INTEGER,
                    ifLastChange
                        TimeTicks,
                    ifInOctets
                        Counter,
                    ifInUcastPkts
                        Counter,
                    ifInNUcastPkts
                        Counter,
                    ifInDiscards
                        Counter,
                    ifInErrors
                        Counter,
                    ifInUnknownProtos
                        Counter,
                    ifOutOctets
                        Counter,
                    ifOutUcastPkts
                        Counter,
                    ifOutNUcastPkts
                        Counter,
                    ifOutDiscards
                        Counter,
                    ifOutErrors
                        Counter,
                    ifOutQLen
                        Gauge
                }

        Definition:
             An interface entry containing objects at the subnetwork
             layer and below for a particular interface.
```

Access:
     read-write.

Status:
     mandatory.

We now consider the individual components of each interface
entry:

OBJECT:
----
     ifIndex { ifEntry 1 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifDescr { ifEntry 2 }

Syntax:
     OCTET STRING

Definition:
     A text 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. The string is intended for presentation to a
     human; it must not contain anything but printable ASCII
     characters.

```
     Access:
          read-only.

     Status:
          mandatory.


     OBJECT:
     -------
          ifType { ifEntry 3 }

     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),
                sdlc(17),
                t1-carrier(18),
                cept(19),           -- european equivalent of T-1
                basicIsdn(20),
                primaryIsdn(21),
                                    -- proprietary serial
                propPointToPointSerial(22)
          }

     Definition:
          The type of interface, distinguished according to the
          physical/link/network protocol(s) immediately "below" IP
          in the protocol stack.

     Access:
          read-only.

     Status:
          mandatory.
```

          OBJECT:
          ----
               ifMtu { ifEntry 4 }

          Syntax:
               INTEGER

          Definition:
               The size of the largest IP datagram which can be
               sent/received on the interface, specified in octets.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ifSpeed { ifEntry 5 }

          Syntax:
               Gauge

          Definition:
               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.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ifPhysAddress { ifEntry 6 }

          Syntax:
               OCTET STRING

          Definition:
               The interface's address at the protocol layer immediately

           "below" IP 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.

      Access:
           read-only.

      Status:
           mandatory.

      OBJECT:
      ----
           ifAdminStatus { ifEntry 7 }

      Syntax:
           INTEGER {
               up(1),   -- ready to pass packets
               down(2),
               testing(3)   -- in some test mode
             }

       Definition:
           The desired state of the interface. The testing(3) state
           indicates that no operational packets can be passed.

       Access:
           read-write.

       Status:
           mandatory.

      OBJECT:
      ----
           ifOperStatus { ifEntry 8 }

      Syntax:
           INTEGER {
               up(1),   -- ready to pass packets
               down(2),
               testing(3)   -- in some test mode
           }

      Definition:
           The current operational state of the interface. The
           testing(3) state indicates that no operational packets
           can be passed.

     Access:
          read-only.

     Status:
          mandatory.

     OBJECT:
     ----
          ifLastChange { ifEntry 9 }

     Syntax:
          TimeTicks

     Definition:
          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.

     Access:
          read-only.

     Status:
          mandatory.

     OBJECT:
     ----
          ifInOctets { ifEntry 10 }

     Syntax:
          Counter

     Definition:
          The total number of octets received on the interface,
          including framing characters.

     Access:
          read-only.

     Status:
          mandatory.

          OBJECT:
          ----
               ifInUcastPkts { ifEntry 11 }

          Syntax:
               Counter

          Definition:
               The number of (subnet) unicast packets delivered to a
               higher-layer protocol.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ifInNUcastPkts { ifEntry 12 }

          Syntax:
               Counter

          Definition:
               The number of non-unicast (i.e., subnet broadcast or
               subnet multicast) packets delivered to a higher-layer
               protocol.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ifInDiscards { ifEntry 13 }

          Syntax:
               Counter

          Definition:
               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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifInErrors { ifEntry 14 }

Syntax:
     Counter

Definition:
     The number of inbound packets that contained errors
     preventing them from being deliverable to a higher-layer
     protocol.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifInUnknownProtos { ifEntry 15 }

Syntax:
     Counter

Definition:
     The number of packets received via the interface which
     were discarded because of an unknown or unsupported
     protocol.

Access:
     read-only.

Status:
     mandatory.

          OBJECT:
          ----
               ifOutOctets { ifEntry 16 }

          Syntax:
               Counter

          Definition:
               The total number of octets transmitted out of the
               interface, including framing characters.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ifOutUcastPkts { ifEntry 17 }

           Syntax:
               Counter

          Definition:
               The total number of packets that higher-level protocols
               requested be transmitted to a subnet-unicast address,
               including those that were discarded or not sent.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
             ifOutNUcastPkts { ifEntry 18 }

          Syntax:
               Counter

          Definition:
               The total number of packets that higher-level protocols
               requested be transmitted to a non-unicast (i.e., a subnet
               broadcast or subnet multicast) address, including those

that were discarded or not sent.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifOutDiscards { ifEntry 19 }

Syntax:
     Counter

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifOutErrors { ifEntry 20 }

Syntax:
     Counter

Definition:
     The number of outbound packets that could not be
     transmitted because of errors.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ifOutQLen { ifEntry 21 }

Syntax:
     Gauge

Definition:
     The length of the output packet queue (in packets).

Access:
     read-only.

Status:
     mandatory.

5.3. The Address Translation Group

    Implementation of the Address Translation group is mandatory
    for all systems.

    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.

            OBJECT:
            ----
                atTable { at 1 }

            Syntax:
                SEQUENCE OF AtEntry

            Definition:
                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.

            Access:
                read-write.

            Status:
                mandatory.

            OBJECT:
            ----
                atEntry { atTable 1 }

            Syntax:
                AtEntry ::= SEQUENCE {
                        atIfIndex

```
                  INTEGER,
             atPhysAddress
                  OCTET STRING,
             atNetAddress
                  NetworkAddress
        }
```

Definition:
     Each entry contains one NetworkAddress to "physical"
     address equivalence.

Access:
     read-write.

Status:
     mandatory.

We now consider the individual components of each Address
Translation table entry:

OBJECT:
----
     atIfIndex { atEntry 1 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-write.

Status:
     mandatory.

OBJECT:
----
     atPhysAddress { atEntry 2 }

Syntax:
     OCTET STRING

          Definition:
                The media-dependent "physical" address.

          Access:
                read-write.

          Status:
                mandatory.

          OBJECT:
          ----
                atNetAddress { atEntry 3 }

          Syntax:
                NetworkAddress

          Definition:
                The NetworkAddress (e.g., the IP address) corresponding to
                the media-dependent "physical" address.

          Access:
                read-write.

          Status:
                mandatory.

5.4. The IP Group

   Implementation of the IP group is mandatory for all systems.

         OBJECT:
         ----
             ipForwarding { ip 1 }

         Syntax:
             INTEGER {
                   gateway(1),   -- entity forwards datagrams
                   host(2)   -- entity does NOT forward datagrams
             }

         Definition:
             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; Hosts do not (except those
             Source-Routed via the host).

         Access:
             read-only.

         Status:
             mandatory.

         OBJECT:
         ----
             ipDefaultTTL { ip 2 }

         Syntax:
             INTEGER

         Definition:
             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.

         Access:
             read-write.

         Status:
             mandatory.

OBJECT:
----
     ipInReceives { ip 3 }

Syntax:
     Counter

Definition:
     The total number of input datagrams received from
     interfaces, including those received in error.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipInHdrErrors { ip 4 }

Syntax:
     Counter

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipInAddrErrors { ip 5 }

Syntax:
     Counter

Definition:
     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.

    Access:
        read-only.

    Status:
        mandatory.

    OBJECT:
    ----
        ipForwDatagrams { ip 6 }

    Syntax:
        Counter

    Definition:
        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.

    Access:
        read-only.

    Status:
        mandatory.

    OBJECT:
    ----
        ipInUnknownProtos { ip 7 }

    Syntax:
        Counter

    Definition:
        The number of locally-addressed datagrams received
        successfully but discarded because of an unknown or
        unsupported protocol.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipInDiscards { ip 8 }

Syntax:
     Counter

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipInDelivers { ip 9 }

Syntax:
     Counter

Definition:
     The total number of input datagrams successfully
     delivered to IP user-protocols (including ICMP).

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipOutRequests { ip 10 }

Syntax:
      Counter

Definition:
      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.

Access:
      read-only.

Status:
      mandatory.

OBJECT:
----
      ipOutDiscards { ip 11 }

Syntax:
      Counter

Definition:
      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 ipForwDatagrams if any such packets
      met this (discretionary) discard criterion.

Access:
      read-only.

Status:
      mandatory.

OBJECT:
----
      ipOutNoRoutes { ip 12 }

Syntax:
      Counter

        Definition:
            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
            ipForwDatagrams which meet this "no-route" criterion.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            ipReasmTimeout { ip 13 }

        Syntax:
            INTEGER

        Definition:
            The maximum number of seconds which received fragments
            are held while they are awaiting reassembly at this
            entity.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            ipReasmReqds { ip 14 }

        Syntax:
            Counter

        Definition:
            The number of IP fragments received which needed to be
            reassembled at this entity.

        Access:
            read-only.

        Status:
            mandatory.

OBJECT:
----
     ipReasmOKs { ip 15 }

Syntax:
     Counter

Definition:
     The number of IP datagrams successfully re-assembled.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipReasmFails { ip 16 }

Syntax:
     Counter

Definition:
     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 RFC 815's) can
     lose track of the number of fragments by combining them
     as they are received.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipFragOKs { ip 17 }

Syntax:
     Counter

          Definition:
               The number of IP datagrams that have been successfully
               fragmented at this entity.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ipFragFails { ip 18 }

          Syntax:
               Counter

          Definition:
               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.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               ipFragCreates { ip 19 }

          Syntax:
               Counter

          Definition:
               The number of IP datagram fragments that have been
               generated as a result of fragmentation at this entity.

          Access:
               read-only.

          Status:
               mandatory.

5.4.1. The IP Address Table

    The Ip Address table contains this entity's IP addressing
    information.

            OBJECT:
            ----
                ipAddrTable { ip 20 }

            Syntax:
                SEQUENCE OF IpAddrEntry

            Definition:
                The table of addressing information relevant to this
                entity's IP addresses.

            Access:
                read-only.

            Status:
                mandatory.

            OBJECT:
            ----
                ipAddrEntry { ipAddrTable 1 }

            Syntax:
                IpAddrEntry ::= SEQUENCE {
                    ipAdEntAddr
                        IpAddress,
                    ipAdEntIfIndex
                        INTEGER,
                    ipAdEntNetMask
                        IpAddress,
                    ipAdEntBcastAddr
                        INTEGER
                }

            Definition:
                The addressing information for one of this entity's IP
                addresses.

            Access:
                read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            ipAdEntAddr { ipAddrEntry 1 }

        Syntax:
            IpAddress

        Definition:
            The IP address to which this entry's addressing
            information pertains.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            ipAdEntIfIndex { ipAddrEntry 2 }

        Syntax:
            INTEGER

        Definition:
            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.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            ipAdEntNetMask { ipAddrEntry 3 }

        Syntax:
             IpAddress

        Definition:
             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.

        Access:
             read-only.

        Status:
             mandatory.

        OBJECT:
        ----
             ipAdEntBcastAddr { ipAddrEntry 4 }

        Syntax:
             INTEGER

        Definition:
             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.

        Access:
             read-only.

        Status:
             mandatory.

5.4.2. The IP Routing Table

   The IP Routing Table contains an entry for each route
   presently known to this entity. Note that the action to be
   taken in response to a request to read a non-existent entry,
   is specific to the network management protocol being used.

        OBJECT:
        ----
             ipRoutingTable { ip 21 }

```
Syntax:
    SEQUENCE OF IpRouteEntry

Definition:
    This entity's IP Routing table.

Access:
    read-write.

Status:
    mandatory.


OBJECT:
-------
    ipRouteEntry { ipRoutingTable 1 }

Syntax:
    IpRouteEntry ::= SEQUENCE {
        ipRouteDest
            IpAddress,
        ipRouteIfIndex
            INTEGER,
        ipRouteMetric1
            INTEGER,
        ipRouteMetric2
            INTEGER,
        ipRouteMetric3
            INTEGER,
        ipRouteMetric4
            INTEGER,
        ipRouteNextHop
            IpAddress,
        ipRouteType
            INTEGER,
        ipRouteProto
            INTEGER,
        ipRouteAge
            INTEGER
    }

Definition:
    A route to a particular destination.

Access:
    read-write.
```

Status:
     mandatory.

We now consider the individual components of each route in the
IP Routing Table:

OBJECT:
----
     ipRouteDest { ipRouteEntry 1 }

Syntax:
     IpAddress

Definition:
     The destination IP address of this route. An entry with
     a value of 0.0.0.0 is considered a default route.
     Multiple such default routes 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.

Access:
     read-write.

Status:
     mandatory.

OBJECT:
----
     ipRouteIfIndex { ipRouteEntry 2 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-write.

Status:
     mandatory.

          OBJECT:
          ----
              ipRouteMetric1 { ipRouteEntry 3 }

          Syntax:
              INTEGER

          Definition:
              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.

          Access:
              read-write.

          Status:
              mandatory.

          OBJECT:
          ----
              ipRouteMetric2 { ipRouteEntry 4 }

          Syntax:
              INTEGER

          Definition:
              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.

          Access:
              read-write.

          Status:
              mandatory.

          OBJECT:
          ----
              ipRouteMetric3 { ipRouteEntry 5 }

          Syntax:
              INTEGER

Definition:
     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.

 Access:
     read-write.

 Status:
     mandatory.

OBJECT:
----
     ipRouteMetric4 { ipRouteEntry 6 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-write.

Status:
     mandatory.

OBJECT:
----
     ipRouteNextHop { ipRouteEntry 7 }

Syntax:
     IpAddress

Definition:
     The IP address of the next hop of this route.

Access:
     read-write.

Status:
     mandatory.

```
          OBJECT:
          -------
               ipRouteType { ipRouteEntry 8 }

          Syntax:
               INTEGER {
                     other(1),          -- none of the following

                     invalid(2),        -- an invalidated route

                                        -- route to directly
                     direct(3),         -- connected (sub-)network

                                        -- route to a non-local
                     remote(4),         -- host/network/sub-network
               }

          Definition:
               The type of route.

          Access:
               read-write.

          Status:
               mandatory.


          OBJECT:
          -------
               ipRouteProto { ipRouteEntry 9 }

          Syntax:
               INTEGER {
                     other(1),        -- none of the following

                                      -- non-protocol information,
                                      -- e.g., manually configured
                     local(2),        -- entries

                                      -- set via a network management
                     netmgmt(3),      -- protocol

                                      -- obtained via ICMP,
                     icmp(4),         -- e.g., Redirect

                                      -- the remaining values are
                                      -- all gateway routing protocols
                     egp(5),
```

```
            ggp(6),
            hello(7),
            rip(8),
            is-is(9),
            es-is(10),
            ciscoIgrp(11),
            bbnSpfIgp(12),
            oigp(13)
        }
```

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     ipRouteAge { ipRouteEntry 10 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-write.

Status:
     mandatory.

5.5. The ICMP Group

     Implementation of the ICMP group is mandatory for all systems.

     The ICMP group contains the ICMP input and output statistics.

     Note that individual counters for ICMP message (sub-)codes have been
     omitted from this (version of the) MIB for simplicity.

          OBJECT:
          ----
               icmpInMsgs { icmp 1 }

          Syntax:
               Counter

          Definition:
               The total number of ICMP messages which the entity
               received. Note that this counter includes all those
               counted by icmpInErrors.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInErrors { icmp 2 }

          Syntax:
               Counter

          Definition:
               The number of ICMP messages which the entity received but
               determined as having errors (bad ICMP checksums, bad
               length, etc.).

          Access:
               read-only.

          Status:
               mandatory.

        OBJECT:
        ----
            icmpInDestUnreachs { icmp 3 }

        Syntax:
            Counter

        Definition:
            The number of ICMP Destination Unreachable messages
            received.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            icmpInTimeExcds { icmp 4 }

        Syntax:
            Counter

        Definition:
            The number of ICMP Time Exceeded messages received.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            icmpInParmProbs { icmp 5 }

        Syntax:
            Counter

        Definition:
            The number of ICMP Parameter Problem messages received.

        Access:
            read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInSrcQuenchs { icmp 6 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Source Quench messages received.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInRedirects { icmp 7 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Redirect messages received.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInEchos { icmp 8 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Echo (request) messages received.

    Access:
         read-only.

    Status:
         mandatory.

    OBJECT:
    ----
         icmpInEchoReps { icmp 9 }

    Syntax:
         Counter

    Definition:
         The number of ICMP Echo Reply messages received.

    Access:
         read-only.

    Status:
         mandatory.

    OBJECT:
    ----
         icmpInTimestamps { icmp 10 }

    Syntax:
         Counter

    Definition:
         The number of ICMP Timestamp (request) messages received.

    Access:
         read-only.

    Status:
         mandatory.

    OBJECT:
    ----
         icmpInTimestampReps { icmp 11 }

    Syntax:
         Counter

          Definition:
               The number of ICMP Timestamp Reply messages received.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInAddrMasks { icmp 12 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Address Mask Request messages
               received.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpInAddrMaskReps { icmp 13 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Address Mask Reply messages received.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpOutMsgs { icmp 14 }

    Syntax:
          Counter

    Definition:
          The total number of ICMP messages which this entity
          attempted to send. Note that this counter includes all
          those counted by icmpOutErrors.

    Access:
          read-only.

    Status:
          mandatory.

    OBJECT:
    ----
          icmpOutErrors { icmp 15 }

    Syntax:
          Counter

    Definition:
          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.

    Access:
          read-only.

    Status:
          mandatory.

    OBJECT:
    ----
          icmpOutDestUnreachs { icmp 16 }

    Syntax:
          Counter

    Definition:
          The number of ICMP Destination Unreachable messages sent.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpOutTimeExcds { icmp 17 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Time Exceeded messages sent.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpOutParmProbs { icmp 18 }

          Syntax:
               Counter

          Definition:
               The number of ICMP Parameter Problem messages sent.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               icmpOutSrcQuenchs { icmp 19 }

          Syntax:
               Counter

          Definition:
                The number of ICMP Source Quench messages sent.

          Access:
                read-only.

          Status:
                mandatory.

          OBJECT:
          ----
                icmpOutRedirects { icmp 20 }

          Syntax:
                Counter

          Definition:
                The number of ICMP Redirect messages sent.

          Access:
                read-only.

          Status:
                mandatory.

          OBJECT:
          ----
                icmpOutEchos { icmp 21 }

          Syntax:
                Counter

          Definition:
                The number of ICMP Echo (request) messages sent.

          Access:
                read-only.

          Status:
                mandatory.

          OBJECT:
          ----
                icmpOutEchoReps { icmp 22 }

Syntax:
     Counter

Definition:
     The number of ICMP Echo Reply messages sent.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     icmpOutTimestamps { icmp 23 }

Syntax:
     Counter

Definition:
     The number of ICMP Timestamp (request) messages sent.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     icmpOutTimestampReps { icmp 24 }

Syntax:
     Counter

Definition:
     The number of ICMP Timestamp Reply messages sent.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     icmpOutAddrMasks { icmp 25 }

Syntax:
     Counter

Definition:
     The number of ICMP Address Mask Request messages sent.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     icmpOutAddrMaskReps { icmp 26 }

Syntax:
     Counter

Definition:
     The number of ICMP Address Mask Reply messages sent.

Access:
     read-only.

Status:
     mandatory.

5.6. The TCP Group

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

     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.

               OBJECT:
               ----
                    tcpRtoAlgorithm { tcp 1 }

               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 [11]
                    }

               Definition:
                    The algorithm used to determine the timeout value used
                    for retransmitting unacknowledged octets.

               Access:
                    read-only.

               Status:
                    mandatory.

               OBJECT:
               ----
                    tcpRtoMin { tcp 2 }

               Syntax:
                    INTEGER

               Definition:
                    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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpRtoMax { tcp 3 }

Syntax:
     INTEGER

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpMaxConn { tcp 4 }

Syntax:
     INTEGER

Definition:
     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".

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpActiveOpens { tcp 5 }

Syntax:
     Counter

Definition:
     The number of times TCP connections have made a direct
     transition to the SYN-SENT state from the CLOSED
     state.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpPassiveOpens { tcp 6 }

Syntax:
     Counter

Definition:
     The number of times TCP connections have made a direct
     transition to the SYN-RCVD state from the LISTEN
     state.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpAttemptFails { tcp 7 }

Syntax:
     Counter

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpEstabResets { tcp 8 }

Syntax:
     Counter

Definition:
     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.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpCurrEstab { tcp 9 }

Syntax:
     Gauge

Definition:
     The number of TCP connections for which the current
     state is either ESTABLISHED or CLOSE-WAIT.

Access:
     read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               tcpInSegs { tcp 10 }

          Syntax:
               Counter

          Definition:
               The total number of segments received, including those
               received in error. This count includes segments
               received on currently established connections.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               tcpOutSegs { tcp 11 }

          Syntax:
               Counter

          Definition:
               The total number of segments sent, including those on
               current connections but excluding those containing
               only retransmitted octets.

          Access:
               read-only.

          Status:
               mandatory.

          OBJECT:
          ----
               tcpRetransSegs { tcp 12 }

          Syntax:
               Counter

Definition:
     The total number of segments retransmitted - that is,
     the number of TCP segments transmitted containing one
     or more previously transmitted octets.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpConnTable { tcp 13 }

Syntax:
     SEQUENCE OF TcpConnEntry

Definition:
     A table containing TCP connection-specific
     information.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     tcpConnEntry { tcpConnTable 1 }

Syntax:
     TcpConnEntry ::= SEQUENCE {
         tcpConnState
             INTEGER,
         tcpConnLocalAddress
             IpAddress,
         tcpConnLocalPort
             INTEGER (0..65535),
         tcpConnRemAddress
             IpAddress,
         tcpConnRemPort
             INTEGER (0..65535)
     }

        Definition:
             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.

        Access:
             read-only.

        Status:
             mandatory.

        OBJECT:
        ----
             tcpConnState { tcpConnEntry 1 }

        Syntax:
             INTEGER {
                   closed(1),
                   listen(2),
                   synSent(3),
                   synReceived(4),
                   established(5),
                   finWait1(6),
                   finWait2(7),
                   closeWait(8),
                   lastAck(9),
                   closing(10),
                   timeWait(11)
             }

        Definition:
             The state of this TCP connection.

        Access:
             read-only.

        Status:
             mandatory.

        OBJECT:
        ----
             tcpConnLocalAddress { tcpConnEntry 2 }

        Syntax:
             IpAddress

        Definition:
            The local IP address for this TCP connection.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            tcpConnLocalPort { tcpConnEntry 3 }

        Syntax:
            INTEGER (0..65535)

        Definition:
            The local port number for this TCP connection.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            tcpConnRemAddress { tcpConnEntry 4 }

        Syntax:
            IpAddress

        Definition:
            The remote IP address for this TCP connection.

        Access:
            read-only.

        Status:
            mandatory.

        OBJECT:
        ----
            tcpConnRemPort { tcpConnEntry 5 }

Syntax:
    INTEGER (0..65535)

Definition:
    The remote port number for this TCP connection.

Access:
    read-only.

Status:
    mandatory.

5.7. The UDP Group

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

           OBJECT:
           ----
               udpInDatagrams { udp 1 }

           Syntax:
               Counter

           Definition:
               The total number of UDP datagrams delivered to UDP
               users.

           Access:
               read-only.

           Status:
               mandatory.

           OBJECT:
           ----
               udpNoPorts { udp 2 }

           Syntax:
               Counter

           Definition:
               The total number of received UDP datagrams for which
               there was no application at the destination port.

           Access:
               read-only.

           Status:
               mandatory.

           OBJECT:
           ----
               udpInErrors { udp 3 }

           Syntax:
               Counter

Definition:
     The number of received UDP datagrams that could not be
     delivered for reasons other than the lack of an
     application at the destination port.

Access:
     read-only.

Status:
     mandatory.

OBJECT:
----
     udpOutDatagrams { udp 4 }

Syntax:
     Counter

Definition:
     The total number of UDP datagrams sent from this
     entity.

Access:
     read-only.

Status:
     mandatory.

5.8. The EGP Group

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

               OBJECT:
               ----
                    egpInMsgs { egp 1 }

               Syntax:
                    Counter

               Definition:
                    The number of EGP messages received without error.

               Access:
                    read-only.

               Status:
                    mandatory.

               OBJECT:
               ----
                    egpInErrors { egp 2 }

               Syntax:
                    Counter

               Definition:
                    The number of EGP messages received that proved to be
                    in error.

               Access:
                    read-only.

               Status:
                    mandatory.

               OBJECT:
               ----
                    egpOutMsgs { egp 3 }

               Syntax:
                    Counter

             Definition:
                  The total number of locally generated EGP messages.

             Access:
                  read-only.

             Status:
                  mandatory.

             OBJECT:
             ----
                  egpOutErrors { egp 4 }

             Syntax:
                  Counter

             Definition:
                  The number of locally generated EGP messages not sent
                  due to resource limitations within an EGP entity.

             Access:
                  read-only.

             Status:
                  mandatory.

5.8.1. The EGP Neighbor Table

   The Egp Neighbor table contains information about this entity's EGP
   neighbors.

             OBJECT:
             ----
                  egpNeighTable { egp 5 }

             Syntax:
                  SEQUENCE OF EgpNeighEntry

             Definition:
                  The EGP neighbor table.

             Access:
                  read-only.

             Status:
                  mandatory.

```
            OBJECT:
            -------
                 egpNeighEntry { egpNeighTable 1 }

            Syntax:
                 EgpNeighEntry ::= SEQUENCE {
                       egpNeighState
                           INTEGER,
                       egpNeighAddr
                           IpAddress
                 }

            Definition:
                 Information about this entity's relationship with a
                 particular EGP neighbor.

            Access:
                 read-only.

            Status:
                 mandatory.


            We now consider the individual components of each EGP
            neighbor entry:


            OBJECT:
            -------
                 egpNeighState { egpNeighEntry 1 }

            Syntax:
                 INTEGER {
                       idle(1),
                       acquisition(2),
                       down(3),
                       up(4),
                       cease(5)
                 }

            Definition:
                 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.

            Access:
                 read-only.
```

Status:
     mandatory.

OBJECT:
----
     egpNeighAddr { egpNeighEntry 2 }

Syntax:
     IpAddress

Definition:
     The IP address of this entry's EGP neighbor.

Access:
     read-only.

Status:
     mandatory.

6. Definitions

```
RFC1066-MIB { iso org(3) dod(6) internet(1) mgmt(2) 1 }

DEFINITIONS ::= BEGIN

IMPORTS
        mgmt, OBJECT-TYPE, NetworkAddress, IpAddress,
        Counter, Gauge, TimeTicks
            FROM RFC1065-SMI;

  mib        OBJECT IDENTIFIER ::= { mgmt 1 }

  system     OBJECT IDENTIFIER ::= { mib 1 }
  interfaces OBJECT IDENTIFIER ::= { mib 2 }
  at         OBJECT IDENTIFIER ::= { mib 3 }
  ip         OBJECT IDENTIFIER ::= { mib 4 }
  icmp       OBJECT IDENTIFIER ::= { mib 5 }
  tcp        OBJECT IDENTIFIER ::= { mib 6 }
  udp        OBJECT IDENTIFIER ::= { mib 7 }
  egp        OBJECT IDENTIFIER ::= { mib 8 }

  -- object types

  -- the System group

  sysDescr OBJECT-TYPE
          SYNTAX  OCTET STRING
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 1 }

  sysObjectID OBJECT-TYPE
          SYNTAX  OBJECT IDENTIFIER
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 2 }

  sysUpTime OBJECT-TYPE
          SYNTAX  TimeTicks
          ACCESS  read-only
          STATUS  mandatory
          ::= { system 3 }

  -- the Interfaces group

  ifNumber OBJECT-TYPE
          SYNTAX  INTEGER
```

```
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { interfaces 1 }

                -- the Interfaces table

                ifTable OBJECT-TYPE
                        SYNTAX  SEQUENCE OF IfEntry
                        ACCESS  read-write
                        STATUS  mandatory
                        ::= { interfaces 2 }

                ifEntry OBJECT-TYPE
                        SYNTAX  IfEntry
                        ACCESS  read-write
                        STATUS  mandatory
                        ::= { ifTable 1 }

                IfEntry ::= SEQUENCE {
                    ifIndex
                        INTEGER,
                    ifDescr
                        OCTET STRING,
                    ifType
                        INTEGER,
                    ifMtu
                        INTEGER,
                    ifSpeed
                        Gauge,
                    ifPhysAddress
                        OCTET STRING,
                    ifAdminStatus
                        INTEGER,
                    ifOperStatus
                        INTEGER,
                    ifLastChange
                        TimeTicks,
                    ifInOctets
                        Counter,
                    ifInUcastPkts
                        Counter,
                    ifInNUcastPkts
                        Counter,
                    ifInDiscards
                        Counter,
                    ifInErrors
                        Counter,
                    ifInUnknownProtos
```

```
                       Counter,
                 ifOutOctets
                       Counter,
                 ifOutUcastPkts
                       Counter,
                 ifOutNUcastPkts
                       Counter,
                 ifOutDiscards
                       Counter,
                 ifOutErrors
                       Counter,
                 ifOutQLen
                       Gauge
            }

            ifIndex OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 1 }

            ifDescr OBJECT-TYPE
                    SYNTAX  OCTET STRING
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { 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),
                            sdlc(17),
                            t1-carrier(18),
                            cept(19),
```

```
                        basicIsdn(20),
                        primaryIsdn(21),
                                        -- proprietary serial
                        propPointToPointSerial(22)
                    }
              ACCESS  read-only
              STATUS  mandatory
              ::= { ifEntry 3 }

      ifMtu OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-only
              STATUS  mandatory
              ::= { ifEntry 4 }

      ifSpeed OBJECT-TYPE
              SYNTAX  Gauge
              ACCESS  read-only
              STATUS  mandatory
              ::= { ifEntry 5 }

      ifPhysAddress OBJECT-TYPE
              SYNTAX  OCTET STRING
              ACCESS  read-only
              STATUS  mandatory
              ::= { 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
              ::= { 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
              ::= { ifEntry 8 }

      ifLastChange OBJECT-TYPE
```

```
                    SYNTAX  TimeTicks
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 9 }

            ifInOctets OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 10 }

            ifInUcastPkts OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::=  { ifEntry 11 }

            ifInNUcastPkts OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 12 }

            ifInDiscards OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 13 }

            ifInErrors OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 14 }

            ifInUnknownProtos OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 15 }

            ifOutOctets OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ifEntry 16 }

            ifOutUcastPkts OBJECT-TYPE
```

```
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { ifEntry 17 }

             ifOutNUcastPkts OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { ifEntry 18 }

             ifOutDiscards OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { ifEntry 19 }

             ifOutErrors OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { ifEntry 20 }

             ifOutQLen OBJECT-TYPE
                     SYNTAX  Gauge
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { ifEntry 21 }

             -- the Address Translation group

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

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

             AtEntry ::= SEQUENCE {
                 atIfIndex
                     INTEGER,
                 atPhysAddress
                     OCTET STRING,
```

```
                    atNetAddress
                        NetworkAddress
                }

        atIfIndex OBJECT-TYPE
                SYNTAX  INTEGER
                ACCESS  read-write
                STATUS  mandatory
                ::= { atEntry 1 }

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

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

        -- the IP group

        ipForwarding OBJECT-TYPE
                SYNTAX  INTEGER {
            gateway(1), -- entity forwards datagrams
            host(2)     -- entity does NOT forward datagrams
                    }
                ACCESS  read-only
                STATUS  mandatory
                ::= { ip 1 }

        ipDefaultTTL OBJECT-TYPE
                SYNTAX  INTEGER
                ACCESS  read-write
                STATUS  mandatory
                ::= { ip 2 }

        ipInReceives OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { ip 3 }

        ipInHdrErrors OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
```

```
                   STATUS  mandatory
                   ::= { ip 4 }


           ipInAddrErrors OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 5 }


           ipForwDatagrams OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 6 }


           ipInUnknownProtos OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 7 }


           ipInDiscards OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 8 }


           ipInDelivers OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 9 }


           ipOutRequests OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 10 }


           ipOutDiscards OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
                   STATUS  mandatory
                   ::= { ip 11 }


           ipOutNoRoutes OBJECT-TYPE
                   SYNTAX  Counter
                   ACCESS  read-only
```

```
                    STATUS  mandatory
                    ::= { ip 12 }

            ipReasmTimeout OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 13 }

            ipReasmReqds OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 14 }

            ipReasmOKs OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 15 }

            ipReasmFails OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 16 }

            ipFragOKs OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 17 }

            ipFragFails OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 18 }

            ipFragCreates OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { ip 19 }

            -- the IP Interface table

            ipAddrTable OBJECT-TYPE
```

```
                       SYNTAX   SEQUENCE OF IpAddrEntry
                       ACCESS   read-only
                       STATUS   mandatory
                       ::= { ip 20 }

              ipAddrEntry OBJECT-TYPE
                       SYNTAX   IpAddrEntry
                       ACCESS   read-only
                       STATUS   mandatory
                       ::= { ipAddrTable 1 }

              IpAddrEntry ::= SEQUENCE {
                   ipAdEntAddr
                       IpAddress,
                   ipAdEntIfIndex
                       INTEGER,
                   ipAdEntNetMask
                       IpAddress,
                   ipAdEntBcastAddr
                       INTEGER
              }

              ipAdEntAddr OBJECT-TYPE
                       SYNTAX   IpAddress
                       ACCESS   read-only
                       STATUS   mandatory
                       ::=  { ipAddrEntry 1 }

              ipAdEntIfIndex OBJECT-TYPE
                       SYNTAX   INTEGER
                       ACCESS   read-only
                       STATUS   mandatory
                       ::=  { ipAddrEntry 2 }

              ipAdEntNetMask OBJECT-TYPE
                       SYNTAX   IpAddress
                       ACCESS   read-only
                       STATUS   mandatory
                       ::=  { ipAddrEntry 3 }

              ipAdEntBcastAddr OBJECT-TYPE
                       SYNTAX   INTEGER
                       ACCESS   read-only
                       STATUS   mandatory
                       ::= { ipAddrEntry 4 }

              -- the IP Routing table
```

```
                ipRoutingTable OBJECT-TYPE
                        SYNTAX   SEQUENCE OF IpRouteEntry
                        ACCESS   read-write
                        STATUS   mandatory
                        ::= { ip 21 }

                ipRouteEntry OBJECT-TYPE
                        SYNTAX   IpRouteEntry
                        ACCESS   read-write
                        STATUS   mandatory
                        ::= { ipRoutingTable 1 }

                IpRouteEntry ::= SEQUENCE {
                    ipRouteDest
                        IpAddress,
                    ipRouteIfIndex
                        INTEGER,
                    ipRouteMetric1
                        INTEGER,
                    ipRouteMetric2
                        INTEGER,
                    ipRouteMetric3
                        INTEGER,
                    ipRouteMetric4
                        INTEGER,
                    ipRouteNextHop
                        IpAddress,
                    ipRouteType
                        INTEGER,
                    ipRouteProto
                        INTEGER,
                    ipRouteAge
                        INTEGER
                }

                ipRouteDest OBJECT-TYPE
                        SYNTAX   IpAddress
                        ACCESS   read-write
                        STATUS   mandatory
                        ::= { ipRouteEntry 1 }

                ipRouteIfIndex  OBJECT-TYPE
                        SYNTAX   INTEGER
                        ACCESS   read-write
                        STATUS   mandatory
                        ::= { ipRouteEntry 2 }

                ipRouteMetric1 OBJECT-TYPE
```

```
                    SYNTAX  INTEGER
                    ACCESS  read-write
                    STATUS  mandatory
                    ::= { ipRouteEntry 3 }

            ipRouteMetric2 OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-write
                    STATUS  mandatory
                    ::= { ipRouteEntry 4 }

            ipRouteMetric3 OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-write
                    STATUS  mandatory
                    ::= { ipRouteEntry 5 }

            ipRouteMetric4 OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-write
                    STATUS  mandatory
                    ::= { ipRouteEntry 6 }

            ipRouteNextHop OBJECT-TYPE
                    SYNTAX  IpAddress
                    ACCESS  read-write
                    STATUS  mandatory
                    ::= { ipRouteEntry 7 }

            ipRouteType OBJECT-TYPE
                    SYNTAX   INTEGER {
                       other(1),      -- none of the following

                         invalid(2),   -- an invalidated route

                                       -- route to directly
                       direct(3),      -- connected (sub-)network

                                       -- route to a non-local
                       remote(4),      -- host/network/sub-network
                         }
                    ACCESS   read-write
                    STATUS   mandatory
                    ::= { ipRouteEntry 8 }

            ipRouteProto OBJECT-TYPE
                    SYNTAX   INTEGER {
                       other(1),      -- none of the following
```

```
                                -- non-protocol information
                                --    e.g., manually
                    local(2),   --    configured entries

                                -- set via a network
                    netmgmt(3), --    management protocol

                                -- obtained via ICMP,
                    icmp(4),    --    e.g., Redirect

                                -- the following are
                                -- gateway routing protocols
                    egp(5),
                    ggp(6),
                    hello(7),
                    rip(8),
                    is-is(9),
                    es-is(10),
                    ciscoIgrp(11),
                    bbnSpfIgp(12),
                    oigp(13)
                     }
            ACCESS  read-only
            STATUS  mandatory
            ::= { ipRouteEntry 9 }

    ipRouteAge OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            ::= { ipRouteEntry 10 }

    -- the ICMP group

    icmpInMsgs OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
            ::= { icmp 1 }

    icmpInErrors OBJECT-TYPE
            SYNTAX  Counter
            ACCESS  read-only
            STATUS  mandatory
            ::= { icmp 2 }

    icmpInDestUnreachs OBJECT-TYPE
            SYNTAX  Counter
```

```
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 3 }

             icmpInTimeExcds OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 4 }

             icmpInParmProbs OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 5 }

             icmpInSrcQuenchs OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 6 }

             icmpInRedirects OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 7 }

             icmpInEchos OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 8 }

             icmpInEchoReps OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 9 }

             icmpInTimestamps OBJECT-TYPE
                     SYNTAX  Counter
                     ACCESS  read-only
                     STATUS  mandatory
                     ::= { icmp 10 }

             icmpInTimestampReps OBJECT-TYPE
                     SYNTAX  Counter
```

```
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 11 }


        icmpInAddrMasks OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 12 }


        icmpInAddrMaskReps OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 13 }


        icmpOutMsgs OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 14 }


        icmpOutErrors OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 15 }


        icmpOutDestUnreachs OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 16 }


        icmpOutTimeExcds OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 17 }


        icmpOutParmProbs OBJECT-TYPE
                SYNTAX  Counter
                ACCESS  read-only
                STATUS  mandatory
                ::= { icmp 18 }


        icmpOutSrcQuenchs OBJECT-TYPE
                SYNTAX  Counter
```

```
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 19 }

                icmpOutRedirects OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 20 }

                icmpOutEchos OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 21 }

                icmpOutEchoReps OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 22 }

                icmpOutTimestamps OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 23 }

                icmpOutTimestampReps OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 24 }

                icmpOutAddrMasks OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 25 }

                icmpOutAddrMaskReps OBJECT-TYPE
                        SYNTAX  Counter
                        ACCESS  read-only
                        STATUS  mandatory
                        ::= { icmp 26 }

                -- the TCP group
```

```
            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 [11]
                        }
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 1 }

            tcpRtoMin OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 2 }

            tcpRtoMax OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 3 }

            tcpMaxConn OBJECT-TYPE
                    SYNTAX  INTEGER
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 4 }

            tcpActiveOpens OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 5 }

            tcpPassiveOpens OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 6 }

            tcpAttemptFails OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 7 }

            tcpEstabResets OBJECT-TYPE
```

```
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 8 }

            tcpCurrEstab OBJECT-TYPE
                    SYNTAX  Gauge
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 9 }

            tcpInSegs OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 10 }

            tcpOutSegs OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 11 }

            tcpRetransSegs OBJECT-TYPE
                    SYNTAX  Counter
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 12 }

            -- the TCP connections table

            tcpConnTable OBJECT-TYPE
                    SYNTAX  SEQUENCE OF TcpConnEntry
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { tcp 13 }

            tcpConnEntry OBJECT-TYPE
                    SYNTAX  TcpConnEntry
                    ACCESS  read-only
                    STATUS  mandatory
                    ::= { 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)
                        }
                ACCESS  read-only
                STATUS  mandatory
                ::= { tcpConnEntry 1 }

        tcpConnLocalAddress OBJECT-TYPE
                SYNTAX  IpAddress
                ACCESS  read-only
                STATUS  mandatory
                ::= { tcpConnEntry 2 }

        tcpConnLocalPort OBJECT-TYPE
                SYNTAX  INTEGER (0..65535)
                ACCESS  read-only
                STATUS  mandatory
                ::= { tcpConnEntry 3 }

        tcpConnRemAddress OBJECT-TYPE
                SYNTAX  IpAddress
                ACCESS  read-only
                STATUS  mandatory
                ::= { tcpConnEntry 4 }

        tcpConnRemPort OBJECT-TYPE
                SYNTAX  INTEGER (0..65535)
                ACCESS  read-only
                STATUS  mandatory
```

```
                          ::= { tcpConnEntry 5 }

              -- the UDP group

              udpInDatagrams OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { udp 1 }

              udpNoPorts OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { udp 2 }

              udpInErrors OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { udp 3 }

              udpOutDatagrams OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { udp 4 }

              -- the EGP group

              egpInMsgs OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { egp 1 }

              egpInErrors OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { egp 2 }

              egpOutMsgs OBJECT-TYPE
                      SYNTAX  Counter
                      ACCESS  read-only
                      STATUS  mandatory
                      ::= { egp 3 }
```

```
egpOutErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 4 }

-- the EGP Neighbor table

egpNeighTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF EgpNeighEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { egp 5 }

egpNeighEntry OBJECT-TYPE
        SYNTAX  EgpNeighEntry
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighTable 1 }

EgpNeighEntry ::= SEQUENCE {
    egpNeighState
        INTEGER,
    egpNeighAddr
        IpAddress
}

egpNeighState OBJECT-TYPE
        SYNTAX  INTEGER {
                    idle(1),
                    acquisition(2),
                    down(3),
                    up(4),
                    cease(5)
                }
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighEntry 1 }

egpNeighAddr OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::= { egpNeighEntry 2 }

END
```

7. Acknowledgements

8. References

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

    [2]   Information processing systems - Open Systems Interconnection,
          "Management Information Services Definition", International
          Organization for Standardization, Draft Proposal 9595/2,
          December 1987.

    [3]   Information processing systems - Open Systems Interconnection,
          "Management Information Protocol Specification", International
          Organization for Standardization, Draft Proposal 9596/2,
          December 1987.

    [4]   Rose M., and K. McCloghrie, "Structure and Identification of
          Management Information for TCP/IP-based internets", RFC 1065,
          TWG, August 1988.

    [5]   Partridge C., and G. Trewitt, "The High-Level Entity Management
          System (HEMS)", RFCs 1021-1024, BBN and Stanford, October 1987.

    [6]   Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple
          Network Management Protocol", RFC 1067, University of Tennessee
          At Knoxville, NYSERNet, Rensselaer Polytechnic, Proteon, August
          1988.

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

    [8]   LaBarre, L., "Transport Layer Management Information: TCP",
          Internet Engineering Task Force working note in preparation.
          Network Information Center, SRI International, Menlo Park,
          California, (unpublished).

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

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

    [11] Jacobson, V., "Congestion Avoidance and Control", SIGCOMM, 1988,

Stanford, California.