Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)

<draft-ietf-hubmib-mau-mib-00.txt>

27 November 1995

Dan Romascanu
LANNET Data Communications, Ltd.
dan@lannet.com

Kathryn de Graaf
3Com Corporation
kdegraaf@chipcom.com

Status of this Memo

or munnari.oz.au (Pacific Rim).

Abstract

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in the Internets community. In particular, it defines objects for managing 10 and 100 Mb/second Medium Attachment Units (MAUs) based on IEEE Std 802.3 Section 30, "10 & 100 Mb/s Management," October 26, 1995.

This memo does not specify a standard for the Internet community.

1. The SNMPv2 Network Management Framework

The SNMPv2 Network Management Framework consists of four major components. They are:

o RFC 1442 which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management.

o STD 17, RFC 1213 defines MIB-II, the core set of managed objects for the Internet suite of protocols.

o RFC 1445 which defines the administrative and other architectural aspects of the framework.

o RFC 1448 which defines the protocol used for network access to managed objects.

The Framework permits new objects to be defined for the purpose of experimentation and evaluation.

1.1. Object Definitions

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) defined in the SMI. In particular, each object

type is named by an OBJECT IDENTIFIER, an administratively assigned name. 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 descriptor, to refer to the object type.

2. Overview

Instances of these object types represent attributes of an IEEE 802.3 MAU. Several types of MAUs are defined in the IEEE **802.3 CSMA/CD standard [1] and [2].** These MAUs may be connected to IEEE 802.3 repeaters or to 802.3 (Ethernet-like) interfaces.

The definitions presented here are based on Section 30.5, "Layer Management for 10 & 100 Mb/s Medium Attachment Units (MAUs)", and Annex 30A, "GDMO Specifications for 802.3 managed objects" of IEEE Std 802.3u-1995. That specification includes definitions for both 10Mb/s and 100Mb/s devices, and is essentially a superset of the 10Mb/s definitions given by IEEE **802.3 Section 20.** This specification is intended to serve the same purpose: to provide for management of both 10Mb/s and 100Mb/s MAUs.

MAUs are components that are often located inside a larger system, and are not always externally visible to a network administrator. The external connection of a MAU is generally a jack to which a network cable can be attached. The internal connection of a MAU can be, as explained above, a repeater port or an interface. In some systems, this internal connectivity may be configurable. Additionally, the introduction of auto-negotiation functionality in the IEEE **802.3 specification allows for a connection which can, to a** certain degree, configure or reconfigure itself during operation. This MIB includes objects for jack configuration and status that are intended to allow the network management software to model and report the connectivity between the external jacks, the MAUs, and the other components (repeater ports or interfaces) within such a system, including both administrative configuration and auto- negotiation. This model assumes a one-to-one relationship between jacks and MAUs.

3. Definitions

```
MAU-MIB DEFINITIONS ::= BEGIN

IMPORTS
    experimental, Counter32, Integer32, Gauge32, Counter64,
    OBJECT-TYPE, MODULE-IDENTITY, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    TimeStamp, DisplayString, MacAddress, TEXTUAL-CONVENTION,
    RowStatus
        FROM SNMPv2-TC
    OBJECT-GROUP, MODULE-COMPLIANCE, NOTIFICATION-GROUP,
        FROM SNMPv2-CONF
    mib-2
        FROM RFC1213-MIB;

mauMod MODULE-IDENTITY
    LAST-UPDATED "9511270000Z"
    ORGANIZATION "IETF HUB MIB Working Group"
    CONTACT-INFO
        "WG E-mail: hubmib@baynetworks.com

             Chair: Dan Romascanu
            Postal: LANNET Data Communications, Ltd.
                    Atidim Technology Park, Bldg. 3
                    Tel Aviv 61131, Israel
               Tel: 972-3-6458414, 6458458
               Fax: 972-3-6487146
            E-mail: dan@lannet.com

            Editor: Kathryn de Graaf
            Postal: 3Com Corporation
                    118 Turnpike Rd.
                    Southborough, MA  01772
                    USA
               Tel: (508)229-1627
               Fax: (508)490-5882
            E-mail: kdegraaf@chipcom.com"
    DESCRIPTION
            "Management information for 802.3 MAUs.

            The following references are used throughout this
            MIB module:
```

```
            [IEEE 802.3 Std]
               refers to IEEE 802.3/ISO 8802-3 Information
               processing systems - Local area networks -
               Part 3: Carrier sense multiple access with
               collision detection (CSMA/CD) access method
               and physical layer specifications (1993),
               and to IEEE Std 802.3u-1995, Supplement to
               IEEE Std 802.3, clauses 22 through 29.

            [IEEE 802.3 Mgt]
               refers to IEEE 802.3u-1995, - 10 Mb/s &
               100 Mb/s Management, Section 30 -
               Supplement to IEEE Std 802.3."
    ::= { snmpDot3MauMgt 6 }


snmpDot3MauMgt OBJECT IDENTIFIER ::= { experimental x }

-- the following subtrees are deprecated

dot3RpMauBasicGroup         OBJECT IDENTIFIER ::= { snmpDot3MauMgt
1 }
dot3IfMauBasicGroup         OBJECT IDENTIFIER ::= { snmpDot3MauMgt
2 }
dot3BroadMauBasicGroup      OBJECT IDENTIFIER ::= { snmpDot3MauMgt
3 }



dot3MauBasicGroup      OBJECT IDENTIFIER ::= { snmpDot3MauMgt 7 }

-- object identifiers for MAU types
--  (see rpMauType and ifMauType for usage)

dot3MauType
    OBJECT IDENTIFIER ::= { snmpDot3MauMgt 4 }
dot3MauTypeAUI         -- no internal MAU, view from AUI
    OBJECT IDENTIFIER ::= { dot3MauType 1 }
dot3MauType10Base5     -- thick coax MAU (per 802.3 section 8)
    OBJECT IDENTIFIER ::= { dot3MauType 2 }
dot3MauTypeFoirl       -- FOIRL MAU (per 802.3 section 9.9)
    OBJECT IDENTIFIER ::= { dot3MauType 3 }
dot3MauType10Base2     -- thin coax MAU (per 802.3 section 10)
    OBJECT IDENTIFIER ::= { dot3MauType 4 }
dot3MauType10BaseT     -- UTP MAU (per 802.3 section 14)
    OBJECT IDENTIFIER ::= { dot3MauType 5 }
dot3MauType10BaseFP    -- passive fiber MAU (per 802.3 section 16)
    OBJECT IDENTIFIER ::= { dot3MauType 6 }
```

```
dot3MauType10BaseFB   -- sync fiber MAU (per 802.3 section 17)
    OBJECT IDENTIFIER ::= { dot3MauType 7 }
dot3MauType10BaseFL   -- async fiber MAU (per 802.3 section 18)
    OBJECT IDENTIFIER ::= { dot3MauType 8 }
dot3MauType10Broad36  -- broadband DTE MAU (per 802.3 section 11)
    -- note that 10BROAD36 MAUs can be attached to interfaces but
    -- not to repeaters
    OBJECT IDENTIFIER ::= { dot3MauType 9 }

-- new for 100 MB/s:

dot3MauType100BaseT4   -- 4 pair categ. 3 UTP (per 802.3 section 23)
    OBJECT IDENTIFIER ::= { dot3MauType 10 }
dot3MauType100BaseTX   -- 2 pair categ. 5 UTP (per 802.3 section 25)
    OBJECT IDENTIFIER ::= { dot3MauType 11 }
dot3MauType100BaseFX   -- X fiber over PMT (per 802.3 section 26)
    OBJECT IDENTIFIER ::= { dot3MauType 12 }


--
-- The Basic MAU Table
--

mauTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF MauEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "Table of descriptive and status information about
            the managed MAU(s) in this system."
    ::= { mauBasicGroup 1 }

mauEntry OBJECT-TYPE
    SYNTAX     MauEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "An entry in the table, containing information
            about a single MAU."
    INDEX      { mauGroupIndex, mauIndex }
    ::= { mauTable 1 }

MauEntry ::=
    SEQUENCE {
        mauGroupIndex
```

```
                Integer32,
            mauIndex
                Integer32,
            mauType
                OBJECT IDENTIFIER,
            mauTypeList
                Integer32,
            mauStatus
                INTEGER,
            mauMediaAvail
                INTEGER,
            mauMediaAvailStateExits
                Counter32,
            mauJabberState
                INTEGER,
            mauJabberingStateEnters
                Counter32,
            mauFalseCarriers
                Counter32
        }

    mauGroupIndex OBJECT-TYPE
        SYNTAX     Integer32 (1..2147483647)
        MAX-ACCESS read-only
        STATUS     mandatory
        DESCRIPTION
                "This variable uniquely identifies the group
                containing the MAU described by this entry.

                Note:  In practice, a group will generally be a
                field-replaceable unit (i.e., module, card, or
                board) that can fit in the physical system
                enclosure, and the group number will correspond to
                a number marked on the physical enclosure.

                For MAUs attached to repeaters, the group denoted
                by a particular value of this object is the same
                as the group denoted by the same value of
                rptrGroupIndex."
        ::= { mauEntry 1 }

    mauIndex OBJECT-TYPE
        SYNTAX     Integer32 (1..2147483647)
        MAX-ACCESS read-only
        STATUS     mandatory
```

```
    DESCRIPTION
            "This variable uniquely identifies the MAU within
            group mauGroupIndex that is described by this
            entry."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.1, aMAUID."
    ::= { mauEntry 2 }

mauType OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This object identifies the 10 or 100 Mb/s
            baseband MAU type.  An initial set of MAU types
            are defined above.  The assignment of OBJECT
            IDENTIFIERs to new types of MAUs is managed by the
            IANA.  If the MAU type is unknown, the object
            identifier

            unknownMauType OBJECT IDENTIFIER ::= { 0 0 }

            is returned.  Note that unknownMauType is a
            syntactically valid object identifier, and any
            conformant implementation of ASN.1 and the BER
            must be able to generate and recognize this
            value."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.2, aMAUType."
    ::= { mauEntry 3 }

mauTypeList OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A value that uniquely identifies the set of
            possible IEEE 802.3 types that the MAU could be.
            The value is a sum which initially takes the value
            zero.  Then, for each type capability of this MAU,
            2 raised to the power noted below is added to the
            sum. For example, a MAU which has the capability
            to be only 10BASE-T would have a value of 512
            (2**9).  In contrast, a MAU which supports both
            10Base-T and 100BASE-TX would have a value of 1536
```

```
            ((2**9) + (2**10)).

            The powers of 2 assigned to the capabilities are
            these:

            Power  Capability
              1      AUI
              2      10BASE-5
              3      FOIRL
              4      10BASE-2
              5      10BASE-T
              6      10BASE-FP
              7      10BASE-FB
              8      10BASE-FL
              9      10BROAD36
             10      100BASE-T4
             11      100BASE-TX
             12      100BASE-FX

            If auto-negotiation is present on the jack to
            which this MAU is attached, this attribute will
            map to the jackAutoNegCapability."
    ::= { mauEntry 4 }

mauStatus OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   unknown(2),
                   operational(3),
                   standby(4),
                   shutdown(5),
                   reset(6)
               }
    MAX-ACCESS read-write
    STATUS     mandatory
    DESCRIPTION
            "The current state of the MAU.  This object may be
            implemented as a read-only object by those agents
            and MAUs that do not implement software control of
            the MAU state.  Some agents may not support
            setting the value of this object to some of the
            enumerated values.

            The value other(1) is returned if the MAU is in a
            state other than one of the states 2 through 6.
```

```
            The value unknown(2) is returned when the MAU's
            true state is unknown; for example, when it is
            being initialized.

            A MAU in the operational(3) state is fully
            functional, operates, and passes signals to its
            attached DTE or repeater port in accordance to its
            specification.

            A MAU in standby(4) state forces DI and CI to idle
            and the media transmitter to idle or fault, if
            supported.  Standby(4) mode only applies to link
            type MAUs.  The state of mauMediaAvail is
            unaffected.

            A MAU in shutdown(5) state assumes the same
            condition on DI, CI, and the media transmitter as
            though it were powered down or not connected.  The
            MAU may return other(1) value for the
            mauJabberState and mauMediaAvail objects when it
            is in this state.  For an AUI, this state will
            remove power from the AUI.

            Setting this variable to the value reset(6) resets
            the MAU in the same manner as a power-off, power-
            on cycle of at least one-half second would.  The
            agent is not required to return the value reset
            (6).

            Setting this variable to the value operational(3),
            standby(4), or shutdown(5) causes the MAU to
            assume the respective state except that setting a
            mixing-type MAU or an AUI to standby(4) will cause
            the MAU to enter the shutdown state."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.7, aMAUAdminState,
            30.5.1.2.2, acMAUAdminControl, and 30.5.1.2.1,
            acRESETMAU."
    ::= { mauEntry 5 }

mauMediaAvail OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   unknown(2),
                   available(3),
```

```
                       notAvailable(4),
                       remoteFault(5),
                       invalidSignal(6),
                       remoteJabber(7),
                       remoteLinkLoss(8),
                       remoteTest(9)
                   }
        MAX-ACCESS read-only
        STATUS     mandatory
        DESCRIPTION
                "If the MAU is a link or fiber type (FOIRL,
                10BASE-T, 10BASE-F) then this is equivalent to the
                link test fail state/low light function.  For an
                AUI or a coax (including broadband) MAU this
                indicates whether or not loopback is detected on
                the DI circuit.  The value of this attribute
                persists between packets for MAU types AUI,
                10BASE5, 10BASE2, 10BROAD36, and 10BASE-FP.

                The value other(1) is returned if the mediaAvail
                state is not one of 2 through 6.

                The value unknown(2) is returned when the MAU's
                true state is unknown; for example, when it is
                being initialized.  At power-up or following a
                reset, the value of this attribute will be unknown
                for AUI, coax, and 10BASE-FP MAUs.  For these MAUs
                loopback will be tested on each transmission
                during which no collision is detected.  If DI is
                receiving input when DO returns to IDL after a
                transmission and there has been no collision
                during the transmission then loopback will be
                detected.  The value of this attribute will only
                change during non-collided transmissions for AUI,
                coax, and 10BASE-FP MAUs.

                For 100BASE-T4, 100BASE-TX and 100BASE-FX the
                enumerations match the states within the
                respective link integrity state diagrams, fig 23-
                12 and 24-15 of sections 23 and 24 of [2].  Any
                MAU which implements management of Auto-
                Negotiation will map remote fault indication to
                remote fault.

                The value available(3) indicates that the link,
```

```
                light, or loopback is normal.  The value
                notAvailable(4) indicates link loss, low light, or
                no loopback.

                The value remoteFault(5) indicates that a fault
                has been detected at the remote end of the link.
                This value applies to 10BASE-FB, 100BASE-T4 Far
                End Fault Indication and non-specified remote
                faults from a system running Auto-Negotiation.
                The values remoteJabber(7), remoteLinkLoss(8), and
                remoteTest(9) should be used instead of
                remoteFault(5) where the reason for remote fault
                is identified in the remote signaling protocol.

                The value invalidSignal(6) indicates that an
                invalid signal has been received from the other
                end of the link.  Both remoteFault(5) and
                invalidSignal(6) apply only to MAUs of type
                10BASE-FB.

                Where an IEEE Draft Std 802.3u/D4 clause 22 MII is
                present, a logic one in the remote fault bit
                (reference section 22.2.4.2.8 of that document)
                maps to the value remoteFault(5), and a logic zero
                in the link status bit (reference section
                22.2.4.2.10 of that document) maps to the value
                notAvailable(4).  The value notAvailable(4) takes
                precedence over the value remoteFault(5)."
    REFERENCE
                "[IEEE 802.3 Mgt], 30.5.1.1.4, aMediaAvailable."
    ::= { mauEntry 6 }

mauMediaAvailStateExits OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
                "A count of the number of times that mauMediaAvail
                for this MAU instance leaves the state
                available(3)."
    REFERENCE
                "[IEEE 802.3 Mgt], 30.5.1.1.5, aLoseMediaCounter."
    ::= { mauEntry 7 }

mauJabberState OBJECT-TYPE
```

```
    SYNTAX     INTEGER {
                   other(1),
                   unknown(2),
                   noJabber(3),
                   jabbering(4)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "The value other(1) is returned if the jabber
            state is not 2, 3, or 4.  The agent must always
            return other(1) for MAU type dot3MauTypeAUI.

            The value unknown(2) is returned when the MAU's
            true state is unknown; for example, when it is
            being initialized.

            If the MAU is not jabbering the agent returns
            noJabber(3).  This is the 'normal' state.

            If the MAU is in jabber state the agent returns
            the jabbering(4) value."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.6,
            aJabber.jabberFlag."
    ::= { mauEntry 8 }

mauJabberingStateEnters OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A count of the number of times that
            mauJabberState for this MAU instance enters the
            state jabbering(4).  For MAUs of type
            dot3MauTypeAUI, dot3MauType100BaseT4,
            dot3MauType100BaseTX, and dot3MauType100BaseFX,
            this counter will always indicate zero."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.6,
            aJabber.jabberCounter."
    ::= { mauEntry 9 }

mauFalseCarriers OBJECT-TYPE
    SYNTAX     Counter32
```

```
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A count of the number of false carrier events
            during IDLE in 100BASE-X links.  This counter does
            not increment at the symbol rate.  It can
            increment after a valid carrier completion at a
            maximum rate of once per 100 ms until the next
            carrier event.

            This counter increments only for MAUs of type
            dot3MauType100BaseT4, dot3MauType100BaseTX, and
            dot3MauType100BaseFX.  For all other MAU types,
            this counter will always indicate zero.

            The approximate minimum time for rollover of this
            counter is 7.4 hours."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.1.10, aFalseCarriers."
    ::= { mauEntry 10 }


--
-- Jack Tables
--
-- The jack object types defined below are intended to provide
-- management information for external jacks on a system.  They
-- describe the jack from the outside looking in, such that for
-- a particular connector at a particular location on the outside
-- of the box, these objects can be used to determine to which
-- internal entity it is mapped, and to which internal entity it
-- could potentially be mapped as a result of the operation of the
-- auto-negotiation function defined by the IEEE 802.3 management
-- standard.
--
-- The IEEE 802.3 auto-negotiation function is defined such that
-- it can be expanded to work for technologies other than those
-- defined by 802.3.  In addition, although the purpose of
-- auto-negotiation is to negotiate the use of a particular
-- (signalling?) technology across a particular link between
-- network components, the outcome of that technology choice
-- can have run-time implications for systems containing a mix
-- of networking features.
--
-- For instance, a system containing both 10 and 100
```

```
-- Mb/s repeater capability on a particular link will,
-- of necessity, require the link to potentially connect to
-- two different repeaters: one 10 Mb/s repeater and one 100
-- Mb/s repeater (there being no such device as a 10/100
-- repeater).  Only one of these connections will be active
-- at run time, depending on the outcome of the auto-negotiation
-- on that link, and therefore the auto-negotiation itself can
-- have the effect of determining the actual connectivity of
-- the link.
--
-- These tables are intended to apply firstly to jacks connected to
-- IEEE 802.3 repeaters, and additionally to any other management
-- domain for which they may be useful.  Certain object types here
-- have been cross-referenced into the 802.3 repeater MIB (which
-- draft is currently under discussion by this working group) and
-- to the interfaces table of MIB-II.  For example, there is
-- some overlap between "jackInternalConnection" and "rptrInfoId"
-- from the repeater MIB draft.  Please refer to the object
-- definitions below for specific details.


-- The jackTable applies to systems which have one or
-- more external jacks (connectors).

jackTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF JackEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "Configuration objects for the external jacks on
            the system."
    ::= { jackBasicGroup 1 }

jackEntry OBJECT-TYPE
    SYNTAX     JackEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "An entry in the table, containing configuration
            information for a a particular jack."
    INDEX    { jackGroupIndex,
               jackjackIndex }
    ::= { jackTable 1 }

JackEntry ::=
```

```
    SEQUENCE {
        jackGroupIndex
            Integer32,
        jackIndex
            Integer32,
        jackType
            INTEGER,
        jackInternalConnection
            OBJECT IDENTIFIER,
        jackConnectionLastChange
            TimeStamp
    }


jackGroupIndex OBJECT-TYPE
    SYNTAX     Integer32 (1..2147483647)
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable uniquely identifies the group
            within the system containing the jack described by
            this entry.

            Note:  In practice, a group will generally be a
            field-replaceable unit (i.e., module, card, or
            board) that can fit in the physical system
            enclosure, and the group number will correspond to
            a number marked on the physical enclosure.

            For jacks attached to repeaters, the group denoted
            by a particular value of this object is the same
            as the group denoted by the same value of
            rptrGroupIndex."
    ::= { jackEntry 1 }

jackIndex OBJECT-TYPE
    SYNTAX     Integer32 (1..2147483647)
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable uniquely identifies the jack
            described by this entry among within other jacks
            within the group denoted by jackGroupIndex.

            Note that if the jack is connected to a repeater,
```

```
            the value of this object is the same as the value
            of rptrPortIndex for the associated port in the
            same group (i.e. jackIndex == rptrPortIndex, and
            jackGroupIndex == rptrPortGroupIndex).

            Jacks may also be connected to other entities,
            including logical interfaces within the system, in
            which case the numbering of the entity may not
            match the numbering of the jack.  In all cases,
            the next-level entity to which this jack is
            connected is specified by the
            jackInternalConnection object for this entry."
    ::= { jackEntry 2 }

jackType OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   rj45(2)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "The jack connector type, as it appears on the
            outside of the system."
    ::= { jackEntry 3 }

jackInternalConnection OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable identifies the instance of an
            internal entity to which the jack is connected.
            For a jack which can be administratively
            configured, or a jack on which auto-negotiation is
            supported, the value of this object may change
            between system resets.

            If the jack is connected to a repeater, the value
            of this object will be rptrInfoId.r, where r
            equals the value of rptrPortRptrId for the port
            with which this jack is associated (see also the
            above description of jackIndex).

            If the jack is connected to an interface, the
```

```
            value of this object will be ifIndex.i, where i is
            the instance of the interface.

            For other types of internal entities, the value of
            this object must be an instance identifier which
            uniquely identifies the entity type and the
            instance of that type within the managed system."
    ::= { jackEntry 4 }

jackConnectionLastChange OBJECT-TYPE
    SYNTAX     TimeStamp
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
         "The value of sysUpTime when any of the following
         conditions occurred:
           1) the agent cold- or warm-started;
           2) this instance of jack was created
              (such as when a device or module was
              added to the system);
           3) a change occurred in the value of
              jackInternalConnection for this entry."
    ::= { jackEntry 5 }


-- The jackAutoNegTable applies to systems in which
-- auto-negotiation is supported on one or more jacks.

jackAutoNegTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF JackAutoNegEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "Configuration and status objects for the auto-
            negotiation function of external jacks on the
            system."
    ::= { jackAutoNegGroup 1 }

jackAutoNegEntry OBJECT-TYPE
    SYNTAX     JackAutoNegEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "An entry in the table, containing configuration
            and status information for the auto-negotiation
```

```
            function of a particular jack."
    AUGMENTS  { jackEntry }
    ::= { jackAutoNegTable 1 }

JackAutoNegEntry ::=
    SEQUENCE {
        jackAutoNegAdminStatus
            INTEGER,
        jackAutoNegRemoteSignaling
            INTEGER,
        jackAutoNegPotentialConnectSet
            Integer32,
        jackAutoNegConfig
            INTEGER,
        jackAutoNegCapability
            Integer32,
        jackAutoNegCapAdvertised
            Integer32,
        jackAutoNegCapReceived
            Integer32,
        jackAutoNegTechnologyInUse
            INTEGER,
        jackAutoNegRestart
            INTEGER

    }


jackAutoNegAdminStatus OBJECT-TYPE
    SYNTAX     INTEGER {
                   enabled(1),
                   disabled(2)
               }
    MAX-ACCESS read-write
    STATUS     mandatory
    DESCRIPTION
            "Setting this object to enabled(1) will cause the
            interface which has the auto-negotiation signaling
            ability to be enabled. If disabled then the
            interface will act as it would if it had no auto-
            negotiation signaling.

            Under these conditions, a jack connected to an
            IEEE 802.3 MAU will immediately be forced to the
            states indicated by a write to the object
```

```
            rpMauType or ifMauType.  [Ed.--This doesn't allow
            for half vs.  full duplex.]"
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.2, aAutoNegAdminState
            and 30.6.1.2.2, acAutoNegAdminControl."
    ::= { jackAutoNegEntry 1 }

jackAutoNegRemoteSignaling OBJECT-TYPE
    SYNTAX     INTEGER {
                   detected(1),
                   notdetected(2)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A value indicating whether the remote end of the
            link is using auto-negotiation signaling. It takes
            the value detected(1) if and only if, during the
            previous link negotiation, FLP Bursts were
            received."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.3,
            aAutoNegRemoteSignaling."
    ::= { jackAutoNegEntry 2 }

jackAutoNegPotentialConnectSet OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable identifies the set of internal
            entities to which this jack can potentially
            connect using as a result of auto-negotiation.

            The set of potential connections can include, at
            most, one entity supporting each of the
            technologies for which this jack has auto-
            negotiation capability.  For example, if the
            jackAutoNegCapability for this entry includes
            10Base-T, then one and only one 10Base-T entity
            may be included in the jack's potential connection
            set.

            The members of the set are defined in the
            jackAutoNegConnectSetTable; the value of this
```

```
            object is the first index
            (jackAutoNegConnectSetIndex) into that table.
            Each entry in that table whose first index has the
            same value as this object is a member of the set
            of potential connections for this jack.

            Note that this object is read-only, and therefore
            does not allow for administrative control of the
            jack connection:  the method of such control, if
            available, is implementation-specific.  [?? allow
            read-write implementations of this ??]

            (See also the definitions for the
            jackAutoNegConnectSetTable.)"
    ::= { jackAutoNegEntry 3 }

jackAutoNegConfig OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   configuring(2),
                   complete(3),
                   disabled(4),
                   parallelDetectFail(5)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A value indicating the current status of the
            auto-negotiation process.  The enumeration
            parallelDetectFail(5) maps to a failure in
            parallel detection as defined in 28.2.3.1 of [IEEE
            802.3 Std]."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.4,
            aAutoNegAutoConfig."
    ::= { jackAutoNegEntry 4 }

jackAutoNegCapability OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A value that uniquely identifies the set of
            capabilities of the local auto-negotiation entity.
            The value is a sum which initially takes the value
```

```
            zero.  Then, for each capability of this
            interface, 2 raised to the power noted below is
            added to the sum. For example, an interface which
            has the capability to support only 100Base-TX
            would have a value of 512 (2**9).  In contrast, an
            interface which supports both 100Base-TX and
            100Base-TX Full Duplex would have a value of 1536
            ((2**9) + (2**10)).

            The powers of 2 assigned to the capabilities are
            these:

            Power  Capability
              1      other
              2      reserved
              3      10BASE-T
              4      10BASE-T Full Duplex
              5      10BASE-FL
              6      10BASE-FL Full Duplex
              7      10BASE-FB
              8      10BASE-FB Full Duplex
              9      100BASE-TX
             10      100BASE-TX Full Duplex
             11      100BASE-FX
             12      100BASE-FX Full Duplex
             13      100BASE-T4

            For jacks connected to IEEE 802.3 MAUs, the half-
            and full-duplex value pairs each map to a single
            MAU type.  For example, 10BASE-T and 10BASE-T Full
            Duplex each use a MAU type of dot3MauType10BaseT.

            Note that interfaces that support this MIB may
            have capabilities that extend beyond the scope of
            this MIB."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.5,
            aAutoNegLocalTechnologyAbility."
    ::= { jackAutoNegEntry 5 }

jackAutoNegCapAdvertised OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-write
    STATUS     mandatory
    DESCRIPTION
```

```
            "A value that uniquely identifies the set of
            capabilities advertised by the local auto-
            negotiation entity. Refer to jackAutoNegCapability
            for a description of the possible values of this
            object.

            Capabilities in this object that are not available
            in jackAutoNegCapability cannot be enabled."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.6,
            aAutoNegAdvertisedTechnologyAbility."
    ::= { jackAutoNegEntry 6 }

jackAutoNegCapReceived OBJECT-TYPE
    SYNTAX     Integer32
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "A value that uniquely identifies the set of
            capabilities received from the remote auto-
            negotiation entity. Refer to jackAutoNegCapability
            for a description of the possible values of this
            object.

            Note that interfaces that support this MIB may be
            attached to remote auto-negotiation entities which
            have capabilities beyond the scope of this MIB."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.1.7,
            aAutoNegReceivedTechnologyAbility."
    ::= { jackAutoNegEntry 7 }

jackAutoNegTechnologyInUse OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   reserved(2),
                   10BASE-T(3),
                   10BASE-T-FD(4),
                   10BASE-FL(5),
                   10BASE-FL-FD(6),
                   10BASE-FB(7),
                   10BASE-FB-FD(8),
                   100BASE-TX(9),
                   100BASE-TX-FD(10),
                   100BASE-FX(11),
```

```
                   100BASE-FX-FD(12),
                   100BASE-T4(13)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "The value of this object identifies the
            technology currently in use on the link to which
            this jack is attached.  This value may be a result
            of auto- negotiation on the link.  If auto-
            negotiation is disabled and the jack is connected
            to an IEEE 802.3 MAU, this object will change to
            reflect the result of a write to the object
            rpMauType or ifMauType."
    ::= { jackAutoNegEntry 8 }


jackAutoNegRestart OBJECT-TYPE
    SYNTAX     INTEGER {
                   restart(1),
                   norestart(2)
               }
    MAX-ACCESS read-write
    STATUS     mandatory
    DESCRIPTION
            "If the value of this object is set to restart(1)
            then this will force auto-negotiation to begin
            link renegotiation. If auto-negotiation signaling
            is disabled, a write to this object has no effect.

            Setting the value of this object to norestart(2)
            has no effect."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.6.1.2.1,
            acAutoNegRestartAutoConfig."
    ::= { jackAutoNegEntry 9 }


-- The jackAutoNegConnectSetTable applies to systems in which
-- auto-negotiation is supported on one or more jacks.

jackAutoNegConnectSetTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF JackAutoNegConnectSetEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
```

```
    DESCRIPTION
            "A table describing sets of entities within the
            system.  Each jack in the system, for which auto-
            negotiation is supported, is associated with one
            of these sets.  Each such jack may connect to any
            ONE of the entities belonging to the set denoted
            by its instance of jackAutoNegPotentialConnectSet,
            the choice to be determined by the jack's auto-
            negotiation function (if enabled).

            Any single set may contain no more than one entity
            representing a particular technology to which the
            jack can auto-negotiate.  For example, a set may
            include one and only one 10BASE-T entity (e.g. a
            repeater), and one and only one 100BASE-TX full
            duplex entity (e.g. an interface). The list of
            potential technologies is denoted by the
            capabilities enumerated in the
            jackAutoNegCapability OBJECT-TYPE description; it
            follows that the maximum number of entities in any
            set is the same as the number of different
            capabilities listed in that description.

            It is expected that in a system which supports
            administrative configuration of connections, the
            administrator will configure the connection set
            for a jack so that all of the entity members are
            part of a single network, in order that the auto-
            negotiation function will not be able to
            disconnect and reconnect a link between various
            networks supported within the system.  That is,
            the jackAutoNegPotentialConnectSet is used to
            determine the network connection (similarly to the
            way a repeater port's repeater id can be used),
            but WITHIN that set, the particular entity chosen
            is determined via auto-negotiation during
            operation."
    ::= { jackAutoNegGroup 2 }

jackAutoNegConnectSetEntry OBJECT-TYPE
    SYNTAX     JackAutoNegConnectSetEntry
    MAX-ACCESS not-accessible
    STATUS     mandatory
    DESCRIPTION
            "An entry defining one entity of a set of entities
```

```
            to which a system jack may potentially connect,
            the determination to be made by the jack's auto-
            negotiation function."
    INDEX  { jackAutoNegConnectSet,
             jackAutoNegConnectEntityType }
    ::= { jackAutoNegConnectSetTable 1 }

JackAutoNegConnectSetEntry ::=
    SEQUENCE {
        jackAutoNegConnectSet
            Integer32,
        jackAutoNegConnectEntityType
            INTEGER,
        jackAutoNegConnectEntity
            OBJECT IDENTIFIER
    }

jackAutoNegConnectSet OBJECT-TYPE
    SYNTAX     Integer32 (1..2147483647)
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable uniquely identifies a set of entity
            instances among the other sets represented within
            this table."
    ::= { jackAutoNegConnectSetEntry 1 }

jackAutoNegConnectEntityType OBJECT-TYPE
    SYNTAX     INTEGER {
                   other(1),
                   reserved(2),
                   10BASE-T(3),
                   10BASE-T-FD(4),
                   10BASE-FL(5),
                   10BASE-FL-FD(6),
                   10BASE-FB(7),
                   10BASE-FB-FD(8),
                   100BASE-TX(9),
                   100BASE-TX-FD(10),
                   100BASE-FX(11),
                   100BASE-FX-FD(12),
                   100BASE-T4(13)
               }
    MAX-ACCESS read-only
    STATUS     mandatory
```

```
    DESCRIPTION
            "This variable identifies the type of internal
            entity about which this entry contains
            information.  Each set of potential connections
            may contain no more than one entry of any
            particular technology type."
    ::= { jackAutoNegConnectSetEntry 2 }

jackAutoNegConnectEntity OBJECT-TYPE
    SYNTAX     OBJECT IDENTIFIER
    MAX-ACCESS read-only
    STATUS     mandatory
    DESCRIPTION
            "This variable identifies the instance of an
            internal entity to which a jack may potentially
            connect.  802.3 repeater and interfaces are
            examples of such entities.

            Note that if the jack is connected to a repeater,
            the value of this object is the same as the value
            of rptrPortIndex for the associated port in the
            same group (i.e. jackIndex == rptrPortIndex, and
            jackGroupIndex == rptrPortGroupIndex).

            Jacks may also be connected to other types of
            entities, including logical interfaces within the
            system, in which case the numbering of the entity
            may not match the numbering of the jack.  In all
            cases, the next-level entity to which this jack is
            connected is specified by the
            jackInternalConnection object for this entry."
    ::= { jackAutoNegConnectSetEntry 3 }



-- Notifications for use by 802.3 MAUs

mauJabberTrap NOTIFICATION-TYPE
    OBJECTS    { mauJabberState }
    DESCRIPTION
            "This trap is sent whenever a managed repeater MAU
            enters the jabber state.

            The agent must throttle the generation of
            consecutive mauJabberTraps so that there is at
```

```
            least a five-second gap between them."
    REFERENCE
            "[IEEE 802.3 Mgt], 30.5.1.3.1, nJabber
            notification."
    ::= { snmpDot3MauMgt 0 1 }


END
```

4. References

[1] IEEE 802.3/ISO 8802-3 Information processing systems - Local area networks - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 1993.

[2] IEEE 802.3u-1995, "MAC Parameters, Physical Layer, Medium Attachment Units and Repeater for 100 Mb/s Operation, Type 100BASE-T," Sections 21 through 29, Supplement to IEEE Std 802.3, October 26, 1995.

[3] IEEE 802.3u-1995, "10 & 100 Mb/s Management," Section 30, Supplement to IEEE Std 802.3, October 26, 1995.

[4] Romascanu, D., and K. de Graaf, "Definitions of Managed Objects for IEEE 802.3 Repeater Devices", November 1995.