

Definitions of Managed Objects for IEEE 802.12 Interfaces

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Table of Contents

1. Introduction	1
2. Object Definitions	2
3. Overview	2
3.1. MAC Addresses	3
3.2. Relation to RFC 1213	3
3.3. Relation to RFC 1573	3
3.3.1. Layering Model	4
3.3.2. Virtual Circuits	4
3.3.3. ifTestTable	4
3.3.4. ifRcvAddressTable	4
3.3.5. ifPhysAddress	4
3.3.6. Specific Interface MIB Objects	5
3.4. Relation to RFC 1643 , RFC 1650 , and RFC 1748	8
3.5. Relation to RFC 1749	8
3.6. Master Mode Operation	9
3.7. Normal and High Priority Counters	9
3.8. IEEE 802.12 Training Frames	10
3.9. Mapping of IEEE 802.12 Managed Objects	12
4. Definitions	14
5. Acknowledgements	30
6. References	30
7. Security Considerations	31
8. Author's Address	31

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing network interfaces based on IEEE 802.12.

2. Object Definitions

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

3. Overview

Instances of these object types represent attributes of an interface to an IEEE 802.12 communications medium. At present, IEEE 802.12 media are identified by one value of the ifType object in the Internet-standard MIB:

```
ieee80212(55)
```

For this interface, the value of the ifSpecific variable in the MIB-II [5] has the OBJECT IDENTIFIER value:

```
dot12MIB    OBJECT IDENTIFIER ::= { transmission 45 }
```

The values for the ifType object are defined by the IANAifType textual convention. The Internet Assigned Numbers Authority (IANA) is responsible for the assignment of all Internet numbers, including new ifType values. Therefore, IANA is responsible for maintaining the definition of this textual convention. The current definition of the IANAifType textual convention is available from IANA's World Wide Web server at:

```
http://www.iana.org/iana/
```

The definitions presented here are based on the IEEE Standard 802.12-1995, [6] Clause 13 "Layer management functions and services", and Annex C "GDMO Specifications for Demand Priority Managed Objects". Implementors of these MIB objects should note that the IEEE document explicitly describes (in the form of Pascal pseudocode) when, where, and how various MAC attributes are measured. The IEEE document also describes the effects of MAC actions that may be invoked by manipulating instances of the MIB objects defined here.

To the extent that some of the attributes defined in [6] are represented by previously defined objects in the Internet-standard MIB [5] or in the Evolution of the Interfaces Group of MIB-II [7], such attributes are not redundantly represented by objects defined in this memo. Among the attributes represented by objects defined in other memos are the number of octets transmitted or received on a particular interface, the MAC address of an interface, and multicast information associated with an interface.

3.1. MAC Addresses

All representations of MAC addresses in this MIB module, and in other related MIB modules (like RFC 1573), are in "canonical" order defined by 802.1a, i.e., as if it were transmitted least significant bit first. This is true even if the interface is operating in token ring framing mode, which requires MAC addresses to be transmitted most significant bit first.

3.2. Relation to RFC 1213

This section applies only when this MIB is used in conjunction with the "old" (i.e., pre-RFC 1573) interface group.

The relationship between an IEEE 802.12 interface and an interface in the context of the Internet-standard MIB is one-to-one. As such, the value of an ifIndex object instance can be directly used to identify corresponding instances of the objects defined herein.

3.3. Relation to RFC 1573

RFC 1573, the Interface MIB Evolution, requires that any MIB which is an adjunct of the Interface MIB, clarify specific areas within the Interface MIB. These areas are intentionally left vague in RFC 1573 to avoid over constraining the MIB, thereby precluding management of certain media-types.

An agent which implements this MIB module must support the ifGeneralGroup, ifStackGroup, ifHCPacketGroup, and ifRcvAddressGroup of RFC 1573.

Section 3.3 of RFC 1573 enumerates several areas which a media-specific MIB must clarify. In addition, there are some objects in RFC 1573 for which additional clarification of how to apply them to an IEEE 802.12 interface would be helpful. Each of these areas is addressed in a following subsection. The implementor is referred to RFC 1573 in order to understand the general intent of these areas.

3.3.1. Layering Model

For the typical usage of this MIB module, there will be no sub-layers "above" or "below" the 802.12 Interface. However, this MIB module does not preclude such layering.

3.3.2. Virtual Circuits

This medium does not support virtual circuits and this area is not applicable to this MIB.

3.3.3. ifTestTable

This MIB does not define any tests for media instrumented by this MIB. Implementation of the ifTestTable is not required. An implementation may optionally implement the ifTestTable to execute vendor specific tests.

3.3.4. ifRcvAddressTable

This table contains all IEEE addresses, unicast, multicast, and broadcast, for which this interface will receive packets and forward them up to a higher layer entity for consumption. In addition, when the interface is using 802.5 framing mode, the ifRcvAddressTable will contain the functional address mask.

In the event that the interface is part of a MAC bridge, this table does not include unicast addresses which are accepted for possible forwarding out some other port. This table is explicitly not intended to provide a bridge address filtering mechanism.

3.3.5. ifPhysAddress

This object contains the IEEE 802.12 address which is placed in the source-address field of any frames that originate at this interface. Usually this will be kept in ROM on the interface hardware. Some systems may set this address via software.

In a system where there are several such addresses the designer has a tougher choice. The address chosen should be the one most likely to be of use to network management (e.g. the address placed in ARP responses for systems which are primarily IP systems).

If the designer truly can not choose, use of the factory-provided ROM address is suggested.

If the address can not be determined, an octet string of zero length should be returned.

The address is stored in binary in this object. The address is stored in "canonical" bit order, that is, the Group Bit is positioned as the low-order bit of the first octet. Thus, the first byte of a multicast address would have the bit 0x01 set. This is true even when the interface is using token ring framing mode, which transmits addresses high-order bit first.

3.3.6. Specific Interface MIB Objects

The following table provides specific implementation guidelines for the interface group objects in the conformance groups listed above.

Object	Use for an 802.12 Interface
ifIndex	Each 802.12 interface is represented by an ifEntry. Interface tables in this MIB module are indexed by ifIndex.
ifDescr	Refer to [7].
ifType	The IANA reserved value for 802.12 - 55.
ifMtu	The value of ifMtu on an 802.12 interface will depend on the type of framing that is in use on that interface. Changing the dot12DesiredFramingType may have the effect of changing ifMtu after the next time that the interface trains. When dot12CurrentFramingType is equal to frameType88023, ifMtu will be equal to 1500. When dot12CurrentFramingType is equal to frameType88025, ifMtu will be 4464.
ifSpeed	The speed of the interface in bits per second. For current 802.12 implementations, this will be equal to 100,000,000 (100 million).
ifPhysAddress	See Section 3.3.5 .

ifAdminStatus	Write access is not required. Support for 'testing' is not required. Setting this object to 'up' will cause dot12Commands to be set to 'open'. Setting this object to 'down' will cause dot12Commands to be set to 'close'. Setting dot12Commands to 'open' will set this object to 'up'. Setting dot12Commands to 'close' will set this object to 'down'. Setting dot12Commands to 'reset' will have no effect on this object.
ifOperStatus	When dot12Status is equal to 'opened', this object will be equal to 'up'. When dot12Status is equal to 'closed', 'opening', 'openFailure' or 'linkFailure', this object will be equal to 'down'. Support for 'testing' is not required, but may be used to indicate that a vendor specific test is in progress. The value 'dormant' has no meaning for an IEEE 802.12 interface.
ifLastChange	Refer to [7].
ifInOctets	The number of octets in valid MAC frames received on this interface, including the MAC header and FCS.
ifInUcastPkts	Refer to [7].
ifInDiscards	Refer to [7].
ifInErrors	The sum for this interface of dot12InIPMErrors, dot12InOversizeFrameErrors, dot12InDataErrors, and any additional internal errors that may occur in an implementation.
ifInUnknownProtos	Refer to [7].
ifOutOctets	The number of octets transmitted in MAC frames on this interface, including the MAC header and FCS.
ifOutUcastPkts	Refer to [7].
ifOutDiscards	Refer to [7].

ifOutErrors	The number of implementation-specific internal transmit errors on this interface.
ifName	Locally-significant textual name for the interface (e.g. <code>vg0</code>).
ifInMulticastPkts	Refer to [7]. When <code>dot12CurrentFramingType</code> is <code>frameType88025</code> , this count includes packets addressed to functional addresses.
ifInBroadcastPkts	Refer to [7].
ifOutMulticastPkts	Refer to [7]. When <code>dot12CurrentFramingType</code> is <code>frameType88025</code> , this count includes packets addressed to functional addresses.
ifOutBroadcastPkts	Refer to [7].
ifHCInOctets	64-bit version of <code>ifInOctets</code> .
ifHCOctets	64-bit version of <code>ifOutOctets</code>
ifHC*Pkts	Not required for 100 MBit interfaces. Future IEEE 802.12 interfaces which operate at higher speeds may require implementation of these counters, but such interfaces are beyond the scope of this memo.
ifLinkUpDownTrapEnable	Refer to [7]. Default is 'enabled'.
ifHighSpeed	The speed of the interface in millions of bits per second. For current 802.12 implementations, this will be equal to 100.
ifPromiscuousMode	Reflects whether the interface has successfully trained and is currently operating in promiscuous mode. <code>dot12DesiredPromiscStatus</code> is used to select the promiscuous mode to be requested in the next training attempt. Setting <code>ifPromiscuousMode</code> will update <code>dot12DesiredPromiscStatus</code> and cause the interface to attempt to retrain using the new promiscuous mode. After the interface has retrained, <code>ifPromiscuousMode</code> will reflect the mode that is in use, not the mode that was requested.

ifConnectorPresent	This will normally be 'true'.
ifStackHigherLayer	Refer to section 3.3.1
ifStackLowerLayer	
ifStackStatus	
ifRcvAddressAddress	Refer to section 3.3.4 .
ifRcvAddressStatus	
ifRcvAddressType	

3.4. Relation to [RFC 1643](#), [RFC 1650](#), and [RFC 1748](#)

An IEEE 802.12 interface can be configured to operate in either ethernet or token ring framing mode. An IEEE 802.12 interface uses the frame format for the configured framing mode, but does not use the media access protocol for ethernet or token ring. Instead, IEEE 802.12 defines its own media access protocol, the Demand Priority Access Method (DPAM).

There are existing standards-track MIB modules for instrumenting ethernet-like interfaces and token ring interfaces. At the time of this writing, they are: STD 50, [RFC 1643](#), "Definitions of Managed Objects for Ethernet-like Interface Types" [8]; [RFC 1650](#), "Definitions of Managed Objects for Ethernet-like Interface Types using SMIV2" [9]; and [RFC 1748](#), "IEEE 802.5 MIB using SMIV2" [10]. These MIB modules are designed to instrument the media access protocol for these respective technologies. Since IEEE 802.12 interfaces do not implement either of these media access protocols, an agent should not implement [RFC 1643](#), [RFC 1650](#), or [RFC 1748](#) for IEEE 802.12 interfaces.

3.5. Relation to [RFC 1749](#)

When an IEEE 802.12 interface is operating in token ring framing mode, and the end node supports token ring source routing, the agent should implement [RFC 1749](#), the IEEE 802.5 Station Source Routing MIB [11] for those interfaces.

3.6. Master Mode Operation

In an IEEE 802.12 network, "master" devices act as network controllers to decide when to grant requesting end-nodes permission to transmit. These master devices may be repeaters, or other active controller devices such as switches.

Devices which do not act as network controllers, such as end-nodes or passive switches, are considered to be operating in "slave" mode.

The dot12ControlMode object indicates if the interface is operating in master mode or slave mode.

3.7. Normal and High Priority Counters

The IEEE 802.12 interface MIB does not provide normal priority transmit counters. Standardization of normal priority transmit counters could not be justified -- ifOutUcastPkts, ifOutMulticastPkts, ifOutBroadcastPkts, ifOutOctets, dot12OutHighPriorityFrames, and dot12OutHighPriorityOctets should suffice. More precisely, the number of normal priority frames transmitted can be calculated as:

$$\begin{aligned} \text{outNormPriorityFrames} = & \text{ifOutUcastPkts} & + \\ & \text{ifOutMulticastPkts} & + \\ & \text{ifOutBroadcastPkts} & - \\ & \text{dot12OutHighPriorityFrames} \end{aligned}$$

The number of normal priority octets transmitted can be calculated as:

$$\begin{aligned} \text{outNormPriorityOctets} = & \text{ifOutOctets} & - \\ & \text{dot12OutHighPriorityOctets} \end{aligned}$$

On the other hand, normal priority receive counters are provided. The main reason for this is that the normal priority and high priority counters include errored frames, whereas the ifIn*Pkts and ifInOctets do not include errored frames. dot12InNormPriorityFrames could be calculated, but the calculation is tedious:

$$\begin{aligned} \text{inNormPriorityFrames} = & \text{ifInUcastPkts} & + \\ & \text{ifInMulticastPkts} & + \\ & \text{ifInBroadcastPkts} & + \\ & \text{dot12InNullAddressedFrames} & + \\ & \text{ifInErrors} & + \\ & \text{ifInDiscards} & + \\ & \text{ifInUnknownProtos} & - \\ & \text{dot12InHighPriorityFrames} \end{aligned}$$

dot12InNormPriorityOctets includes octets in unreadable frames, which is not available elsewhere. The number of octets in unreadable frames can be calculated as:

$$\text{octetsInUnreadableFrames} = \text{dot12InNormPriorityOctets} + \text{dot12InHighPriorityOctets} - \text{ifInOctets}$$

Also, the total traffic at this interface can be calculated as:

$$\text{traffic} = \text{dot12InNormPriorityOctets} + \text{dot12InHighPriorityOctets} + \text{ifOutOctets}$$

In other words, the normal priority receive counters were deemed useful, whereas the normal priority transmit counters can be easily calculated from other available counters.

3.8. IEEE 802.12 Training Frames

Training frames are special MAC frames that are used only during link initialization. Training frames are initially constructed by the device at the lower end of a link, which is the slave mode device for the link. The training frame format is as follows:

```
+---+---+---+---+---+---+---+---+---+---+
| DA | SA | Req Config | Allow Config | Data | FCS |
+---+---+---+---+---+---+---+---+---+---+
```

DA = destination address (six octets)
 SA = source address (six octets)
 Req Config = requested configuration (2 octets)
 Allow Config = allowed configuration (2 octets)
 Data = data (594 to 675 octets)
 FCS = frame check sequence (4 octets)

Training frames are always sent with a null destination address. To pass training, an end node must use its source address in the source address field of the training frame. A repeater may use a non-null source address if it has one, or it may use a null source address.

The requested configuration field allows the slave mode device to inform the master mode device about itself and to request configuration options. The training response frame from the master mode device contains the slave mode device's requested configuration from the training request frame. The currently defined format of the requested configuration field as defined in the IEEE Standard 802.12-1995 standard is shown below. Please refer to the most current version of the IEEE document for a more up to date description of this field. In particular, the reserved bits may be used in later versions of the standard.

First Octet:	Second Octet:
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
+-----+	+-----+
v v v r r r r	r r r F F P P R
+-----+	+-----+

vvv: The version of the 802.12 training protocol with which the training initiator is compliant. The current version is 100.

r: Reserved bits (set to zero)

FF: 00 = frameType88023

01 = frameType88025

10 = reserved

11 = frameTypeEither

PP: 00 = singleAddressMode

01 = promiscuousMode

10 = reserved

11 = reserved

R: 0 = the training initiator is an end node

1 = the training initiator is a repeater

The allowed configuration field allows the master mode device to respond with the allowed configuration. The slave mode device sets the contents of this field to all zero bits. The master mode device sets the allowed configuration field as follows:

First Octet:	Second Octet:
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
+-----+	+-----+
v v v D C N r r	r r r F F P P R
+-----+	+-----+

vvv: The version of the 802.12 training protocol with which the training responder is compliant. The current version is 100.

```

D:  0 = No duplicate address has been detected.
    1 = Duplicate address has been detected
C:  0 = The requested configuration is compatible with the
      network.
    1 = The requested configuration is not compatible with
        the network.  In this case, the FF, PP, and R bits
        indicate the configuration that would be allowed.
N:  0 = Access will be allowed, providing the configuration
      is compatible (C = 0).
    1 = Access is not granted because of security
        restrictions
r:   Reserved bits (set to zero)
FF:  00 = frameType88023 will be used
      01 = frameType88025 will be used
      10 = reserved
      11 = reserved
PP:  00 = singleAddressMode
      01 = promiscuousMode
      10 = reserved
      11 = reserved
R:  0 = Requested access as an end node is allowed
    1 = Requested access as a repeater is allowed

```

Again, note that the most recent version of the IEEE 802.12 standard should be consulted for the most up to date definition of the requested configuration and allowed configuration fields.

The data field contains between 594 and 675 octets and is filled in by the training initiator. The first 55 octets may be used for vendor specific protocol information. The remaining octets are all zeros. The length of the training frame combined with the requirement that 24 consecutive training frames be received without error to complete training ensures that marginal links will not complete training.

3.9. Mapping of IEEE 802.12 Managed Objects

The following table lists all the managed objects defined for oEndNode in the IEEE 802.12 Standard, and the corresponding SNMP objects.

IEEE 802.12 Managed Object	Corresponding SNMP Object
oEndNode	
.aBroadcastFramesReceived	IF-MIB - ifInBroadcastPkts
.aBroadcastFramesTransmitted	IF-MIB - ifOutBroadcastPkts
.aDataErrorFramesReceived	dot12InDataErrors
.aDesiredFramingType	dot12DesiredFramingType

.aDesiredPromiscuousStatus	dot12DesiredPromiscStatus
.aFramesTransmitted	IF-MIB - ifOutUCastPkts + ifOutMulticastPkts + ifOutBroadcastPkts
.aFramingCapability	dot12FramingCapability
.aFunctionalAddresses	IF-MIB - ifRcvAddressTable
.aHighPriorityFramesReceived	dot12InHighPriorityFrames
.aHighPriorityFramesTransmitted	dot12OutHighPriorityFrames
.aHighPriorityOctetsReceived	dot12InHighPriorityOctets or dot12InHCHighPriorityOctets
.aHighPriorityOctetsTransmitted	dot12OutHighPriorityOctets or dot12OutHCHighPriorityOctets
.aIPMFramesReceived	dot12InIPMErrors
.aLastTrainingConfig	dot12LastTrainingConfig
.aMACID	IF-MIB - ifIndex
.aMACStatus	dot12Status
.aMACVersion	dot12TrainingVersion
.aMediaType	<not yet mapped> Tranceiver MIB issue
.aMulticastFramesReceived	IF-MIB - ifInMulticastPkts
.aMulticastFramesTransmitted	IF-MIB - ifOutMulticastPkts
.aMulticastReceiveStatus	IF-MIB - ifRcvAddressTable
.aNormalPriorityFramesReceived	dot12InNormPriorityFrames
.aNormalPriorityOctetsReceived	dot12InNormPriorityOctets or dot12InHCNormPriorityOctets
.aNullAddressedFramesReceived	dot12InNullAddressedFrames
.aOctetsTransmitted	IF-MIB - ifOutOctets or ifHCOutOctets
.aOversizeFramesReceived	dot12InOversizeFrameErrors
.aReadableFramesReceived	IF-MIB - ifInUCastPkts + ifInMulticastPkts + ifInBroadcastPkts
.aReadableOctetsReceived	IF-MIB - ifInOctets or ifHCInOctets
.aReadMulticastList	IF-MIB - ifRcvAddressTable
.aReadWriteMACAddress	IF-MIB - ifPhysAddress
.aTransitionsIntoTraining	dot12TransitionIntoTrainings
.acAddGroupAddress	IF-MIB - ifRcvAddressTable
.acClose	dot12Commands: 'close'
.acDeleteGroupAddress	IF-MIB - ifRcvAddressTable
.acExecuteSelftest	IF-MIB - ifAdminStatus
.acInitializeMAC	dot12Commands: 'reset'
.acOpen	dot12Commands: 'open'

4. Definitions

DOT12-IF-MIB DEFINITIONS ::= BEGIN

IMPORTS

transmission, Counter32, Counter64, OBJECT-TYPE,
MODULE-IDENTITY
FROM SNMPv2-SMI
MODULE-COMPLIANCE, OBJECT-GROUP
FROM SNMPv2-CONF
ifIndex
FROM IF-MIB;

dot12MIB MODULE-IDENTITY

LAST-UPDATED "9602220452Z" -- February 22, 1996
ORGANIZATION "IETF 100VG-AnyLAN MIB Working Group"
CONTACT-INFO

" John Flick

Postal: Hewlett Packard Company
8000 Foothills Blvd. M/S 5556
Roseville, CA 95747-5556
Tel: +1 916 785 4018
Fax: +1 916 785 3583

E-mail: johnf@hprnd.rose.hp.com"

DESCRIPTION

"This MIB module describes objects for
managing IEEE 802.12 interfaces."

::= { transmission 45 }

dot12MIBObjects OBJECT IDENTIFIER ::= { dot12MIB 1 }

dot12ConfigTable OBJECT-TYPE

SYNTAX SEQUENCE OF Dot12ConfigEntry
MAX-ACCESS not-accessible
STATUS current

DESCRIPTION

"Configuration information for a collection of
802.12 interfaces attached to a particular
system."

::= { dot12MIBObjects 1 }

dot12ConfigEntry OBJECT-TYPE

SYNTAX Dot12ConfigEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

```

        "Configuration for a particular interface to an
        802.12 medium."
INDEX      { ifIndex }
 ::= { dot12ConfigTable 1 }

Dot12ConfigEntry ::=
    SEQUENCE {
        dot12CurrentFramingType      INTEGER,
        dot12DesiredFramingType      INTEGER,
        dot12FramingCapability       INTEGER,
        dot12DesiredPromiscStatus    INTEGER,
        dot12TrainingVersion         INTEGER,
        dot12LastTrainingConfig      OCTET STRING,
        dot12Commands                INTEGER,
        dot12Status                  INTEGER,
        dot12ControlMode             INTEGER
    }

dot12CurrentFramingType OBJECT-TYPE
    SYNTAX      INTEGER {
                    frameType88023(1),
                    frameType88025(2),
                    frameTypeUnknown(3)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "When dot12DesiredFramingType is one of
        'frameType88023' or 'frameType88025', this is the
        type of framing asserted by the interface.

        When dot12DesiredFramingType is 'frameTypeEither',
        dot12CurrentFramingType shall be one of
        'frameType88023' or 'frameType88025' when the
        dot12Status is 'opened'.  When the dot12Status is
        anything other than 'opened',
        dot12CurrentFramingType shall take the value of
        'frameTypeUnknown'."
    ::= { dot12ConfigEntry 1 }

dot12DesiredFramingType OBJECT-TYPE
    SYNTAX      INTEGER {
                    frameType88023(1),
                    frameType88025(2),
                    frameTypeEither(3)
                }
    MAX-ACCESS  read-write
    STATUS      current

```

DESCRIPTION

"The type of framing which will be requested by the interface during the next interface MAC initialization or open action.

In master mode, this is the framing mode which will be granted by the interface. Note that for a master mode interface, this object must be equal to 'frameType88023' or 'frameType88025', since a master mode interface cannot grant 'frameTypeEither'."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aDesiredFramingType."

::= { dot12ConfigEntry 2 }

dot12FramingCapability OBJECT-TYPE

SYNTAX INTEGER {
 frameType88023(1),
 frameType88025(2),
 frameTypeEither(3)
}

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The type of framing this interface is capable of supporting."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aFramingCapability."

::= { dot12ConfigEntry 3 }

dot12DesiredPromiscStatus OBJECT-TYPE

SYNTAX INTEGER {
 singleAddressMode(1),
 promiscuousMode(2)
}

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object is used to select the promiscuous mode that this interface will request in the next training packet issued on this interface. Whether the repeater grants the requested mode must be verified by examining the state of the PP bits in the corresponding instance of dot12LastTrainingConfig."

In master mode, this object controls whether or not promiscuous mode will be granted by the interface when requested by the lower level device.

Note that this object indicates the desired mode for the next time the interface trains. The currently active mode will be reflected in dot12LastTrainingConfig and in ifPromiscuousMode."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aDesiredPromiscuousStatus."

::= { dot12ConfigEntry 4 }

dot12TrainingVersion OBJECT-TYPE

SYNTAX INTEGER (0..7)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The value that will be used in the version bits (vvv bits) in training frames on this interface. This is the highest version number supported by this MAC."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aMACVersion."

::= { dot12ConfigEntry 5 }

dot12LastTrainingConfig OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(2))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This 16 bit field contains the configuration bits from the most recent error-free training frame received during training on this interface. Training request frames are received when in master mode, while training response frames are received in slave mode. On master mode interfaces, this object contains the contents of the requested configuration field of the most recent training request frame. On slave mode interfaces, this object contains the contents of the allowed configuration field of the most recent training response frame. The format of the current version of this field is described in [section 3.8](#). Please refer to the most recent version of the IEEE 802.12 standard for the most up-to-date definition

of the format of this object."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aLastTrainingConfig."

::= { dot12ConfigEntry 6 }

dot12Commands OBJECT-TYPE

SYNTAX INTEGER {
noOp(1),
open(2),
reset(3),
close(4)
}

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"If the current value of dot12Status is 'closed', setting the value of this object to 'open' will change the corresponding instance of MIB-II's ifAdminStatus to 'up', cause this interface to enter the 'opening' state, and will cause training to be initiated on this interface. The progress and success of the open is given by the values of the dot12Status object. Setting this object to 'open' when dot12Status has a value other than 'closed' has no effect.

Setting the corresponding instance of ifAdminStatus to 'up' when the current value of dot12Status is 'closed' will have the same effect as setting this object to 'open'. Setting ifAdminStatus to 'up' when dot12Status has a value other than 'closed' has no effect.

Setting the value of this object to 'close' will move this interface into the 'closed' state and cause all transmit and receive actions to stop. This object will then have to be set to 'open' in order to reinitiate training.

Setting the corresponding instance of ifAdminStatus to 'down' will have the same effect as setting this object to 'close'.

Setting the value of this object to 'reset' when the current value of dot12Status has a value other than 'closed' will reset the interface. On a reset, all MIB counters should retain their values.

This will cause the MAC to initiate an acInitializeMAC action as specified in IEEE 802.12. This will cause training to be reinitiated on this interface. Setting this object to 'reset' when dot12Status has a value of 'closed' has no effect. Setting this object to 'reset' has no effect on the corresponding instance of ifAdminStatus.

Setting the value of this object to 'noOp' has no effect.

When read, this object will always have a value of 'noOp'."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.2, acOpen, acClose, acInitializeMAC. Also, [RFC1231](#) IEEE802.5 Token Ring MIB, dot5Commands."

::= { dot12ConfigEntry 7 }

dot12Status OBJECT-TYPE

```
SYNTAX      INTEGER {
                opened(1),
                closed(2),
                opening(3),
                openFailure(5),
                linkFailure(6)
            }
```

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The current interface status with respect to training. One of the following values:

opened	- Training has completed successfully.
closed	- MAC has been disabled by setting dot12Commands to 'close'.
opening	- MAC is in training. Training signals have been received.
openFailure	- Passed 24 error-free packets, but there is a problem, noted in the training configuration bits (dot12LastTrainingConfig).
linkFailure	- Training signals not received, or could not pass 24 error-free packets.

Whenever the dot12Commands object is set to 'close' or ifAdminStatus is set to 'down', the MAC will go silent, dot12Status will be 'closed', and ifOperStatus will be 'down'.

When the value of this object is equal to 'closed' and the dot12Commands object is set to 'open' or the ifAdminStatus object is set to 'up', training will be initiated on this interface. When the value of this object is not equal to 'closed' and the dot12Commands object is set to 'reset', training will be reinitiated on this interface. Note that sets of some other objects (e.g. dot12ControlMode) or external events (e.g. MAC protocol violations) may also cause training to be reinitiated on this interface.

When training is initiated or reinitiated on an interface, the end node will send Training_Up to the master and initially go to the 'linkFailure' state and ifOperStatus will go to 'down'. When the master sends back Training_Down, dot12Status will change to the 'opening' state, and training packets will be transferred.

After all of the training packets have been passed, dot12Status will change to 'linkFailure' if 24 consecutive error-free packets were not passed, 'opened' if 24 consecutive error-free packets were passed and the training configuration bits were OK, or 'openFailure' if there were 24 consecutive error-free packets, but there was a problem with the training configuration bits.

When in the 'openFailure' state, the dot12LastTrainingConfig object will contain the configuration bits from the last training packet which can be examined to determine the exact reason for the training configuration failure.

If training did not succeed (dot12Status is 'linkFailure' or 'openFailure'), the entire process will be restarted after MAC_Retraining_Delay_Timer seconds.

If training does succeed (dot12Status changes to

'opened'), ifOperStatus will change to 'up'. If training does not succeed (dot12Status changes to 'linkFailure' or 'openFailure'), ifOperStatus will remain 'down'."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aMACStatus."

::= { dot12ConfigEntry 8 }

dot12ControlMode OBJECT-TYPE

SYNTAX INTEGER {
 masterMode(1),
 slaveMode(2),
 learn(3)
}

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object is used to configure and report whether or not this interface is operating in master mode. In a Demand Priority network, end node interfaces typically operate in slave mode, while switch interfaces may control the Demand Priority protocol and operate in master mode.

This object may be implemented as a read-only object by those agents and interfaces that do not implement software control of master mode. In particular, interfaces that cannot operate in master mode, and interfaces on which master mode is controlled by a pushbutton on the device, should implement this object read-only.

Some interfaces do not require network management configuration of this feature and can autosense whether to use master mode or slave mode. The value 'learn' is used for that purpose. While autosense is taking place, the value 'learn' is returned.

A network management operation which modifies the value of dot12ControlMode causes the interface to retrain."

::= { dot12ConfigEntry 9 }

dot12StatTable OBJECT-TYPE

SYNTAX SEQUENCE OF Dot12StatEntry

MAX-ACCESS not-accessible

```

STATUS      current
DESCRIPTION
    "Statistics for a collection of 802.12 interfaces
    attached to a particular system."
 ::= { dot12MIBObjects 2 }

```

```

dot12StatEntry OBJECT-TYPE
    SYNTAX      Dot12StatEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Statistics for a particular interface to an
        802.12 medium.  The receive statistics in this
        table apply only to packets received by this
        station (i.e., packets whose destination address
        is either the local station address, the
        broadcast address, or a multicast address that
        this station is receiving, unless the station is
        in promiscuous mode)."
    INDEX       { ifIndex }
    ::= { dot12StatTable 1 }

```

```

Dot12StatEntry ::=
    SEQUENCE {
        dot12InHighPriorityFrames      Counter32,
        dot12InHighPriorityOctets      Counter32,
        dot12InNormPriorityFrames      Counter32,
        dot12InNormPriorityOctets      Counter32,
        dot12InIPMErrors              Counter32,
        dot12InOversizeFrameErrors     Counter32,
        dot12InDataErrors              Counter32,
        dot12InNullAddressedFrames     Counter32,
        dot12OutHighPriorityFrames      Counter32,
        dot12OutHighPriorityOctets      Counter32,
        dot12TransitionIntoTrainings   Counter32,
        dot12HCInHighPriorityOctets     Counter64,
        dot12HCInNormPriorityOctets     Counter64,
        dot12HCOuthighPriorityOctets    Counter64
    }

```

```

dot12InHighPriorityFrames OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object is a count of high priority frames
        that have been received on this interface.
        Includes both good and bad high priority frames,

```

as well as high priority training frames. Does not include normal priority frames which were priority promoted."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aHighPriorityFramesReceived."

::= { dot12StatEntry 1 }

dot12InHighPriorityOctets OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of the number of octets contained in high priority frames that have been received on this interface. This counter is incremented by OctetCount for each frame received on this interface which is counted by dot12InHighPriorityFrames."

Note that this counter will roll over very quickly. It is provided for backward compatibility for Network Management protocols that do not support 64 bit counters (e.g. SNMP version 1)."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aHighPriorityOctetsReceived."

::= { dot12StatEntry 2 }

dot12InNormPriorityFrames OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of normal priority frames that have been received on this interface. Includes both good and bad normal priority frames, as well as normal priority training frames and normal priority frames which were priority promoted."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aNormalPriorityFramesReceived."

::= { dot12StatEntry 3 }

dot12InNormPriorityOctets OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object is a count of the number of octets contained in normal priority frames that have been received on this interface. This counter is incremented by OctetCount for each frame received on this interface which is counted by dot12InNormPriorityFrames.

 Note that this counter will roll over very quickly. It is provided for backward compatibility for Network Management protocols that do not support 64 bit counters (e.g. SNMP version 1)."
REFERENCE
 "IEEE Standard 802.12-1995, 13.2.5.2.1, aNormalPriorityOctetsReceived."
::= { dot12StatEntry 4 }

dot12InIPMErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object is a count of the number of frames that have been received on this interface with an invalid packet marker and no PMI errors. A repeater will write an invalid packet marker to the end of a frame containing errors as it is forwarded through the repeater to the other ports. This counter is incremented by one for each frame received on this interface which has had an invalid packet marker added to the end of the frame."
REFERENCE
 "IEEE Standard 802.12-1995, 13.2.5.2.1, aIPMFramesReceived."
::= { dot12StatEntry 5 }

dot12InOversizeFrameErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object is a count of oversize frames received on this interface. This counter is incremented by one for each frame received on

this interface whose OctetCount is larger than the maximum legal frame size. The frame size which causes this counter to increment is dependent on the current framing type."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aOversizeFramesReceived."

::= { dot12StatEntry 6 }

dot12InDataErrors OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of errored frames received on this interface. This counter is incremented by one for each frame received on this interface with any of the following errors: bad FCS (with no IPM), PMI errors (excluding frames with an IPM as the only PMI error), undersize, bad start of frame delimiter, or bad end of packet marker. Does not include frames counted by dot12InIPMErrors, dot12InNullAddressedFrames, or dot12InOversizeFrameErrors.

This counter indicates problems with the cable directly attached to this interface, while dot12InIPMErrors indicates problems with remote cables."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1, aDataErrorFramesReceived."

::= { dot12StatEntry 7 }

dot12InNullAddressedFrames OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of null addressed frames received on this interface. This counter is incremented by one for each frame received on this interface with a destination MAC address consisting of all zero bits. Both void and training frames are included in this counter.

Note that since this station would normally not

receive null addressed frames, this counter is only incremented when this station is operating in promiscuous mode or in training."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aNullAddressedFramesReceived."

::= { dot12StatEntry 8 }

dot12OutHighPriorityFrames OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This counter is incremented by one for each high priority frame successfully transmitted out this interface."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aHighPriorityFramesTransmitted."

::= { dot12StatEntry 9 }

dot12OutHighPriorityOctets OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This counter is incremented by OctetCount for each frame counted by dot12OutHighPriorityFrames."

Note that this counter will roll over very quickly. It is provided for backward compatibility for Network Management protocols that do not support 64 bit counters (e.g. SNMP version 1)."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aHighPriorityOctetsTransmitted."

::= { dot12StatEntry 10 }

dot12TransitionIntoTrainings OBJECT-TYPE

SYNTAX Counter32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of the number of times this interface has entered the training state. This counter is incremented by one each time dot12Status transitions to 'linkFailure' from any

state other than 'opening' or 'openFailure'."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aTransitionsIntoTraining."
::= { dot12StatEntry 11 }

dot12HCInHighPriorityOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of the number of octets contained in high priority frames that have been received on this interface. This counter is incremented by OctetCount for each frame received on this interface which is counted by dot12InHighPriorityFrames.

This counter is a 64 bit version of dot12InHighPriorityOctets. It should be used by Network Management protocols which support 64 bit counters (e.g. SNMPv2)."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aHighPriorityOctetsReceived."
::= { dot12StatEntry 12 }

dot12HCInNormPriorityOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is a count of the number of octets contained in normal priority frames that have been received on this interface. This counter is incremented by OctetCount for each frame received on this interface which is counted by dot12InNormPriorityFrames.

This counter is a 64 bit version of dot12InNormPriorityOctets. It should be used by Network Management protocols which support 64 bit counters (e.g. SNMPv2)."

REFERENCE

"IEEE Standard 802.12-1995, 13.2.5.2.1,
aNormalPriorityOctetsReceived."
::= { dot12StatEntry 13 }

```
dot12HCOuthighPriorityOctets OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This counter is incremented by OctetCount for
        each frame counted by dot12OutHighPriorityFrames.

        This counter is a 64 bit version of
        dot12OutHighPriorityOctets. It should be used by
        Network Management protocols which support 64 bit
        counters (e.g. SNMPv2)."
```

REFERENCE

```
    "IEEE Standard 802.12-1995, 13.2.5.2.1,
    aHighPriorityOctetsTransmitted."
 ::= { dot12StatEntry 14 }
```

-- conformance information

```
dot12Conformance OBJECT IDENTIFIER ::= { dot12MIB 2 }
```

```
dot12Compliances OBJECT IDENTIFIER ::= { dot12Conformance 1 }
dot12Groups      OBJECT IDENTIFIER ::= { dot12Conformance 2 }
```

-- compliance statements

```
dot12Compliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for managed network
        entities that have 802.12 interfaces."
```

MODULE -- this module

```
    MANDATORY-GROUPS { dot12ConfigGroup, dot12StatsGroup }
```

OBJECT dot12DesiredFramingType

```
MIN-ACCESS read-only
DESCRIPTION
    "Write access to this object is not required."
```

OBJECT dot12DesiredPromiscStatus

```
MIN-ACCESS read-only
DESCRIPTION
    "Write access to this object is not required."
```

OBJECT dot12Commands

```
MIN-ACCESS read-only
DESCRIPTION
```

```
        "Write access to this object is not required."

        OBJECT      dot12ControlMode
        MIN-ACCESS   read-only
        DESCRIPTION
            "Write access to this object is not required."
        ::= { dot12Compliances 1 }

-- units of conformance

dot12ConfigGroup OBJECT-GROUP
    OBJECTS      { dot12DesiredFramingType,
                    dot12FramingCapability,
                    dot12DesiredPromiscStatus,
                    dot12TrainingVersion,
                    dot12LastTrainingConfig,
                    dot12Commands, dot12Status,
                    dot12CurrentFramingType,
                    dot12ControlMode }
    STATUS      current
    DESCRIPTION
        "A collection of objects for managing the status
         and configuration of IEEE 802.12 interfaces."
    ::= { dot12Groups 1 }

dot12StatsGroup OBJECT-GROUP
    OBJECTS      { dot12InHighPriorityFrames,
                    dot12InHighPriorityOctets,
                    dot12InNormPriorityFrames,
                    dot12InNormPriorityOctets,
                    dot12InIPMErrors,
                    dot12InOversizeFrameErrors,
                    dot12InDataErrors,
                    dot12InNullAddressedFrames,
                    dot12OutHighPriorityFrames,
                    dot12OutHighPriorityOctets,
                    dot12TransitionIntoTrainings,
                    dot12HCInHighPriorityOctets,
                    dot12HCInNormPriorityOctets,
                    dot12HCOuthighPriorityOctets }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing statistics for
         IEEE 802.12 interfaces."
    ::= { dot12Groups 2 }
```

END

5. Acknowledgements

This document was produced by the IETF 100VG-AnyLAN Working Group. It is based on the work of IEEE 802.12.

6. References

- [1] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization. International Standard 8824 (December, 1987).
- [2] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1902](#), SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.
- [3] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1903](#), SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.
- [4] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1904](#), SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, January 1996.
- [5] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets - MIB-II", STD 17, [RFC 1213](#), Hughes LAN Systems, Performance Systems International, March 1991.
- [6] IEEE, "Demand Priority Access Method, Physical Layer and Repeater Specifications for 100 Mb/s Operation", IEEE Standard 802.12-1995"
- [7] McCloghrie, K., and Kastenholz, F., "Evolution of the Interfaces Group of MIB-II", [RFC 1573](#), Hughes LAN Systems, FTP Software, January 1994.
- [8] Kastenholz, F., "Definitions of Managed Objects for the Ethernet-like Interface Types", STD 50, [RFC 1643](#), FTP Software, Inc., July, 1994.

- [9] Kastenholz, F., "Definitions of Managed Objects for the Ethernet-like Interface Types using SMIV2", [RFC 1650](#), FTP Software, Inc., August, 1994.
- [10] McCloghrie, K., and Decker, E., "IEEE 802.5 MIB using SMIV2", [RFC 1748](#), Cisco Systems, Inc., December, 1994.
- [11] McCloghrie, K., Baker, F., and Decker, E., "IEEE 802.5 Station Source Routing MIB using SMIV2", [RFC 1749](#), Cisco Systems, Inc., December, 1994.

7. Security Considerations

Security issues are not discussed in this memo.

8. Author's Address

John Flick
Hewlett Packard Company
8000 Foothills Blvd. M/S 5556
Roseville, CA 95747-5556

Phone: +1 916 785 4018
Email: johnf@hprnd.rose.hp.com