

Network Working Group  
Request for Comments: 1623  
Obsoletes: 1398  
STD: 50  
Category: Standards Track

F. Kastenholz  
FTP Software, Inc.  
May 1994

## Definitions of Managed Objects for the Ethernet-like Interface Types

### 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

Introduction .....	1
1. The SNMP Network Management Framework .....	2
1.1 Object Definitions .....	2
2. Change Log .....	2
3. Overview .....	3
3.1 Relation to RFC 1213 .....	4
3.2 Relation to RFC 1573 .....	4
3.2.1 Layering Model .....	4
3.2.2 Virtual Circuits .....	4
3.2.3 ifTestTable .....	4
3.2.4 ifRcvAddressTable .....	5
3.2.5 ifPhysAddress .....	5
3.2.6 ifType .....	6
4. Definitions .....	6
5. Acknowledgements .....	16
6. References .....	17
7. Security Considerations .....	19
8. Author's Address .....	19

### Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it defines objects for managing ethernet-like objects.

This memo also includes a MIB module. This MIB module corrects minor errors in the earlier version of this MIB: RFC 1398 [15].

## 1. The SNMP Network Management Framework

The SNMP Network Management Framework consists of three major components. They are:

- o STD 16/RFC 1155 [3] which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management. STD 16/RFC 1212 [13] defines a more concise description mechanism, which is wholly consistent with the SMI.
- o RFC 1156 [4] which defines MIB-I, the core set of managed objects for the Internet suite of protocols. STD 17/RFC 1213 [6] defines MIB-II, an evolution of MIB-I based on implementation experience and new operational requirements.
- o STD 15/RFC 1157 [5] which defines the SNMP, 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) [7] defined in the SMI [16]. 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. Change Log

This section enumerates changes made to RFC 1398 to produce this document.

- (1) A section describing the applicability of various parts of RFC 1573 to ethernet-like interfaces has been added.
- (2) A minor error in the description of the TDR test was fixed.
- (3) A loopback test was defined to replace the standard loopback test that was defined in RFC 1229.

- (4) The description of dot3CollFrequencies was made a bit clearer.
- (5) A new object, EtherChipset, has been added. This object replaces the ifExtnsChipSet object, which has been removed per the Interface MIB Evolution effort.
- (6) Several minor editorial changes, spelling corrections, grammar and punctuation corrections, and so forth, were made.

### 3. Overview

Instances of these object types represent attributes of an interface to an ethernet-like communications medium. At present, ethernet-like media are identified by three values of the ifType object in the Internet-standard MIB:

```
ethernet-csmacd(6)
iso88023-csmacd(7)
starLan(11)
```

For these interfaces, the value of the ifSpecific variable in the MIB-II [6] has the OBJECT IDENTIFIER value:

```
dot3    OBJECT IDENTIFIER ::= { experimental 3 }
```

The definitions presented here are based on the IEEE 802.3 Layer Management Specification [9], as originally interpreted by Frank Kastenholz then of Interlan in [10]. 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 [9] are represented by previously defined objects in the Internet-standard MIB or in the Generic Interface Extensions MIB [11], 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 number of frames transmitted or received on a particular interface, the promiscuous status of an interface, the MAC address of an interface, and multicast information associated with an interface.

### 3.1. 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 ethernet-like 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.

### 3.2. 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 were intentionally left vague in RFC 1573 to avoid over constraining the MIB, thereby precluding management of certain media-types.

Section 3.3 of RFC 1573 enumerates several areas which a media-specific MIB must clarify. 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.

#### 3.2.1. Layering Model

This MIB does not provide for layering. There are no sublayers.

##### EDITOR'S NOTE:

I could foresee the development of an 802.2 and enet-transceiver MIB. They could be higher and lower sublayers, respectively. All that THIS document should do is allude to the possibilities and urge the implementor to be aware of the possibility and that they may have requirements which supersede the requirements in this document.

#### 3.2.2. Virtual Circuits

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

#### 3.2.3. ifTestTable

This MIB defines two tests for media which are instrumented with this MIB; TDR and Loopback. Implementation of these tests is not required. Many common interface chips do not support one or both of these tests.

These two tests are provided as a convenience, allowing a common method to invoke the test.

Standard MIBs do not include objects in which to return the results of the TDR test. Any needed objects **MUST** be provided in the vendor specific MIB.

#### 3.2.4. ifRcvAddressTable

This table contains all IEEE 802.3 addresses, unicast, multicast, and broadcast, for which this interface will receive packets and forward them up to a higher layer entity for local consumption. The format of the address, contained in ifRcvAddressAddress, is the same as for ifPhysAddress.

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.

#### 3.2.5. ifPhysAddress

This object contains the IEEE 802.3 address which is placed in the source-address field of any Ethernet, Starlan, or IEEE 802.3 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 chose, 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.

### 3.2.6. ifType

This MIB applies to interfaces which have any of the following three ifType values:

```
ethernet-csmacd(6)
iso88023-csmacd(7)
starLan(11)
```

Interfaces with any of these ifType values map to the EtherLike-MIB in the same manner. The EtherLike-MIB applies equally to all three types; there are no implementation differences.

## 4. Definitions

```
EtherLike-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    Counter, Gauge    FROM RFC1155-SMI
    transmission       FROM RFC1213-MIB
    OBJECT-TYPE        FROM RFC-1212;
```

```
-- This MIB module uses the extended OBJECT-TYPE macro as
-- defined in RFC-1212.
```

```
dot3      OBJECT IDENTIFIER ::= { transmission 7 }
```

```
-- the Ethernet-like Statistics group
```

```
dot3StatsTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3StatsEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Statistics for a collection of ethernet-like
         interfaces attached to a particular system."
    ::= { dot3 2 }
```

```
dot3StatsEntry  OBJECT-TYPE
    SYNTAX      Dot3StatsEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Statistics for a particular interface to an
         ethernet-like medium."
    INDEX      { dot3StatsIndex }
    ::= { dot3StatsTable 1 }
```

```

Dot3StatsEntry ::= SEQUENCE {
    dot3StatsIndex                INTEGER,
    dot3StatsAlignmentErrors      Counter,
    dot3StatsFCSErrors            Counter,
    dot3StatsSingleCollisionFrames Counter,
    dot3StatsMultipleCollisionFrames Counter,
    dot3StatsSQETestErrors        Counter,
    dot3StatsDeferredTransmissions Counter,
    dot3StatsLateCollisions       Counter,
    dot3StatsExcessiveCollisions  Counter,
    dot3StatsInternalMacTransmitErrors Counter,
    dot3StatsCarrierSenseErrors   Counter,
    dot3StatsFrameTooLongs        Counter,
    dot3StatsInternalMacReceiveErrors Counter
}

dot3StatsIndex OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS       read-only
    STATUS       mandatory
    DESCRIPTION
        "An index value that uniquely identifies an
        interface to an ethernet-like medium. The
        interface identified by a particular value of
        this index is the same interface as identified
        by the same value of ifIndex."
    ::= { dot3StatsEntry 1 }

dot3StatsAlignmentErrors OBJECT-TYPE
    SYNTAX      Counter
    ACCESS       read-only
    STATUS       mandatory
    DESCRIPTION
        "A count of frames received on a particular
        interface that are not an integral number of
        octets in length and do not pass the FCS check.

        The count represented by an instance of this
        object is incremented when the alignmentError
        status is returned by the MAC service to the
        LLC (or other MAC user). Received frames for
        which multiple error conditions obtain are,
        according to the conventions of IEEE 802.3
        Layer Management, counted exclusively according
        to the error status presented to the LLC."
    REFERENCE
        "IEEE 802.3 Layer Management"
    ::= { dot3StatsEntry 2 }

```

dot3StatsFCSErrors    OBJECT-TYPE  
    SYNTAX            Counter  
    ACCESS            read-only  
    STATUS            mandatory  
    DESCRIPTION  
        "A count of frames received on a particular  
        interface that are an integral number of octets  
        in length but do not pass the FCS check."

The count represented by an instance of this object is incremented when the frameCheckError status is returned by the MAC service to the LLC (or other MAC user). Received frames for which multiple error conditions obtain are, according to the conventions of IEEE 802.3 Layer Management, counted exclusively according to the error status presented to the LLC."

REFERENCE  
    "IEEE 802.3 Layer Management"  
    ::= { dot3StatsEntry 3 }

dot3StatsSingleCollisionFrames    OBJECT-TYPE  
    SYNTAX            Counter  
    ACCESS            read-only  
    STATUS            mandatory  
    DESCRIPTION  
        "A count of successfully transmitted frames on  
        a particular interface for which transmission  
        is inhibited by exactly one collision."

A frame that is counted by an instance of this object is also counted by the corresponding instance of either the ifOutUcastPkts, ifOutMulticastPkts, or ifOutBroadcastPkts, and is not counted by the corresponding instance of the dot3StatsMultipleCollisionFrames object."

REFERENCE  
    "IEEE 802.3 Layer Management"  
    ::= { dot3StatsEntry 4 }

dot3StatsMultipleCollisionFrames    OBJECT-TYPE  
    SYNTAX            Counter  
    ACCESS            read-only  
    STATUS            mandatory  
    DESCRIPTION  
        "A count of successfully transmitted frames on  
        a particular interface for which transmission



is inhibited by more than one collision.

A frame that is counted by an instance of this object is also counted by the corresponding instance of either the ifOutUcastPkts, ifOutMulticastPkts, or ifOutBroadcastPkts, and is not counted by the corresponding instance of the dot3StatsSingleCollisionFrames object."

REFERENCE

"IEEE 802.3 Layer Management"  
::= { dot3StatsEntry 5 }

dot3StatsSQETestErrors OBJECT-TYPE

SYNTAX Counter  
ACCESS read-only  
STATUS mandatory

DESCRIPTION

"A count of times that the SQE TEST ERROR message is generated by the PLS sublayer for a particular interface. The SQE TEST ERROR message is defined in section 7.2.2.2.4 of ANSI/IEEE 802.3-1985 and its generation is described in section 7.2.4.6 of the same document."

REFERENCE

"ANSI/IEEE Std 802.3-1985 Carrier Sense Multiple Access with Collision Detection Access Method and Physical Layer Specifications"  
::= { dot3StatsEntry 6 }

dot3StatsDeferredTransmissions OBJECT-TYPE

SYNTAX Counter  
ACCESS read-only  
STATUS mandatory

DESCRIPTION

"A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy.

The count represented by an instance of this object does not include frames involved in collisions."

REFERENCE

"IEEE 802.3 Layer Management"  
::= { dot3StatsEntry 7 }

dot3StatsLateCollisions OBJECT-TYPE

SYNTAX        Counter  
 ACCESS       read-only  
 STATUS       mandatory  
 DESCRIPTION  
 "The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet.

Five hundred and twelve bit-times corresponds to 51.2 microseconds on a 10 Mbit/s system. A (late) collision included in a count represented by an instance of this object is also considered as a (generic) collision for purposes of other collision-related statistics."

REFERENCE  
 "IEEE 802.3 Layer Management"  
 ::= { dot3StatsEntry 8 }

dot3StatsExcessiveCollisions    OBJECT-TYPE

SYNTAX       Counter  
 ACCESS       read-only  
 STATUS       mandatory  
 DESCRIPTION  
 "A count of frames for which transmission on a particular interface fails due to excessive collisions."  
 REFERENCE  
 "IEEE 802.3 Layer Management"  
 ::= { dot3StatsEntry 9 }

dot3StatsInternalMacTransmitErrors    OBJECT-TYPE

SYNTAX       Counter  
 ACCESS       read-only  
 STATUS       mandatory  
 DESCRIPTION  
 "A count of frames for which transmission on a particular interface fails due to an internal MAC sublayer transmit error. A frame is only counted by an instance of this object if it is not counted by the corresponding instance of either the dot3StatsLateCollisions object, the dot3StatsExcessiveCollisions object, or the dot3StatsCarrierSenseErrors object.

The precise meaning of the count represented by

an instance of this object is implementation-specific. In particular, an instance of this object may represent a count of transmission errors on a particular interface that are not otherwise counted."

## REFERENCE

"IEEE 802.3 Layer Management"

::= { dot3StatsEntry 10 }

dot3StatsCarrierSenseErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

## DESCRIPTION

"The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface."

The count represented by an instance of this object is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt."

## REFERENCE

"IEEE 802.3 Layer Management"

::= { dot3StatsEntry 11 }

-- { dot3StatsEntry 12 } is not assigned

dot3StatsFrameTooLongs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

## DESCRIPTION

"A count of frames received on a particular interface that exceed the maximum permitted frame size."

The count represented by an instance of this object is incremented when the frameTooLong status is returned by the MAC service to the LLC (or other MAC user). Received frames for which multiple error conditions obtain are, according to the conventions of IEEE 802.3 Layer Management, counted exclusively according to the error status presented to the LLC."

## REFERENCE

"IEEE 802.3 Layer Management"  
 ::= { dot3StatsEntry 13 }

-- { dot3StatsEntry 14 } is not assigned

-- { dot3StatsEntry 15 } is not assigned

dot3StatsInternalMacReceiveErrors    OBJECT-TYPE  
     SYNTAX       Counter  
     ACCESS       read-only  
     STATUS       mandatory  
     DESCRIPTION  
         "A count of frames for which reception on a  
         particular interface fails due to an internal  
         MAC sublayer receive error. A frame is only  
         counted by an instance of this object if it is  
         not counted by the corresponding instance of  
         either the dot3StatsFrameTooLongs object, the  
         dot3StatsAlignmentErrors object, or the  
         dot3StatsFCSErrors object.

The precise meaning of the count represented by  
 an instance of this object is implementation-  
 specific. In particular, an instance of this  
 object may represent a count of receive errors  
 on a particular interface that are not  
 otherwise counted."

REFERENCE  
 "IEEE 802.3 Layer Management"  
 ::= { dot3StatsEntry 16 }

dot3StatsEtherChipSet    OBJECT-TYPE  
     SYNTAX       OBJECT IDENTIFIER  
     ACCESS       read-only  
     STATUS       mandatory  
     DESCRIPTION  
         "This object contains an OBJECT IDENTIFIER  
         which identifies the chipset used to  
         realize the interface. Ethernet-like  
         interfaces are typically built out of  
         several different chips. The MIB implementor  
         is presented with a decision of which chip  
         to identify via this object. The implementor  
         should identify the chip which is usually  
         called the Medium Access Control chip.  
         If no such chip is easily identifiable,  
         the implementor should identify the chip  
         which actually gathers the transmit

and receive statistics and error indications. This would allow a manager station to correlate the statistics and the chip generating them, giving it the ability to take into account any known anomalies in the chip."

::= { dot3StatsEntry 17 }

-- the Ethernet-like Collision Statistics group

-- Implementation of this group is optional; it is appropriate  
 -- for all systems which have the necessary metering

dot3CollTable OBJECT-TYPE  
 SYNTAX SEQUENCE OF Dot3CollEntry  
 ACCESS not-accessible  
 STATUS mandatory  
 DESCRIPTION  
 "A collection of collision histograms for a particular set of interfaces."  
 ::= { dot3 5 }

dot3CollEntry OBJECT-TYPE  
 SYNTAX Dot3CollEntry  
 ACCESS not-accessible  
 STATUS mandatory  
 DESCRIPTION  
 "A cell in the histogram of per-frame collisions for a particular interface. An instance of this object represents the frequency of individual MAC frames for which the transmission (successful or otherwise) on a particular interface is accompanied by a particular number of media collisions."  
 INDEX { ifIndex, dot3CollCount }  
 ::= { dot3CollTable 1 }

Dot3CollEntry ::= SEQUENCE {  
 dot3CollCount INTEGER,  
 dot3CollFrequencies Counter  
 }

-- { dot3CollEntry 1 } is no longer in use

dot3CollCount OBJECT-TYPE  
 SYNTAX INTEGER (1..16)

```

ACCESS      not-accessible
STATUS      mandatory
DESCRIPTION
"The number of per-frame media collisions for
which a particular collision histogram cell
represents the frequency on a particular
interface."
 ::= { dot3CollEntry 2 }

```

```

dot3CollFrequencies  OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
    "A count of individual MAC frames for which the
    transmission (successful or otherwise) on a
    particular interface occurs after the
    frame has experienced exactly the number
    of collisions in the associated
    dot3CollCount object.

```

```

    For example, a frame which is transmitted
    on interface 77 after experiencing
    exactly 4 collisions would be indicated
    by incrementing only dot3CollFrequencies.77.4.
    No other instance of dot3CollFrequencies would
    be incremented in this example."
    ::= { dot3CollEntry 3 }

```

#### -- 802.3 Tests

```
dot3Tests  OBJECT IDENTIFIER ::= { dot3 6 }
```

```
dot3Errors OBJECT IDENTIFIER ::= { dot3 7 }
```

#### -- TDR Test

```

-- The Time-Domain Reflectometry (TDR) test is specific
-- to ethernet-like interfaces with the exception of
-- 10BaseT and 10BaseF. The TDR value may be useful
-- in determining the approximate distance to a cable fault.
-- It is advisable to repeat this test to check for a
-- consistent resulting TDR value, to verify that there
-- is a fault.

```

```
dot3TestTdr OBJECT IDENTIFIER ::= { dot3Tests 1 }
```

```
-- A TDR test returns as its result the time interval,  
-- measured in 10 MHz ticks or 100 nsec units, between  
-- the start of TDR test transmission and the subsequent  
-- detection of a collision or deassertion of carrier. On  
-- successful completion of a TDR test, the result is  
-- stored as the value of the appropriate instance of the  
-- MIB object dot3TestTdrValue, and the OBJECT IDENTIFIER  
-- of that instance is stored in the corresponding instance  
-- of ifExtnsTestCode (thereby indicating where the  
-- result has been stored).
```

```
-- Loopback Test
```

```
-- Another test is the full-duplex loopback test.  
-- This test configures the MAC chip and executes  
-- an internal loopback test of memory, data paths,  
-- and the MAC chip logic. This loopback test can  
-- only be executed if the interface is offline.  
-- Once the test has completed, the MAC chip should  
-- be reinitialized for network operation, but it  
-- should remain offline.
```

```
dot3TestLoopBack OBJECT IDENTIFIER ::= { dot3Tests 2 }
```

```
-- If an error occurs during a test, the object  
-- ifTestResult (defined in RFC1573) will be set  
-- to failed(7). The following two OBJECT  
-- IDENTIFIERS may be used to provide more  
-- information as values for ifTestCode.
```

```
        -- couldn't initialize MAC chip for test  
dot3ErrorInitError      OBJECT IDENTIFIER ::= { dot3Errors 1 }
```

```
        -- expected data not received (or not  
        -- received correctly) in loopback test  
dot3ErrorLoopbackError OBJECT IDENTIFIER ::= { dot3Errors 2 }
```

```
-- RFC1573 does away with the interface chipset object.  
-- The following OBJECT IDENTIFIER definitions are  
-- retained for purposes of backwards compatibility  
-- with pre-RFC1573 systems.  
-- 802.3 Hardware Chipsets
```

```
-- The object ifExtnsChipSet is provided in RFC1229 to  
-- identify the MAC hardware used to communicate on an  
-- interface. The following hardware chipsets are  
-- provided for 802.3:
```

```

dot3ChipSets          OBJECT IDENTIFIER ::= { dot3 8 }
dot3ChipSetAMD        OBJECT IDENTIFIER ::= { dot3ChipSets 1 }
dot3ChipSetAMD7990    OBJECT IDENTIFIER ::= { dot3ChipSetAMD 1 }
dot3ChipSetAMD79900   OBJECT IDENTIFIER ::= { dot3ChipSetAMD 2 }
dot3ChipSetAMD79C940  OBJECT IDENTIFIER ::= { dot3ChipSetAMD 3 }

dot3ChipSetIntel      OBJECT IDENTIFIER ::= { dot3ChipSets 2 }
dot3ChipSetIntel82586 OBJECT IDENTIFIER ::= { dot3ChipSetIntel 1 }
dot3ChipSetIntel82596 OBJECT IDENTIFIER ::= { dot3ChipSetIntel 2 }

dot3ChipSetSeeq       OBJECT IDENTIFIER ::= { dot3ChipSets 3 }
dot3ChipSetSeeq8003   OBJECT IDENTIFIER ::= { dot3ChipSetSeeq 1 }

dot3ChipSetNational   OBJECT IDENTIFIER ::= { dot3ChipSets 4 }
dot3ChipSetNational8390 OBJECT IDENTIFIER ::=
    { dot3ChipSetNational 1 }
dot3ChipSetNationalSonic OBJECT IDENTIFIER ::=
    { dot3ChipSetNational 2 }

dot3ChipSetFujitsu     OBJECT IDENTIFIER ::= { dot3ChipSets 5 }
dot3ChipSetFujitsu86950 OBJECT IDENTIFIER ::=
    { dot3ChipSetFujitsu 1 }

dot3ChipSetDigital     OBJECT IDENTIFIER ::= { dot3ChipSets 6 }
dot3ChipSetDigitalDC21040 OBJECT IDENTIFIER ::=
    { dot3ChipSetDigital 1 }

-- For those chipsets not represented above, OBJECT IDENTIFIER
-- assignment is required in other documentation, e.g., assignment
-- within that part of the registration tree delegated to
-- individual enterprises (see RFC1155).

```

END

## 5. Acknowledgements

This document was produced by the Ethernet MIB Working Group.

This document is based on the Proposed Standard Ethernet MIB, RFC 1284 [14], of which Jihn Cook of Chipcom was the editor. The Ethernet MIB Working Group gathered implementation experience of the variables specified in RFC 1284 and used that information to develop this revised MIB.

RFC 1284, in turn, is based on a document written by Frank Kastenholz of Interlan entitled IEEE 802.3 Layer Management Draft M compatible MIB for TCP/IP Networks [10]. This document has been modestly reworked, initially by the SNMP Working Group, and then by the



Transmission Working Group, to reflect the current conventions for defining objects for MIB interfaces. James Davin, of the MIT Laboratory for Computer Science, and Keith McCloghrie of Hughes LAN Systems, contributed to later drafts of this memo. Marshall Rose of Performance Systems International, Inc. converted the document into its current concise format. Anil Rijasinghani of DEC contributed text that more adequately describes the TDR test. Thanks to Frank Kastenholz of Interlan and Louis Steinberg of IBM for their experimentation.

## 6. References

- [1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, NRI, April 1988.
- [2] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, NRI, August 1989.
- [3] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", STD 16, RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.
- [4] McCloghrie K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1156, Hughes LAN Systems, Performance Systems International, May 1990.
- [5] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.
- [6] McCloghrie K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets", STD 17, RFC 1213, Performance Systems International, March 1991.
- [7] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization, International Standard 8824, December 1987.
- [8] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization, International Standard 8825, December 1987.
- [9] IEEE, "IEEE 802.3 Layer Management", November 1988.

- [10] Kastenholz, F., "IEEE 802.3 Layer Management Draft compatible MIB for TCP/IP Networks", electronic mail message to mib-wg@nnsf.net, 9 June 1989.
- [11] McCloghrie, K., Editor, "Extensions to the Generic-Interface MIB", RFC 1229, Hughes LAN Systems, Inc., May 1991.
- [12] IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", ANSI/IEEE Std 802.3-1985.
- [13] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.
- [14] Cook, J., Editor, "Definitions of Managed Objects for Ethernet-Like Interface Types", RFC 1284, Chipcom Corporation, December 1991.
- [15] Kastenholz, F., "Definitions of Managed Objects for the Ethernet-like Interface Types", RFC 1398, FTP Software, Inc., January 1993.
- [16] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1442, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
- [17] Galvin, J., and K. McCloghrie, "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1445, Trusted Information Systems, Hughes LAN Systems, April 1993.
- [18] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1448, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
- [19] McCloghrie, K., and F. Kastenholz, "Evolution of the Interfaces Group of MIB-II", RFC 1573, Hughes LAN Systems, FTP Software, January 1994.

## 7. Security Considerations

Security issues are not discussed in this memo.

## 8. Author's Address

Frank Kastenholz  
FTP Software, Inc.  
2 High Street  
North Andover, Mass, USA 01845

Phone: 508-685-4000  
EMail: kasten@ftp.com

