

Definitions of Managed Objects for IEEE 802.3 Repeater Devices

Status of this Memo

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

Abstract

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 IEEE 802.3 10 Mb/second baseband repeaters, sometimes referred to as "hubs."

Table of Contents

1. Management Framework	2
2. Objects	2
2.1 Format of Definitions	3
3. Overview	3
3.1 Terminology	3
3.1.1 Repeaters, Hubs and Concentrators	3
3.1.2 Repeaters, Ports, and MAUs	4
3.1.3 Ports and Groups	6
3.2 Supporting Functions	7
3.3 Structure of MIB	9
3.3.1 The Basic Group Definitions	10
3.3.2 The Monitor Group Definitions	10
3.3.3 The Address Tracking Group Definitions	10
3.4 Relationship to Other MIBs	10
3.4.1 Relationship to the 'system' group	10
3.4.2 Relationship to the 'interfaces' group	10
3.5 Textual Conventions	11
4. Definitions	11
4.1 MIB Groups in the Repeater MIB	12
4.2 The Basic Group Definitions	13
4.3 The Monitor Group Definitions	23
4.4 The Address Tracking Group Definitions	33

4.5 Traps for use by Repeaters	35
5. Acknowledgments	37
6. References	39
7. Security Considerations.....	40
8. Authors' Addresses.....	40

1. Management Framework

The Internet-standard Network Management Framework consists of three components. They are:

STD 16/RFC 1155 [1] which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management. STD 16/RFC 1212 [7] defines a more concise description mechanism, which is wholly consistent with the SMI.

RFC 1156 [2] which defines MIB-I, the core set of managed objects for the Internet suite of protocols. STD 17/RFC 1213 [4] defines MIB-II, an evolution of MIB-I based on implementation experience and new operational requirements.

STD 15/RFC 1157 [3] 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.

2. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [5] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

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

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the

notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [6], subject to the additional requirements imposed by the SNMP.

2.1. Format of Definitions

Section 4 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [7,8].

3. Overview

Instances of the object types defined in this memo represent attributes of an IEEE 802.3 (Ethernet-like) repeater, as defined by Section 9, "Repeater Unit for 10 Mb/s Baseband Networks" in the IEEE 802.3/ISO 8802-3 CSMA/CD standard [9].

These Repeater MIB objects may be used to manage non-standard repeater-like devices, but defining objects to describe implementation-specific properties of non-standard repeater-like devices is outside the scope of this memo.

The definitions presented here are based on the IEEE draft standard P802.3K, "Layer Management for 10 Mb/s Baseband Repeaters." [10] Implementors of these MIB objects should note that [10] explicitly describes when, where, and how various repeater attributes are measured. The IEEE document also describes the effects of repeater actions that may be invoked by manipulating instances of the MIB objects defined here.

The counters in this document are defined to be the same as those counters in the IEEE 802.3 Repeater Management draft, with the intention that a single instrumentation can be used to implement both the IEEE and IETF management standards.

3.1. Terminology

3.1.1. Repeaters, Hubs and Concentrators

In late 1988, the IEEE 802.3 Hub Management task force was chartered to define managed objects for both 802.3 repeaters and the proposed 10BASE-FA synchronous active stars. The term "hub" was used to cover both repeaters and active stars.

In March, 1991, the active star proposal was dropped from the 10BASE-F draft. Subsequently the 802.3 group changed the name of the

task force to be the IEEE 802.3 Repeater Management Task Force, and likewise renamed their draft.

The use of the term "hub" has led to some confusion, as the terms "hub," "intelligent hub," and "concentrator" are often used to indicate a modular chassis with plug-in modules that provide generalized LAN/WAN connectivity, often with a mix of 802.3 repeater, token ring, and FDDI connectivity, internetworked by bridges, routers, and terminal servers.

To be clear that this work covers the management of IEEE 802.3 repeaters only, the editors of this MIB definitions document chose to call this a "Repeater MIB" instead of a "Hub MIB."

3.1.2. Repeaters, Ports, and MAUs

The following text roughly defines the terms "repeater," "port," and "MAU" as used in the context of this memo. This text is imprecise and omits many technical details. For a more complete and precise definition of these terms, refer to Section 9 of [9].

An IEEE 802.3 repeater connects "Ethernet-like" media segments together to extend the network length and topology beyond what can be achieved with a single coax segment. It can be pictured as a star structure with two or more input/output ports. The diagram below illustrates a 6-port repeater:

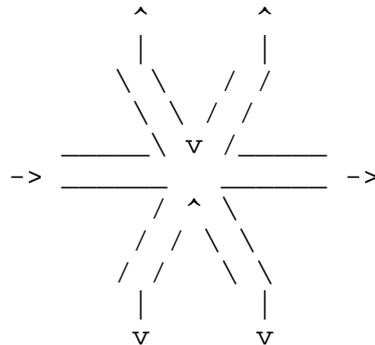


Figure 1. Repeater Unit

All the stations on the media segments connected to a given repeater's ports participate in a single collision domain. A packet transmitted by any of these stations is seen by all of these stations.

Data coming in on any port in the repeater is transmitted out through

each of the remaining n-1 ports. If data comes in to the repeater on two or more ports simultaneously or the repeater detects a collision on the incoming port, the repeater transmits a jamming signal out on all ports for the duration of the collision.

A repeater is a bit-wise store-and-forward device. It is differentiated from a bridge (a frame store-and-forward device) in that it is primarily concerned with carrier sense and data bits, and does not make data-handling decisions based on the legality or contents of a packet. A repeater retransmits data bits as they are received. Its data FIFO holds only enough bits to make sure that the FIFO does not underflow when the data rate of incoming bits is slightly slower than the repeater's transmission rate.

A repeater is not an end-station on the network, and does not count toward the overall limit of 1024 stations. A repeater has no MAC address associated with it, and therefore packets may not be addressed to the repeater or to its ports. (Packets may be addressed to the MAC address of a management entity that is monitoring a repeater. This management entity may or may not be connected to the network through one of the repeater's ports. How the management entity obtains information about the activity on the repeater is an implementation issue, and is not discussed in this memo.)

A repeater is connected to the network with Medium Attachment Units (MAUs), and sometimes through Attachment Unit Interfaces (AUIs) as well. ("MAUs" are also known as transceivers, and an "AUI" is the same as a 15-pin Ethernet or DIX connector.)

The 802.3 standard defines a "repeater set" as the "repeater unit" plus its associated MAUs (and AUIs if present). The "repeater unit" is defined as the portion of the repeater set that is inboard of the physical media interfaces. The MAUs may be physically separate from the repeater unit, or they may be integrated into the same physical package.

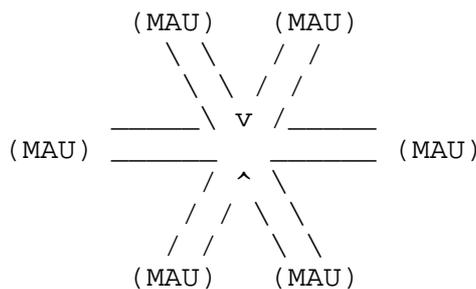


Figure 2. Repeater Set

The most commonly-used MAUs are the 10BASE-5 (AUI to thick "yellow" coax), 10BASE-2 (BNC to thin coax), 10BASE-T (unshielded twisted-pair), and FOIRL (asynchronous fiber optic inter-repeater link, which is being combined into the 10BASE-F standard as 10BASE-FL). The draft 10BASE-F standard also includes the definition for a new synchronous fiber optic attachment, known as 10BASE-FB.

It should be stressed that the repeater MIB being defined by the IEEE covers only the repeater unit management - it does not include management of the MAUs that form the repeater set. The IEEE recognizes that MAU management should be the same for MAUs connected to end-stations (DTEs) as it is for MAUs connected to repeaters. This memo follows the same strategy; the definition of management information for MAUs is being addressed in a separate memo.

3.1.3. Ports and Groups

Repeaters are often implemented in modular "concentrators," where a card cage holds several field-replaceable cards. Several cards may form a single repeater unit, with each card containing one or more of the repeater's ports. Because of this modular architecture, users typically identify these repeater ports with a card number plus the port number relative to the card, e.g., Card 3, Port 11.

To support this modular numbering scheme, this document follows the example of the IEEE Repeater Management draft [10], allowing an implementor to separate the ports in a repeater into "groups", if desired. For example, an implementor might choose to represent field-replaceable units as groups of ports so that the port numbering would match the modular hardware implementation.

This group mapping is recommended but optional. An implementor may choose to put all of a modular repeater's ports into a single group, or to divide the ports into groups that do not match physical divisions.

The object `rptrGroupCapacity`, which has a maximum value of 1024, indicates the maximum number of groups that a given repeater may contain. The value of `rptrGroupCapacity` must remain constant from one management restart to the next.

Each group within the repeater is uniquely identified by a group number in the range `1..rptrGroupCapacity`. Groups may come and go without causing a management reset, and may be sparsely numbered within the repeater. For example, in a 12-card cage, cards 3, 5, 6, and 7 may together form a single repeater, and the implementor may choose to number them as groups 3, 5, 6, and 7, respectively.

The object `rpPtrGroupPortCapacity`, which also has a maximum value of 1024, indicates the maximum number of ports that a given group may contain. The value of `rpPtrGroupPortCapacity` must not change for a given group. However, a group may be deleted from the repeater and replaced with a group containing a different number of ports. The value of `rpPtrGroupLastOperStatusChange` will indicate that a change took place.

Each port within the repeater is uniquely identified by a combination of group number and port number, where port number is an integer in the range `1..rpPtrGroupPortCapacity`. As with groups within a repeater, ports within a group may be sparsely numbered. Likewise, ports may come and go within a group without causing a management reset.

3.2. Supporting Functions

The IEEE 802.3 Hub Management draft [10] defines the following seven functions and seven signals used to describe precisely when port counters are incremented. The relationship between the functions and signals is shown in Figure 3.

The `CollisionEvent`, `ActivityDuration`, `CarrierEvent`, `FramingError`, `OctetCount`, `FCSError`, and `SourceAddress` output signals defined here are not retrievable MIB objects, but rather are concepts used in defining the MIB objects. The inputs are defined in Section 9 of the IEEE 802.3 standard [9].

signal is asserted. The framing function strips preamble and start of frame delimiter from the received data stream. The remaining bits are aligned along octet boundaries. If there is not an integral number of octets, then FramingError shall be asserted. The FramingError signal is cleared upon the assertion of the CarrierEvent signal due to the reception of the following event.

Activity Timing Function: The activity timing function measures the duration of the assertion of the CarrierEvent signal. This duration value must be adjusted by removing the value of Carrier Recovery Time (Ref: 9.5.6.5 [9]) to obtain the true duration of activity on the network. The output of the Activity Timing function is the ActivityDuration value, which represents the duration of the CarrierEvent signal as expressed in units of bit times.

Octet Counting Function: The octet counting function counts the number of complete octets received from the output of the framing function. The output of the octet counting function is the OctetCount value. The OctetCount value is reset to zero upon the assertion of the CarrierEvent signal due to the reception of the following event.

Cyclic Redundancy Check Function: The cyclic redundancy check function verifies that the sequence of octets output by the framing function contains a valid frame check sequence field. The frame check sequence field is the last four octets received from the output of the framing function. The algorithm for generating an FCS from the octet stream is specified in 3.2.8 [9]. If the FCS generated according to this algorithm is not the same as the last four octets received from the framing function then the FCSError signal is asserted. The FCSError signal is cleared upon the assertion of the CarrierEvent signal due to the reception of the following event.

Source Address Function: The source address function extracts octets from the stream output by the framing function. The seventh through twelfth octets shall be extracted from the octet stream and output as the SourceAddress variable. The SourceAddress variable is set to an invalid state upon the assertion of the CarrierEvent signal due to the reception of the following event.

3.3. Structure of MIB

Objects in this MIB are arranged into MIB groups. Each MIB group is organized as a set of related objects.

3.3.1. The Basic Group Definitions

This mandatory group contains the objects which are applicable to all repeaters. It contains status, parameter and control objects for the repeater as a whole, the port groups within the repeater, as well as for the individual ports themselves.

3.3.2. The Monitor Group Definitions

This optional group contains monitoring statistics for the repeater as a whole and for individual ports.

3.3.3. The Address Tracking Group Definitions

This optional group contains objects for tracking the MAC addresses of the DTEs attached to the ports of the repeater.

3.4. Relationship to Other MIBs

It is assumed that a repeater implementing this MIB will also implement (at least) the 'system' group defined in MIB-II [4].

3.4.1. Relationship to the 'system' group

In MIB-II, the 'system' group is defined as being mandatory for all systems such that each managed entity contains one instance of each object in the 'system' group. Thus, those objects apply to the entity even if the entity's sole functionality is management of a repeater.

3.4.2. Relationship to the 'interfaces' group

In MIB-II, the 'interfaces' group is defined as being mandatory for all systems and contains information on an entity's interfaces, where each interface is thought of as being attached to a 'subnetwork'. (Note that this term is not to be confused with 'subnet' which refers to an addressing partitioning scheme used in the Internet suite of protocols.)

This Repeater MIB uses the notion of ports on a repeater. The concept of a MIB-II interface has NO specific relationship to a repeater's port. Therefore, the 'interfaces' group applies only to the one (or more) network interfaces on which the entity managing the repeater sends and receives management protocol operations, and does not apply to the repeater's ports.

This is consistent with the physical-layer nature of a repeater. A repeater is a bitwise store-and-forward device. It recognizes

activity and bits, but does not process incoming data based on any packet-related information (such as checksum or addresses). A repeater has no MAC address, no MAC implementation, and does not pass packets up to higher-level protocol entities for processing.

(When a network management entity is observing the repeater, it may appear as though the repeater is passing packets to a higher-level protocol entity. However, this is only a means of implementing management, and this passing of management information is not part of the repeater functionality.)

3.5. Textual Conventions

The datatype MacAddress is used as a textual convention in this document. This textual convention has NO effect on either the syntax nor the semantics of any managed object. Objects defined using this convention are always encoded by means of the rules that define their primitive type. Hence, no changes to the SMI or the SNMP are necessary to accommodate this textual convention which is adopted merely for the convenience of readers.

4. Definitions

```
SNMP-REPEATER-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    Counter, TimeTicks, Gauge
```

```
    FROM RFC1155-SMI
```

```
    mib-2, DisplayString
```

```
    FROM RFC1213-MIB
```

```
    TRAP-TYPE
```

```
    FROM RFC-1215
```

```
    OBJECT-TYPE
```

```
    FROM RFC-1212;
```

```
snmpDot3RptrMgt OBJECT IDENTIFIER ::= { mib-2 22 }
```

```
-- All representations of MAC addresses in this MIB Module use,
-- as a textual convention (i.e., this convention does not affect
-- their encoding), the data type:
```

```
MacAddress ::= OCTET STRING (SIZE (6))    -- a 6 octet address in
                                           -- the "canonical" order
-- defined by IEEE 802.1a, i.e., as if it were transmitted least
-- significant bit first.
```

References

```
--
--
-- The following references are used throughout this MIB:
--
-- [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
--   (2nd edition, September 21, 1990).
--
-- [IEEE 802.3 Rptr Mgt]
--   refers to IEEE P802.3K, 'Layer Management for 10 Mb/s
--   Baseband Repeaters, Section 19,' Draft Supplement to
--   ANSI/IEEE 802.3, (Draft 8, April 9, 1992)
```

MIB Groups

```
--
--
-- The rptrBasicPackage group is mandatory.
-- The rptrMonitorPackage and rptrAddrTrackPackage
-- groups are optional.
```

```
rptrBasicPackage
  OBJECT IDENTIFIER ::= { snmpDot3RptrMgt 1 }
```

```
rptrMonitorPackage
  OBJECT IDENTIFIER ::= { snmpDot3RptrMgt 2 }
```

```
rptrAddrTrackPackage
  OBJECT IDENTIFIER ::= { snmpDot3RptrMgt 3 }
```

```
-- object identifiers for organizing the information
-- in the groups by repeater, port-group, and port
```

```
rptrRptrInfo
  OBJECT IDENTIFIER ::= { rptrBasicPackage 1 }
```

```
rptrGroupInfo
  OBJECT IDENTIFIER ::= { rptrBasicPackage 2 }
```

```
rptrPortInfo
  OBJECT IDENTIFIER ::= { rptrBasicPackage 3 }
```

```
rptrMonitorRptrInfo
  OBJECT IDENTIFIER ::= { rptrMonitorPackage 1 }
```

```
rptrMonitorGroupInfo
  OBJECT IDENTIFIER ::= { rptrMonitorPackage 2 }
```

```

rpPtrMonitorPortInfo
    OBJECT IDENTIFIER ::= { rpPtrMonitorPackage 3 }

rpPtrAddrTrackRpPtrInfo      -- this subtree is currently unused
    OBJECT IDENTIFIER ::= { rpPtrAddrTrackPackage 1 }
rpPtrAddrTrackGroupInfo     -- this subtree is currently unused
    OBJECT IDENTIFIER ::= { rpPtrAddrTrackPackage 2 }
rpPtrAddrTrackPortInfo
    OBJECT IDENTIFIER ::= { rpPtrAddrTrackPackage 3 }

--
--                               The BASIC GROUP
--
-- Implementation of the Basic Group is mandatory for all
-- managed repeaters.

--
-- Basic Repeater Information
--
-- Configuration, status, and control objects for the overall
-- repeater
--

rpPtrGroupCapacity OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The rpPtrGroupCapacity is the number of groups
        that can be contained within the repeater. Within
        each managed repeater, the groups are uniquely
        numbered in the range from 1 to rpPtrGroupCapacity.

        Some groups may not be present in the repeater, in
        which case the actual number of groups present
        will be less than rpPtrGroupCapacity. The number
        of groups present will never be greater than
        rpPtrGroupCapacity.

        Note: In practice, this will generally be the
        number of field-replaceable units (i.e., modules,
        cards, or boards) that can fit in the physical
        repeater enclosure, and the group numbers will
        correspond to numbers marked on the physical
        enclosure."

REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.3.2,
    aRepeaterGroupCapacity."

```

```
::= { rpPtrRptrInfo 1 }
```

```
rpPtrOperStatus OBJECT-TYPE
```

```
SYNTAX INTEGER {
    other(1),           -- undefined or unknown status
    ok(2),             -- no known failures
    rpPtrFailure(3),   -- repeater-related failure
    groupFailure(4),   -- group-related failure
    portFailure(5),    -- port-related failure
    generalFailure(6)  -- failure, unspecified type
}
```

```
ACCESS read-only
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"The rpPtrOperStatus object indicates the operational state of the repeater. The rpPtrHealthText object may be consulted for more specific information about the state of the repeater's health.

In the case of multiple kinds of failures (e.g., repeater failure and port failure), the value of this attribute shall reflect the highest priority failure in the following order:

```
rpPtrFailure(3)
groupFailure(4)
portFailure(5)
generalFailure(6)."
```

```
REFERENCE
```

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.2, aRepeaterHealthState."

```
::= { rpPtrRptrInfo 2 }
```

```
rpPtrHealthText OBJECT-TYPE
```

```
SYNTAX DisplayString (SIZE (0..255))
```

```
ACCESS read-only
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"The health text object is a text string that provides information relevant to the operational state of the repeater. Agents may use this string to provide detailed information on current failures, including how they were detected, and/or instructions for problem resolution. The contents are agent-specific."

```
REFERENCE
```

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.2,

```

        aRepeaterHealthText."
 ::= { rpTrRpTrInfo 3 }

```

rpTrReset OBJECT-TYPE

```

SYNTAX      INTEGER {
                noReset(1),
                reset(2)
            }

```

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Setting this object to reset(2) causes a transition to the START state of Fig 9-2 in section 9 [IEEE 802.3 Std].

Setting this object to noReset(1) has no effect. The agent will always return the value noReset(1) when this object is read.

This action does not reset the management counters defined in this document nor does it affect the portAdminStatus parameters. Included in this action is the execution of a disruptive Self-Test with the following characteristics: a) The nature of the tests is not specified. b) The test resets the repeater but without affecting management information about the repeater. c) The test does not inject packets onto any segment. d) Packets received during the test may or may not be transferred. e) The test does not interfere with management functions.

As a result of this action a rpTrResetEvent trap should be sent."

REFERENCE

"Reference IEEE 802.3 RpTr Mgt, 19.2.3.3, acResetRepeater."

```

 ::= { rpTrRpTrInfo 4 }

```

rpTrNonDisruptTest OBJECT-TYPE

```

SYNTAX      INTEGER {
                noSelfTest(1),
                selfTest(2)
            }

```

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Setting this object to selfTest(2) causes the

repeater to perform a agent-specific, non-disruptive self-test that has the following characteristics: a) The nature of the tests is not specified. b) The test does not change the state of the repeater or management information about the repeater. c) The test does not inject packets onto any segment. d) The test does not prevent the relay of any packets. e) The test does not interfere with management functions.

After performing this test the agent will update the repeater health information and send a rpTrHealth trap.

Setting this object to noSelfTest(1) has no effect. The agent will always return the value noSelfTest(1) when this object is read."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.3, acExecuteNonDisruptiveSelfTest."

::= { rpTrRptrInfo 5 }

rpTrTotalPartitionedPorts OBJECT-TYPE

SYNTAX Gauge

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This object returns the total number of ports in the repeater whose current state meets all three of the following criteria: rpTrPortOperStatus does not have the value notPresent(3), rpTrPortAdminStatus is enabled(1), and rpTrPortAutoPartitionState is autoPartitioned(2)."

::= { rpTrRptrInfo 6 }

--

-- The Basic Port Group Table

--

rpTrGroupTable OBJECT-TYPE

SYNTAX SEQUENCE OF RpTrGroupEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"Table of descriptive and status information about the groups of ports."

::= { rpTrGroupInfo 1 }

```

rpPtrGroupEntry OBJECT-TYPE
    SYNTAX      RpPtrGroupEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "An entry in the table, containing information
         about a single group of ports."
    INDEX       { rpPtrGroupIndex }
    ::= { rpPtrGroupTable 1 }

RpPtrGroupEntry ::=
    SEQUENCE {
        rpPtrGroupIndex
            INTEGER,
        rpPtrGroupDescr
            DisplayString,
        rpPtrGroupObjectID
            OBJECT IDENTIFIER,
        rpPtrGroupOperStatus
            INTEGER,
        rpPtrGroupLastOperStatusChange
            TimeTicks,
        rpPtrGroupPortCapacity
            INTEGER
    }

rpPtrGroupIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This object identifies the group within the
         repeater for which this entry contains
         information. This value is never greater than
         rpPtrGroupCapacity."
    REFERENCE
        "Reference IEEE 802.3 RpPtr Mgt, 19.2.5.2,
         aGroupID."
    ::= { rpPtrGroupEntry 1 }

rpPtrGroupDescr OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "A textual description of the group. This value
         should include the full name and version
         identification of the group's hardware type and

```

indicate how the group is differentiated from other groups in the repeater. Plug-in Module, Rev A' or 'Barney Rubble 10BASE-T 4-port SIMM socket Version 2.1' are examples of valid group descriptions.

It is mandatory that this only contain printable ASCII characters."

```
::= { rptrGroupEntry 2 }
```

rpPtrGroupObjectID OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The vendor's authoritative identification of the group. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides a straight-forward and unambiguous means for determining what kind of group is being managed.

For example, this object could take the value 1.3.6.1.4.1.4242.1.2.14 if vendor 'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, and had assigned the identifier 1.3.6.1.4.1.4242.1.2.14 to its 'Wilma Flintstone 6-Port FOIRL Plug-in Module.'"

```
::= { rpPtrGroupEntry 3 }
```

rpPtrGroupOperStatus OBJECT-TYPE

```
SYNTAX INTEGER {
    other(1),
    operational(2),
    malfunctioning(3),
    notPresent(4),
    underTest(5),
    resetInProgress(6)
}
```

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An object that indicates the operational status of the group.

A status of notPresent(4) indicates that the group is temporarily or permanently physically and/or logically not a part of the repeater. It is an implementation-specific matter as to whether the

agent effectively removes notPresent entries from the table.

A status of operational(2) indicates that the group is functioning, and a status of malfunctioning(3) indicates that the group is malfunctioning in some way."

::= { rpPtrGroupEntry 4 }

rpPtrGroupLastOperStatusChange OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An object that contains the value of sysUpTime at the time that the value of the rpPtrGroupOperStatus object for this group last changed.

A value of zero indicates that the group's oper status has not changed since the agent last restarted."

::= { rpPtrGroupEntry 5 }

rpPtrGroupPortCapacity OBJECT-TYPE

SYNTAX INTEGER (1..1024)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The rpPtrGroupPortCapacity is the number of ports that can be contained within the group. Valid range is 1-1024. Within each group, the ports are uniquely numbered in the range from 1 to rpPtrGroupPortCapacity.

Note: In practice, this will generally be the number of ports on a module, card, or board, and the port numbers will correspond to numbers marked on the physical embodiment."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.5.2, aGroupPortCapacity."

::= { rpPtrGroupEntry 6 }

```
--
-- The Basic Port Table
--

rptrPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF RptrPortEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Table of descriptive and status information about
         the ports."
    ::= { rptrPortInfo 1 }

rptrPortEntry OBJECT-TYPE
    SYNTAX      RptrPortEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "An entry in the table, containing information
         about a single port."
    INDEX      { rptrPortGroupIndex, rptrPortIndex }
    ::= { rptrPortTable 1 }

RptrPortEntry ::=
    SEQUENCE {
        rptrPortGroupIndex
            INTEGER,
        rptrPortIndex
            INTEGER,
        rptrPortAdminStatus
            INTEGER,
        rptrPortAutoPartitionState
            INTEGER,
        rptrPortOperStatus
            INTEGER
    }

rptrPortGroupIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This object identifies the group containing the
         port for which this entry contains information."
    ::= { rptrPortEntry 1 }

rptrPortIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
```

```

ACCESS      read-only
STATUS      mandatory
DESCRIPTION
    "This object identifies the port within the group
    for which this entry contains information.  This
    value can never be greater than
    rptrGroupPortCapacity for the associated group."
REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
    aPortID."
 ::= { rptrPortEntry 2 }

```

rpPtrPortAdminStatus OBJECT-TYPE

```

SYNTAX      INTEGER {
                enabled(1),
                disabled(2)
            }

```

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Setting this object to disabled(2) disables the port. A disabled port neither transmits nor receives. Once disabled, a port must be explicitly enabled to restore operation. A port which is disabled when power is lost or when a reset is exerted shall remain disabled when normal operation resumes.

The admin status takes precedence over auto-partition and functionally operates between the auto-partition mechanism and the AUI/PMA.

Setting this object to enabled(1) enables the port and exerts a BEGIN on the port's auto-partition state machine.

(In effect, when a port is disabled, the value of rpPtrPortAutoPartitionState for that port is frozen until the port is next enabled. When the port becomes enabled, the rpPtrPortAutoPartitionState becomes notAutoPartitioned(1), regardless of its pre-disabling state.)"

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aPortAdminState and 19.2.6.3, acPortAdminControl."

```
 ::= { rptrPortEntry 3 }
```

rpPtrPortAutoPartitionState OBJECT-TYPE

```

SYNTAX      INTEGER {

```

```

        notAutoPartitioned(1),
        autoPartitioned(2)
    }
ACCESS    read-only
STATUS    mandatory
DESCRIPTION
    "The autoPartitionState flag indicates whether the
    port is currently partitioned by the repeater's
    auto-partition protection.

    The conditions that cause port partitioning are
    specified in partition state machine in Section 9
    [IEEE 802.3 Std]. They are not differentiated
    here."
REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
    aAutoPartitionState."
 ::= { rptrPortEntry 4 }

rptrPortOperStatus OBJECT-TYPE
SYNTAX    INTEGER {
            operational(1),
            notOperational(2),
            notPresent(3)
        }
ACCESS    read-only
STATUS    mandatory
DESCRIPTION
    "This object indicates the port's operational
    status. The notPresent(3) status indicates the
    port is physically removed (note this may or may
    not be possible depending on the type of port.)

    The operational(1) status indicates that the port
    is enabled (see rptrPortAdminStatus) and working,
    even though it might be auto-partitioned (see
    rptrPortAutoPartitionState).

    If this object has the value operational(1) and
    rptrPortAdminStatus is set to disabled(2), it is
    expected that this object's value will change to
    notOperational(2) soon after."
 ::= { rptrPortEntry 5 }

```

```

--
--                               The MONITOR GROUP
--
-- Implementation of this group is optional, but within the
-- group all elements are mandatory.  If a managed repeater
-- implements any part of this group, the entire group shall
-- be implemented.
--
--
-- Repeater Monitor Information
--
-- Performance monitoring statistics for the repeater
--
rptrMonitorTransmitCollisions OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This counter is incremented every time the
        repeater state machine enters the TRANSMIT
        COLLISION state from any state other than ONE PORT
        LEFT (Ref: Fig 9-2, IEEE 802.3 Std).

        The approximate minimum time for rollover of this
        counter is 16 hours."
    REFERENCE
        "Reference IEEE 802.3 Rptr Mgt, 19.2.3.2,
        aTransmitCollisions."
    ::= { rptrMonitorRptrInfo 1 }

--
-- The Group Monitor Table
--
rptrMonitorGroupTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF RptrMonitorGroupEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Table of performance and error statistics for the
        groups."
    ::= { rptrMonitorGroupInfo 1 }

rptrMonitorGroupEntry OBJECT-TYPE
    SYNTAX      RptrMonitorGroupEntry

```

ACCESS not-accessible
 STATUS mandatory
 DESCRIPTION

"An entry in the table, containing total performance and error statistics for a single group. Regular retrieval of the information in this table provides a means of tracking the performance and health of the networked devices attached to this group's ports.

The counters in this table are redundant in the sense that they are the summations of information already available through other objects. However, these sums provide a considerable optimization of network management traffic over the otherwise necessary retrieval of the individual counters included in each sum."

INDEX { rpPtrMonitorGroupIndex }
 ::= { rpPtrMonitorGroupTable 1 }

RpPtrMonitorGroupEntry ::=
 SEQUENCE {
 rpPtrMonitorGroupIndex
 INTEGER,
 rpPtrMonitorGroupTotalFrames
 Counter,
 rpPtrMonitorGroupTotalOctets
 Counter,
 rpPtrMonitorGroupTotalErrors
 Counter
 }

rpPtrMonitorGroupIndex OBJECT-TYPE
 SYNTAX INTEGER (1..1024)
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "This object identifies the group within the
 repeater for which this entry contains
 information."
 ::= { rpPtrMonitorGroupEntry 1 }

rpPtrMonitorGroupTotalFrames OBJECT-TYPE
 SYNTAX Counter
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "The total number of frames of valid frame length

that have been received on the ports in this group. This counter is the summation of the values of the rpPtrMonitorPortReadableFrames counters for all of the ports in the group.

This statistic provides one of the parameters necessary for obtaining the packet error rate. The approximate minimum time for rollover of this counter is 80 hours."

```
::= { rpPtrMonitorGroupEntry 2 }
```

```
rpPtrMonitorGroupTotalOctets OBJECT-TYPE
```

```
SYNTAX Counter
```

```
ACCESS read-only
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"The total number of octets contained in the valid frames that have been received on the ports in this group. This counter is the summation of the values of the rpPtrMonitorPortReadableOctets counters for all of the ports in the group.

This statistic provides an indicator of the total data transferred. The approximate minimum time for rollover of this counter is 58 minutes."

```
::= { rpPtrMonitorGroupEntry 3 }
```

```
rpPtrMonitorGroupTotalErrors OBJECT-TYPE
```

```
SYNTAX Counter
```

```
ACCESS read-only
```

```
STATUS mandatory
```

```
DESCRIPTION
```

"The total number of errors which have occurred on all of the ports in this group. This counter is the summation of the values of the rpPtrMonitorPortTotalErrors counters for all of the ports in the group."

```
::= { rpPtrMonitorGroupEntry 4 }
```

```
--
```

```
-- The Port Monitor Table
```

```
--
```

```
rpPtrMonitorPortTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF RpPtrMonitorPortEntry
```

```
ACCESS not-accessible
```

```
STATUS mandatory
```

DESCRIPTION

"Table of performance and error statistics for the ports."

```
::= { rpPtrMonitorPortInfo 1 }
```

rpPtrMonitorPortEntry OBJECT-TYPE

SYNTAX RptrMonitorPortEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"An entry in the table, containing performance and error statistics for a single port."

INDEX { rpPtrMonitorPortGroupIndex, rpPtrMonitorPortIndex }

```
::= { rpPtrMonitorPortTable 1 }
```

RptrMonitorPortEntry ::=

SEQUENCE {

rpPtrMonitorPortGroupIndex

INTEGER,

rpPtrMonitorPortIndex

INTEGER,

rpPtrMonitorPortReadableFrames
Counter,

rpPtrMonitorPortReadableOctets
Counter,

rpPtrMonitorPortFCSErrors
Counter,

rpPtrMonitorPortAlignmentErrors
Counter,

rpPtrMonitorPortFrameTooLongs
Counter,

rpPtrMonitorPortShortEvents
Counter,

rpPtrMonitorPortRunts
Counter,

rpPtrMonitorPortCollisions
Counter,

rpPtrMonitorPortLateEvents
Counter,

rpPtrMonitorPortVeryLongEvents
Counter,

rpPtrMonitorPortDataRateMismatches
Counter,

rpPtrMonitorPortAutoPartitions
Counter,

rpPtrMonitorPortTotalErrors
Counter

}

```
rpPtrMonitorPortGroupIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This object identifies the group containing the
         port for which this entry contains information."
    ::= { rpPtrMonitorPortEntry 1 }

rpPtrMonitorPortIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..1024)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This object identifies the port within the group
         for which this entry contains information."
    REFERENCE
        "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
         aPortID."
    ::= { rpPtrMonitorPortEntry 2 }

rpPtrMonitorPortReadableFrames OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "This object is the number of frames of valid
         frame length that have been received on this port.
         This counter is incremented by one for each frame
         received on this port whose OctetCount is greater
         than or equal to minFrameSize and less than or
         equal to maxFrameSize (Ref: IEEE 802.3 Std,
         4.4.2.1) and for which the FCSError and
         CollisionEvent signals are not asserted.

         This statistic provides one of the parameters
         necessary for obtaining the packet error rate.
         The approximate minimum time for rollover of this
         counter is 80 hours."
    REFERENCE
        "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
         aReadableFrames."
    ::= { rpPtrMonitorPortEntry 3 }

rpPtrMonitorPortReadableOctets OBJECT-TYPE
    SYNTAX      Counter
    ACCESS      read-only
    STATUS      mandatory
```

DESCRIPTION

"This object is the number of octets contained in valid frames that have been received on this port. This counter is incremented by OctetCount for each frame received on this port which has been determined to be a readable frame.

This statistic provides an indicator of the total data transferred. The approximate minimum time for rollover of this counter is 58 minutes."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aReadableOctets."

::= { rptrMonitorPortEntry 4 }

rptrMonitorPortFCSErrors OBJECT-TYPE

SYNTAX Counter
ACCESS read-only
STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each frame received on this port with the FCSError signal asserted and the FramingError and CollisionEvent signals deasserted and whose OctetCount is greater than or equal to minFrameSize and less than or equal to maxFrameSize (Ref: 4.4.2.1, IEEE 802.3 Std).

The approximate minimum time for rollover of this counter is 80 hours."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aFrameCheckSequenceErrors."

::= { rptrMonitorPortEntry 5 }

rptrMonitorPortAlignmentErrors OBJECT-TYPE

SYNTAX Counter
ACCESS read-only
STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each frame received on this port with the FCSError and FramingError signals asserted and CollisionEvent signal deasserted and whose OctetCount is greater than or equal to minFrameSize and less than or equal to maxFrameSize (Ref: IEEE 802.3 Std, 4.4.2.1). If rptrMonitorPortAlignmentErrors is incremented then the rptrMonitorPortFCSErrors

Counter shall not be incremented for the same frame.

The approximate minimum time for rollover of this counter is 80 hours."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aAlignmentErrors."

::= { rptrMonitorPortEntry 6 }

rptrMonitorPortFrameTooLongs OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each frame received on this port whose OctetCount is greater than maxFrameSize (Ref: 4.4.2.1, IEEE 802.3 Std). If rptrMonitorPortFrameTooLongs is incremented then neither the rptrMonitorPortAlignmentErrors nor the rptrMonitorPortFCSErrors counter shall be incremented for the frame.

The approximate minimum time for rollover of this counter is 61 days."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aFramesTooLong."

::= { rptrMonitorPortEntry 7 }

rptrMonitorPortShortEvents OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each CarrierEvent on this port with ActivityDuration less than ShortEventMaxTime. ShortEventMaxTime is greater than 74 bit times and less than 82 bit times. ShortEventMaxTime has tolerances included to provide for circuit losses between a conformance test point at the AUI and the measurement point within the state machine.

Note: shortEvents may indicate externally generated noise hits which will cause the repeater to transmit Runtts to its other ports, or propagate a collision (which may be late) back to the

transmitting DTE and damaged frames to the rest of the network.

Implementors may wish to consider selecting the ShortEventMaxTime towards the lower end of the allowed tolerance range to accommodate bit losses suffered through physical channel devices not budgeted for within this standard.

The approximate minimum time for rollover of this counter is 16 hours."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aShortEvents."

::= { rptrMonitorPortEntry 8 }

rptrMonitorPortRunts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each CarrierEvent on this port that meets one of the following two conditions. Only one test need be made. a) The ActivityDuration is greater than ShortEventMaxTime and less than ValidPacketMinTime and the CollisionEvent signal is deasserted. b) The OctetCount is less than 64, the ActivityDuration is greater than ShortEventMaxTime and the CollisionEvent signal is deasserted. ValidPacketMinTime is greater than or equal to 552 bit times and less than 565 bit times.

An event whose length is greater than 74 bit times but less than 82 bit times shall increment either the shortEvents counter or the runts counter but not both. A CarrierEvent greater than or equal to 552 bit times but less than 565 bit times may or may not be counted as a runt.

ValidPacketMinTime has tolerances included to provide for circuit losses between a conformance test point at the AUI and the measurement point within the state machine.

Runts usually indicate collision fragments, a normal network event. In certain situations associated with large diameter networks a

percentage of runts may exceed ValidPacketMinTime.

The approximate minimum time for rollover of this counter is 16 hours."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aRunts."

::= { rpPtrMonitorPortEntry 9 }

rpPtrMonitorPortCollisions OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for any CarrierEvent signal on any port for which the CollisionEvent signal on this port is asserted.

The approximate minimum time for rollover of this counter is 16 hours."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aCollisions."

::= { rpPtrMonitorPortEntry 10 }

rpPtrMonitorPortLateEvents OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each CarrierEvent on this port in which the CollIn(X) variable transitions to the value SQE (Ref: 9.6.6.2, IEEE 802.3 Std) while the ActivityDuration is greater than the LateEventThreshold. Such a CarrierEvent is counted twice, as both a collision and as a lateEvent.

The LateEventThreshold is greater than 480 bit times and less than 565 bit times. LateEventThreshold has tolerances included to permit an implementation to build a single threshold to serve as both the LateEventThreshold and ValidPacketMinTime threshold.

The approximate minimum time for rollover of this counter is 81 hours."

REFERENCE

```

        "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
        aLateEvents."
 ::= { rptrMonitorPortEntry 11 }

```

rptrMonitorPortVeryLongEvents OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each CarrierEvent on this port whose ActivityDuration is greater than the MAU Jabber Lockup Protection timer TW3 (Ref: 9.6.1 & 9.6.5, IEEE 802.3 Std). Other counters may be incremented as appropriate."

REFERENCE

```

        "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
        aVeryLongEvents."
 ::= { rptrMonitorPortEntry 12 }

```

rptrMonitorPortDataRateMismatches OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each frame received on this port that meets all of the following conditions: a) The CollisionEvent signal is not asserted. b) The ActivityDuration is greater than ValidPacketMinTime. c) The frequency (data rate) is detectably mismatched from the local transmit frequency. The exact degree of mismatch is vendor specific and is to be defined by the vendor for conformance testing.

When this event occurs, other counters whose increment conditions were satisfied may or may not also be incremented, at the implementor's discretion. Whether or not the repeater was able to maintain data integrity is beyond the scope of this standard."

REFERENCE

```

        "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
        aDataRateMismatches."
 ::= { rptrMonitorPortEntry 13 }

```

rptrMonitorPortAutoPartitions OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"This counter is incremented by one for each time the repeater has automatically partitioned this port. The conditions that cause port partitioning are specified in the partition state machine in Section 9 [IEEE 802.3 Std]. They are not differentiated here."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.6.2, aAutoPartitions."

::= { rptrMonitorPortEntry 14 }

rptrMonitorPortTotalErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of errors which have occurred on this port. This counter is the summation of the values of other error counters (for the same port), namely:

rptrMonitorPortFCSErrors,
rptrMonitorPortAlignmentErrors,
rptrMonitorPortFrameTooLongs,
rptrMonitorPortShortEvents,
rptrMonitorPortLateEvents,
rptrMonitorPortVeryLongEvents, and
rptrMonitorPortDataRateMismatches.

This counter is redundant in the sense that it is the summation of information already available through other objects. However, it is included specifically because the regular retrieval of this object as a means of tracking the health of a port provides a considerable optimization of network management traffic over the otherwise necessary retrieval of the summed counters."

::= { rptrMonitorPortEntry 15 }

--

--

The ADDRESS TRACKING GROUP

--

-- Implementation of this group is optional; it is appropriate
-- for all systems which have the necessary metering. If a
-- managed repeater implements any part of this group, the entire

```
-- group shall be implemented.
```

```
--
```

```
-- The Port Address Tracking Table
```

```
--
```

```
rpPtrAddrTrackTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF RptrAddrTrackEntry
```

```
ACCESS not-accessible
```

```
STATUS mandatory
```

```
DESCRIPTION
```

```
"Table of address mapping information about the
ports."
```

```
::= { rpPtrAddrTrackPortInfo 1 }
```

```
rpPtrAddrTrackEntry OBJECT-TYPE
```

```
SYNTAX RptrAddrTrackEntry
```

```
ACCESS not-accessible
```

```
STATUS mandatory
```

```
DESCRIPTION
```

```
"An entry in the table, containing address mapping
information about a single port."
```

```
INDEX { rpPtrAddrTrackGroupIndex, rpPtrAddrTrackPortIndex }
```

```
::= { rpPtrAddrTrackTable 1 }
```

```
RptrAddrTrackEntry ::=
```

```
SEQUENCE {
```

```
rpPtrAddrTrackGroupIndex
```

```
INTEGER,
```

```
rpPtrAddrTrackPortIndex
```

```
INTEGER,
```

```
rpPtrAddrTrackLastSourceAddress
```

```
MacAddress,
```

```
rpPtrAddrTrackSourceAddrChanges
```

```
Counter
```

```
}
```

```
rpPtrAddrTrackGroupIndex OBJECT-TYPE
```

```
SYNTAX INTEGER (1..1024)
```

```
ACCESS read-only
```

```
STATUS mandatory
```

```
DESCRIPTION
```

```
"This object identifies the group containing the
port for which this entry contains information."
```

```
::= { rpPtrAddrTrackEntry 1 }
```

```
rpPtrAddrTrackPortIndex OBJECT-TYPE
```

```
SYNTAX INTEGER (1..1024)
```

```
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
    "This object identifies the port within the group
    for which this entry contains information."
REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
    aPortID."
 ::= { rptrAddrTrackEntry 2 }
```

```
rptrAddrTrackLastSourceAddress OBJECT-TYPE
SYNTAX      MacAddress
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
    "This object is the SourceAddress of the last
    readable frame (i.e., counted by
    rpTrMonitorPortReadableFrames) received by this
    port."
REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
    aLastSourceAddress."
 ::= { rptrAddrTrackEntry 3 }
```

```
rptrAddrTrackSourceAddrChanges OBJECT-TYPE
SYNTAX      Counter
ACCESS      read-only
STATUS      mandatory
DESCRIPTION
    "This counter is incremented by one for each time
    that the rpTrAddrTrackLastSourceAddress attribute
    for this port has changed.

    This may indicate whether a link is connected to a
    single DTE or another multi-user segment.
    The approximate minimum time for rollover of this
    counter is 81 hours."
REFERENCE
    "Reference IEEE 802.3 Rptr Mgt, 19.2.6.2,
    aSourceAddressChanges."
 ::= { rptrAddrTrackEntry 4 }
```

-- Traps for use by Repeaters

-- Traps are defined using the conventions in RFC 1215 [8].

```
rpTrHealth TRAP-TYPE
```

```
ENTERPRISE    snmpDot3RptrMgt
VARIABLES     { rptrOperStatus }
DESCRIPTION
```

"The rptrHealth trap conveys information related to the operational status of the repeater. This trap is sent only when the oper status of the repeater changes.

The rptrHealth trap must contain the rptrOperStatus object. The agent may optionally include the rptrHealthText object in the varBind list. See the rptrOperStatus and rptrHealthText objects for descriptions of the information that is sent.

The agent must throttle the generation of consecutive rptrHealth traps so that there is at least a five-second gap between them."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.4, hubHealth notification."

::= 1

rptrGroupChange TRAP-TYPE

```
ENTERPRISE    snmpDot3RptrMgt
VARIABLES     { rptrGroupIndex }
DESCRIPTION
```

"This trap is sent when a change occurs in the group structure of a repeater. This occurs only when a group is logically or physically removed from or added to a repeater. The varBind list contains the identifier of the group that was removed or added.

The agent must throttle the generation of consecutive rptrGroupChange traps for the same group so that there is at least a five-second gap between them."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.4, groupMapChange notification."

::= 2

rptrResetEvent TRAP-TYPE

```
ENTERPRISE    snmpDot3RptrMgt
VARIABLES     { rptrOperStatus }
DESCRIPTION
```

"The rptrResetEvent trap conveys information

related to the operational status of the repeater. This trap is sent on completion of a repeater reset action. A repeater reset action is defined as an a transition to the START state of Fig 9-2 in section 9 [IEEE 802.3 Std], when triggered by a management command (e.g., an SNMP Set on the rpPtrReset object).

The agent must throttle the generation of consecutive rpPtrResetEvent traps so that there is at least a five-second gap between them.

The rpPtrResetEvent trap is not sent when the agent restarts and sends an SNMP coldStart or warmStart trap. However, it is recommended that a repeater agent send the rpPtrOperStatus object as an optional object with its coldStart and warmStart trap PDUs.

The rpPtrOperStatus object must be included in the varbind list sent with this trap. The agent may optionally include the rpPtrHealthText object as well."

REFERENCE

"Reference IEEE 802.3 Rptr Mgt, 19.2.3.4, hubReset notification."

::= 3

END

5. Acknowledgments

This document is the work of the IETF Hub MIB Working Group. It is based on drafts of the IEEE 802.3 Repeater Management Task Force. Members of the working group included:

Karl Auerbach	karl@eng.sun.com
Jim Barnes	barnes@xylogics.com
Steve Bostock	steveb@novell.com
David Bridgham	dab@asylum.sf.ca.us
Jack Brown	jbrown@huahuca-emh8.army.mil
Howard Brown	brown@ctrn.com
Lida Canin	lida@apple.com
Jeffrey Case	case@cs.utk.edu
Carson Cheung	carson@bnr.com.ca
James Codespote	jpcodes@tycho.ncsc.mil
John Cook	cook@chipcom.com
Dave Cullerot	cullerot@ctrn.com

James Davin	jrd@ptt.lcs.mit.edu
Gary Ellis	garye@hpspd.spd.hp.com
David Engel	david@cds.com
Mike Erlinger	mike@mti.com
Jeff Erwin	
Bill Fardy	fardy@ctron.com
Jeff Fried	jmf@relay.proteon.com
Bob Friesenhahn	pdrusa!bob@uunet.uu.net
Shawn Gallagher	gallagher@quiver.enet.dec.com
Mike Grieves	mgrieves@chipcom.com
Walter Guilarte	70026.1715@compuserve.com
Phillip Hasse	phasse@honchuca-emh8.army.mil
Mark Hoerth	mark_hoerth@hp0400.desk.hp.com
Greg Hollingsworth	greg@mailer.jhuapl.edu
Ron Jacoby	rj@sgi.com
Mike Janson	mjanson@mot.com
Ken Jones	konkord!ksj@uunet.uu.net
Satish Joshi	sjoshi@synoptics.com
Frank Kastenholz	kasten@europa.clearpoint.com
Manu Kaycee	kaycee@trlan.enet.dec.com
Mark Kepke	mak@cnd.hp.com
Mark Kerestes	att!alux2!hawk@uunet.uu.net
Kenneth Key	key@cs.utk.edu
Yoav Kluger	ykluger@fibhaifa.com
Cheryl Krupczak	cheryl@cc.gatech.edu
Ron Lau	rlau@synoptics.com
Chao-Yu Liang	cliang@synoptics.com
Dave Lindemulder	da@mtung.att.com
Richie McBride	rm@bix.co.uk
Keith McCloghrie	kzm@hls.com
Evan McGinnis	bem@3com.com
Donna McMaster	mcmaster@synoptics.com
David Minnich	dwm@fibercom.com
Lynn Monsanto	monsanto@sun.com
Miriam Nihart	miriam@decwet.zso.dec.com
Niels Ole Brunsgaard	nob@dowtyns.dk
Edison Paw	esp@3com.com
David Perkins	dperkins@synoptics.com
Jason Perreault	perreaul@interlan.interlan.com
John Pickens	jrp@3com.com
Jim Reinstedler	jimr@sceng.ub.com
Anil Rijsinghani	anil@levers.enet.dec.com
Sam Roberts	sroberts@farallon.com
Dan Romascanu	dan@lannet.com
Marshall Rose	mrose@dbc.mtview.ca.us
Rick Royston	rick@lsumus.sncc.lsu.edu
Michael Sabo	sabo@dockmaster.ncsc.mil
Jonathan Saperia	saperia@tcpjon.enet.dec.com

Mark Schaefer	schaefer@davidsys.com
Anil Singhal	nsinghal@hawk.ulowell.edu
Timon Sloane	peernet!timon@uunet.uu.net
Bob Stewart	rlstewart@eng.xyplex.com
Emil Sturniolo	emil@dss.com
Bruce Taber	taber@interlan.com
Iris Tal	437-3580@mcimail.com
Mark Therieau	markt@python.eng.microcom.com
Geoff Thompson	thompson@synoptics.com
Dean Throop	throop@dg-rtp.dg.com
Steven Waldbusser	waldbusser@andrew.cmu.edu
Timothy Walden	tmwalden@saturn.sys.acc.com
Philip Wang	watadn!phil@uunet.uu.net
Drew Wansley	dwansley@secola.columbia.ncr.com
David Ward	dward@chipcom.com
Steve Wong	wong@took.enet.dec.com
Paul Woodruff	paul-woodruff@3com.com
Brian Wyld	brianw@spider.co.uk
June-Kang Yang	natadm!yang@uunet.uu.net
Henry Yip	natadm!henry@uunet.uu.net
John Ziegler	ziegler@artel.com
Joseph Zur	fibronics!zur@uunet.uu.net

6. References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] Rose M., Editor, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Performance Systems International, March 1991.
- [5] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization, International Standard 8824, December 1987.

- [6] 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.
- [7] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", STD 16, RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.
- [8] Rose, M., Editor, "A Convention for Defining Traps for use with the SNMP", RFC 1215, Performance Systems International, March 1991.
- [9] 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, 2nd edition, September 21, 1990.
- [10] IEEE P802.3K, "Layer Management for 10 Mb/s Baseband Repeaters, Section 19," Draft Supplement to ANSI/IEEE 802.3, Draft 8, April 9, 1992.

7. Security Considerations

Security issues are not discussed in this memo.

8. Authors' Addresses

Donna McMaster
SynOptics Communications, Inc.
4401 Great America Parkway
P.O. Box 58185
Santa Clara, CA 95052-8185

EMail: mcmaster@synoptics.com

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

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