

Remote Network Monitoring Management Information Base

Status of this Memo

This memo is an extension to the SNMP MIB. 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.

Table of Contents

1. Abstract	2
2. The Network Management Framework.....	2
3. Objects	2
3.1 Format of Definitions	3
4. Overview	3
4.1 Remote Network Management Goals	3
4.2 Textual Conventions	5
4.3 Structure of MIB	5
4.3.1 The Statistics Group	6
4.3.2 The History Group	6
4.3.3 The Alarm Group	6
4.3.4 The Host Group	6
4.3.5 The HostTopN Group	6
4.3.6 The Matrix Group	7
4.3.7 The Filter Group	7
4.3.8 The Packet Capture Group	7
4.3.9 The Event Group	7
5. Control of Remote Network Monitoring Devices	7
5.1 Resource Sharing Among Multiple Management Stations ..	8
5.2 Row Addition Among Multiple Management Stations	9
6. Definitions	10
7. Acknowledgments	80
8. References	80
Security Considerations.....	81
Author's Address.....	81

1. 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 remote network monitoring devices.

2. The Network Management Framework

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

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

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

RFC 1157 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.

3. 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) [7] 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 [3] 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 [8], subject to the additional requirements imposed by the SNMP.

3.1. Format of Definitions

Section 6 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 [9,10].

4. Overview

Remote network monitoring devices are instruments that exist for the purpose of managing a network. Often these remote probes are stand-alone devices and devote significant internal resources for the sole purpose of managing a network. An organization may employ many of these devices, one per network segment, to manage its internet. In addition, these devices may be used for a network management service provider to access a client network, often geographically remote.

While many of the objects in this document are suitable for the management of any type of network, there are some which are specific to managing Ethernet networks. The design of this MIB allows similar objects to be defined for other network types. It is intended that future versions of this document will define extensions for other network types such as Token Ring and FDDI.

4.1. Remote Network Management Goals

o Offline Operation

There are sometimes conditions when a management station will not be in constant contact with its remote monitoring devices. This is sometimes by design in an attempt to lower communications costs (especially when communicating over a WAN or dialup link), or by accident as network failures affect the communications between the management station and the probe.

For this reason, this MIB allows a probe to be configured to perform diagnostics and to collect statistics continuously, even when communication with the management station may not be possible or

efficient. The probe may then attempt to notify the management station when an exceptional condition occurs. Thus, even in circumstances where communication between management station and probe is not continuous, fault, performance, and configuration information may be continuously accumulated and communicated to the management station conveniently and efficiently.

- o Preemptive Monitoring

Given the resources available on the monitor, it is potentially helpful for it continuously to run diagnostics and to log network performance. The monitor is always available at the onset of any failure. It can notify the management station of the failure and can store historical statistical information about the failure. This historical information can be played back by the management station in an attempt to perform further diagnosis into the cause of the problem.

- o Problem Detection and Reporting

The monitor can be configured to recognize conditions, most notably error conditions, and continuously to check for them. When one of these conditions occurs, the event may be logged, and management stations may be notified in a number of ways.

- o Value Added Data

Because a remote monitoring device represents a network resource dedicated exclusively to network management functions, and because it is located directly on the monitored portion of the network, the remote network monitoring device has the opportunity to add significant value to the data it collects. For instance, by highlighting those hosts on the network that generate the most traffic or errors, the probe can give the management station precisely the information it needs to solve a class of problems.

- o Multiple Managers

An organization may have multiple management stations for different units of the organization, for different functions (e.g. engineering and operations), and in an attempt to provide disaster recovery. Because environments with multiple management stations are common, the remote network monitoring device has to

deal with more than own management station,
potentially using its resources concurrently.

4.2. Textual Conventions

Two new data types are introduced as a textual convention in this MIB document. These textual conventions enhance the readability of the specification and can ease comparison with other specifications if appropriate. It should be noted that the introduction of these textual conventions has no effect on either the syntax nor the semantics of any managed objects. The use of these is merely an artifact of the explanatory method used. Objects defined in terms of one of these methods are always encoded by means of the rules that define the primitive type. Hence, no changes to the SMI or the SNMP are necessary to accommodate these textual conventions which are adopted merely for the convenience of readers and writers in pursuit of the elusive goal of clear, concise, and unambiguous MIB documents.

The new data types are: OwnerString and EntryStatus.

4.3. Structure of MIB

The objects are arranged into the following groups:

- statistics
- history
- alarm
- host
- hostTopN
- matrix
- filter
- packet capture
- event

These groups are the basic unit of conformance. If a remote monitoring device implements a group, then it must implement all objects in that group. For example, a managed agent that implements the host group must implement the hostControlTable, the hostTable and the hostTimeTable.

All groups in this MIB are optional. Implementations of this MIB must also implement the system and interfaces group of MIB-II [6]. MIB-II may also mandate the implementation of additional groups.

These groups are defined to provide a means of assigning object identifiers, and to provide a method for managed agents to know which objects they must implement.

4.3.1. The Statistics Group

The statistics group contains statistics measured by the probe for each monitored interface on this device. This group currently consists of the etherStatsTable but in the future will contain tables for other media types including Token Ring and FDDI.

4.3.2. The History Group

The history group records periodic statistical samples from a network and stores them for later retrieval. This group currently consists of the historyControlTable and the etherHistoryTable. In future versions of the MIB, this group may contain tables for other media types including Token Ring and FDDI.

4.3.3. The Alarm Group

The alarm group periodically takes statistical samples from variables in the probe and compares them to previously configured thresholds. If the monitored variable crosses a threshold, an event is generated. A hysteresis mechanism is implemented to limit the generation of alarms. This group consists of the alarmTable and requires the implementation of the event group.

4.3.4. The Host Group

The host group contains statistics associated with each host discovered on the network. This group discovers hosts on the network by keeping a list of source and destination MAC Addresses seen in good packets promiscuously received from the network. This group consists of the hostControlTable, the hostTable, and the hostTimeTable.

4.3.5. The HostTopN Group

The hostTopN group is used to prepare reports that describe the hosts that top a list ordered by one of their statistics. The available statistics are samples of one of their base statistics over an interval specified by the management station. Thus, these statistics are rate based. The management station also selects how many such

hosts are reported. This group consists of the hostTopNControlTable and the hostTopNTable, and requires the implementation of the host group.

4.3.6. The Matrix Group

The matrix group stores statistics for conversations between sets of two addresses. As the device detects a new conversation, it creates a new entry in its tables. This group consists of the matrixControlTable, the matrixSDTable and the matrixDSTable.

4.3.7. The Filter Group

The filter group allows packets to be matched by a filter equation. These matched packets form a data stream that may be captured or may generate events. This group consists of the filterTable and the channelTable.

4.3.8. The Packet Capture Group

The Packet Capture group allows packets to be captured after they flow through a channel. This group consists of the bufferControlTable and the captureBufferTable, and requires the implementation of the filter group.

4.3.9. The Event Group

The event group controls the generation and notification of events from this device. This group consists of the eventTable and the logTable.

5. Control of Remote Network Monitoring Devices

Due to the complex nature of the available functions in these devices, the functions often need user configuration. In many cases, the function requires parameters to be set up for a data collection operation. The operation can proceed only after these parameters are fully set up.

Many functional groups in this MIB have one or more tables in which to set up control parameters, and one or more data tables in which to place the results of the operation. The control tables are typically read-write in nature, while the data tables are typically read-only. Because the parameters in the control table often describe resulting data in the data table, many of the parameters can be modified only when the control entry is invalid. Thus, the method for modifying these parameters is to invalidate the control entry, causing its deletion and the deletion of any associated data entries, and then

create a new control entry with the proper parameters. Deleting the control entry also gives a convenient method for reclaiming the resources used by the associated data.

Some objects in this MIB provide a mechanism to execute an action on the remote monitoring device. These objects may execute an action as a result of a change in the state of the object. For those objects in this MIB, a request to set an object to the same value as it currently holds would thus cause no action to occur.

To facilitate control by multiple managers, resources have to be shared among the managers. These resources are typically the memory and computation resources that a function requires.

5.1. Resource Sharing Among Multiple Management Stations

When multiple management stations wish to use functions that compete for a finite amount of resources on a device, a method to facilitate this sharing of resources is required. Potential conflicts include:

- o Two management stations wish to simultaneously use resources that together would exceed the capability of the device.
- o A management station uses a significant amount of resources for a long period of time.
- o A management station uses resources and then crashes, forgetting to free the resources so others may use them.

A mechanism is provided for each management station initiated function in this MIB to avoid these conflicts and to help resolve them when they occur. Each function has a label identifying the initiator (owner) of the function. This label is set by the initiator to provide for the following possibilities:

- o A management station may recognize resources it owns and no longer needs.
- o A network operator can find the management station that owns the resource and negotiate for it to be freed.
- o A network operator may decide to unilaterally free resources another network operator has reserved.
- o Upon initialization, a management station may recognize resources it had reserved in the past. With this

information it may free the resources if it no longer needs them.

Management stations and probes should support any format of the owner string dictated by the local policy of the organization. It is suggested that this name contain one or more of the following: IP address, management station name, network manager's name, location, or phone number. This information will help users to share the resources more effectively.

There is often default functionality that the device wishes to set up. The resources associated with this functionality are then owned by the device itself. In this case, the device will set the relevant owner object to a string starting with 'monitor'. Indiscriminate modification of the monitor-owned configuration by network management stations is discouraged. In fact, a network management station should only modify these objects under the direction of the administrator of the probe, often the network administrator.

When a network management station wishes to utilize a function in a monitor, it is encouraged to first scan the control table of that function to find an instance with similar parameters to share. This is especially true for those instances owned by the monitor, which can be assumed to change infrequently. If a management station decides to share an instance owned by another management station, it should understand that the management station that owns the instance may indiscriminately modify or delete it.

5.2. Row Addition Among Multiple Management Stations

The addition of new rows is achieved using the method described in [9]. In this MIB, rows are often added to a table in order to configure a function. This configuration usually involves parameters that control the operation of the function. The agent must check these parameters to make sure they are appropriate given restrictions defined in this MIB as well as any implementation specific restrictions such as lack of resources. The agent implementor may be confused as to when to check these parameters and when to signal to the management station that the parameters are invalid. There are two opportunities:

- o When the management station sets each parameter object.
- o When the management station sets the entry status object to valid.

If the latter is chosen, it would be unclear to the management station which of the several parameters was invalid and caused the

badValue error to be emitted. Thus, wherever possible, the implementor should choose the former as it will provide more information to the management station.

A problem can arise when multiple management stations attempt to set configuration information simultaneously using SNMP. When this involves the addition of a new conceptual row in the same control table, the managers may collide, attempting to create the same entry. To guard against these collisions, each such control entry contains a status object with special semantics that help to arbitrate among the managers. If an attempt is made with the row addition mechanism to create such a status object and that object already exists, an error is returned. When more than one manager simultaneously attempts to create the same conceptual row, only the first will succeed. The others will receive an error.

6. Definitions

```
RFC1271-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    Counter                FROM RFC1155-SMI
    DisplayString           FROM RFC1158-MIB
    mib-2                   FROM RFC1213-MIB
    OBJECT-TYPE             FROM RFC-1212;
```

```
-- This MIB module uses the extended OBJECT-TYPE macro as
-- defined in [9].
```

```
-- Remote Network Monitoring MIB
```

```
rmon    OBJECT IDENTIFIER ::= { mib-2 16 }
```

```
-- textual conventions
```

```
OwnerString ::= DisplayString
```

```
-- This data type is used to model an administratively
-- assigned name of the owner of a resource. This
-- information is taken from the NVT ASCII character set.
-- It is suggested that this name contain one or more
-- of the following:
-- IP address, management station name, network manager's
-- name, location, or phone number.
-- In some cases the agent itself will be the owner of
-- an entry. In these cases, this string shall be set
-- to a string starting with 'monitor'.
```

```
--
-- SNMP access control is articulated entirely in terms of
-- the contents of MIB views; access to a particular SNMP
-- object instance depends only upon its presence or
-- absence in a particular MIB view and never upon its
-- value or the value of related object instances. Thus,
-- objects of this type afford resolution of resource
-- contention only among cooperating managers; they
-- realize no access control function with respect
-- to uncooperative parties.
--
-- By convention, objects with this syntax are declared
-- as having
--
--     SIZE (0..127)

EntryStatus ::= INTEGER
    { valid(1),
      createRequest(2),
      underCreation(3),
      invalid(4)
    }

-- The status of a table entry.
--
-- Setting this object to the value invalid(4) has the
-- effect of invalidating the corresponding entry.
-- That is, it effectively disassociates the mapping
-- identified with said entry.
-- It is an implementation-specific matter as to whether
-- the agent removes an invalidated entry from the table.
-- Accordingly, management stations must be prepared to
-- receive tabular information from agents that corresponds
-- to entries currently not in use. Proper
-- interpretation of such entries requires examination
-- of the relevant EntryStatus object.
--
-- An existing instance of this object cannot be set to
-- createRequest(2). This object may only be set to
-- createRequest(2) when this instance is created. When
-- this object is created, the agent may wish to create
-- supplemental object instances to complete a conceptual
-- row in this table. Immediately after completing the
-- create operation, the agent must set this object to
-- underCreation(3).
--
-- Entries shall exist in the underCreation(3) state until
```

```
-- the management station is finished configuring the
-- entry and sets this object to valid(1) or aborts,
-- setting this object to invalid(4).  If the agent
-- determines that an entry has been in the
-- underCreation(3) state for an abnormally long time,
-- it may decide that the management station has
-- crashed.  If the agent makes this decision,
-- it may set this object to invalid(4) to reclaim the
-- entry.  A prudent agent will understand that the
-- management station may need to wait for human input
-- and will allow for that possibility in its
-- determination of this abnormally long period.
```

statistics	OBJECT IDENTIFIER ::= { rmon 1 }
history	OBJECT IDENTIFIER ::= { rmon 2 }
alarm	OBJECT IDENTIFIER ::= { rmon 3 }
hosts	OBJECT IDENTIFIER ::= { rmon 4 }
hostTopN	OBJECT IDENTIFIER ::= { rmon 5 }
matrix	OBJECT IDENTIFIER ::= { rmon 6 }
filter	OBJECT IDENTIFIER ::= { rmon 7 }
capture	OBJECT IDENTIFIER ::= { rmon 8 }
event	OBJECT IDENTIFIER ::= { rmon 9 }

```
-- The Statistics Group
```

```
--
```

```
-- Implementation of the Statistics group is optional.
```

```
--
```

```
-- The statistics group contains statistics measured by the
-- probe for each monitored interface on this device.  These
-- statistics take the form of free running counters that
-- start from zero when a valid entry is created.
```

```
--
```

```
-- This group currently has statistics defined only for
-- Ethernet interfaces.  Each etherStatsEntry contains
-- statistics for one Ethernet interface.  The probe must
-- create one etherStats entry for each monitored Ethernet
-- interface on the device.
```

```
etherStatsTable OBJECT-TYPE
```

```
    SYNTAX SEQUENCE OF EtherStatsEntry
```

```
    ACCESS not-accessible
```

```
    STATUS mandatory
```

```
    DESCRIPTION
```

```
        "A list of Ethernet statistics entries."
```

```
    ::= { statistics 1 }
```

```

etherStatsEntry OBJECT-TYPE
    SYNTAX EtherStatsEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A collection of statistics kept for a particular
        Ethernet interface."
    INDEX { etherStatsIndex }
    ::= { etherStatsTable 1 }

EtherStatsEntry ::= SEQUENCE {
    etherStatsIndex                INTEGER (1..65535),
    etherStatsDataSource           OBJECT IDENTIFIER,
    etherStatsDropEvents          Counter,
    etherStatsOctets              Counter,
    etherStatsPkts                Counter,
    etherStatsBroadcastPkts       Counter,
    etherStatsMulticastPkts       Counter,
    etherStatsCRCAlignErrors      Counter,
    etherStatsUndersizePkts       Counter,
    etherStatsOversizePkts        Counter,
    etherStatsFragments           Counter,
    etherStatsJabbers             Counter,
    etherStatsCollisions          Counter,
    etherStatsPkts64Octets        Counter,
    etherStatsPkts65to127Octets   Counter,
    etherStatsPkts128to255Octets  Counter,
    etherStatsPkts256to511Octets  Counter,
    etherStatsPkts512to1023Octets Counter,
    etherStatsPkts1024to1518Octets Counter,
    etherStatsOwner               OwnerString,
    etherStatsStatus              INTEGER
}

etherStatsIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object uniquely identifies this
        etherStats entry."
    ::= { etherStatsEntry 1 }

etherStatsDataSource OBJECT-TYPE
    SYNTAX OBJECT IDENTIFIER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION

```

"This object identifies the source of the data that this etherStats entry is configured to analyze. This source can be any ethernet interface on this device. In order to identify a particular interface, this object shall identify the instance of the ifIndex object, defined in [4,6], for the desired interface. For example, if an entry were to receive data from interface #1, this object would be set to ifIndex.1.

The statistics in this group reflect all packets on the local network segment attached to the identified interface.

This object may not be modified if the associated etherStatsStatus object is equal to valid(1)."

::= { etherStatsEntry 2 }

etherStatsDropEvents OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of events in which packets were dropped by the probe due to lack of resources. Note that this number is not necessarily the number of packets dropped; it is just the number of times this condition has been detected."

::= { etherStatsEntry 3 }

etherStatsOctets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets)."

::= { etherStatsEntry 4 }

etherStatsPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets (including error packets) received."

::= { etherStatsEntry 5 }

etherStatsBroadcastPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of good packets received that were
 directed to the broadcast address."
 ::= { etherStatsEntry 6 }

etherStatsMulticastPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of good packets received that were
 directed to a multicast address. Note that this
 number does not include packets directed to the
 broadcast address."
 ::= { etherStatsEntry 7 }

etherStatsCRCAlignErrors OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of packets received that
 had a length (excluding framing bits, but
 including FCS octets) of between 64 and 1518
 octets, inclusive, but were not an integral number
 of octets in length or had a bad Frame Check
 Sequence (FCS)."
 ::= { etherStatsEntry 8 }

etherStatsUndersizePkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of packets received that were
 less than 64 octets long (excluding framing bits,
 but including FCS octets) and were otherwise well
 formed."
 ::= { etherStatsEntry 9 }

etherStatsOversizePkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory

DESCRIPTION

"The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed."

::= { etherStatsEntry 10 }

etherStatsFragments OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets received that were not an integral number of octets in length or that had a bad Frame Check Sequence (FCS), and were less than 64 octets in length (excluding framing bits but including FCS octets)."

::= { etherStatsEntry 11 }

etherStatsJabbers OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets), and were not an integral number of octets in length or had a bad Frame Check Sequence (FCS)."

::= { etherStatsEntry 12 }

etherStatsCollisions OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The best estimate of the total number of collisions on this Ethernet segment."

::= { etherStatsEntry 13 }

etherStatsPkts64Octets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets (including error packets) received that were 64 octets in length (excluding framing bits but including FCS octets)."

```
::= { etherStatsEntry 14 }

etherStatsPkts65to127Octets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of packets (including error
        packets) received that were between
        65 and 127 octets in length inclusive
        (excluding framing bits but including FCS octets)."
```

```
::= { etherStatsEntry 15 }

etherStatsPkts128to255Octets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of packets (including error
        packets) received that were between
        128 and 255 octets in length inclusive
        (excluding framing bits but including FCS octets)."
```

```
::= { etherStatsEntry 16 }

etherStatsPkts256to511Octets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of packets (including error
        packets) received that were between
        256 and 511 octets in length inclusive
        (excluding framing bits but including FCS octets)."
```

```
::= { etherStatsEntry 17 }

etherStatsPkts512to1023Octets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The total number of packets (including error
        packets) received that were between
        512 and 1023 octets in length inclusive
        (excluding framing bits but including FCS octets)."
```

```
::= { etherStatsEntry 18 }
```

etherStatsPkts1024to1518Octets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets (including error packets) received that were between 1024 and 1518 octets in length inclusive (excluding framing bits but including FCS octets)."

::= { etherStatsEntry 19 }

etherStatsOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { etherStatsEntry 20 }

etherStatsStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status of this etherStats entry."

::= { etherStatsEntry 21 }

-- The History Group

-- Implementation of the History group is optional.

--

-- The history group records periodic statistical samples from
-- a network and stores them for later retrieval. The
-- historyControl table stores configuration entries that each
-- define an interface, polling period, and other parameters.
-- Once samples are taken, their data is stored in an entry
-- in a media-specific table. Each such entry defines one
-- sample, and is associated with the historyControlEntry that
-- caused the sample to be taken. Currently the only media-
-- specific table defined is the etherHistoryTable, for
-- Ethernet networks.

--

-- If the probe keeps track of the time of day, it should
-- start the first sample of the history at a time such that
-- when the next hour of the day begins, a sample is
-- started at that instant. This tends to make more

```
-- user-friendly reports, and enables comparison of reports
-- from different probes that have relatively accurate time
-- of day.
--
-- The monitor is encouraged to add two history control entries
-- per monitored interface upon initialization that describe
-- a short term and a long term polling period. Suggested
-- parameters are 30 seconds for the short term polling
-- period and 30 minutes for the long term period.
```

```
historyControlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF HistoryControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of history control entries."
    ::= { history 1 }
```

```
historyControlEntry OBJECT-TYPE
    SYNTAX HistoryControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of parameters that set up a periodic
        sampling of statistics."
    INDEX { historyControlIndex }
    ::= { historyControlTable 1 }
```

```
HistoryControlEntry ::= SEQUENCE {
    historyControlIndex          INTEGER (1..65535),
    historyControlDataSource     OBJECT IDENTIFIER,
    historyControlBucketsRequested  INTEGER (1..65535),
    historyControlBucketsGranted  INTEGER (1..65535),
    historyControlInterval       INTEGER (1..3600),
    historyControlOwner           OwnerString,
    historyControlStatus         INTEGER
}
```

```
historyControlIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry in the
        historyControl table. Each such entry defines a
        set of samples at a particular interval for an
        interface on the device."
    ::= { historyControlEntry 1 }
```

historyControlDataSource OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"This object identifies the source of the data for which historical data was collected and placed in a media-specific table on behalf of this historyControlEntry. This source can be any interface on this device. In order to identify a particular interface, this object shall identify the instance of the ifIndex object, defined in [4,6], for the desired interface. For example, if an entry were to receive data from interface #1, this object would be set to ifIndex.1.

The statistics in this group reflect all packets on the local network segment attached to the identified interface.

This object may not be modified if the associated historyControlStatus object is equal to valid(1)."

::= { historyControlEntry 2 }

historyControlBucketsRequested OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The requested number of discrete time intervals over which data is to be saved in the part of the media-specific table associated with this historyControl entry.

When this object is created or modified, the probe should set historyControlBucketsGranted as closely to this object as is possible for the particular probe implementation and available resources."

DEFVAL { 50 }

::= { historyControlEntry 3 }

historyControlBucketsGranted OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of discrete sampling intervals over which data shall be saved in the part of

the media-specific table associated with this historyControl entry.

When the associated historyControlBucketsRequested object is created or modified, the probe should set this object as closely to the requested value as is possible for the particular probe implementation and available resources. The probe must not lower this value except as a result of a modification to the associated historyControlBucketsRequested object.

There will be times when the actual number of buckets associated with this entry is less than the value of this object. In this case, at the end of each sampling interval, a new bucket will be added to the media-specific table.

When the number of buckets reaches the value of this object and a new bucket is to be added to the media-specific table, the oldest bucket associated with this historyControlEntry shall be deleted by the agent so that the new bucket can be added.

When the value of this object changes to a value less than the current value, entries are deleted from the media-specific table associated with this historyControlEntry. Enough of the oldest of these entries shall be deleted by the agent so that their number remains less than or equal to the new value of this object.

When the value of this object changes to a value greater than the current value, the number of associated media-specific entries may be allowed to grow."

::= { historyControlEntry 4 }

historyControlInterval OBJECT-TYPE

SYNTAX INTEGER (1..3600)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The interval in seconds over which the data is sampled for each bucket in the part of the media-specific table associated with this historyControl entry. This interval can be set to any number of seconds between 1 and

3600 (1 hour).

Because the counters in a bucket may overflow at their maximum value with no indication, a prudent manager will take into account the possibility of overflow in any of the associated counters. It is important to consider the minimum time in which any counter could overflow on a particular media type and set the historyControlInterval object to a value less than this interval. This is typically most important for the 'octets' counter in any media-specific table. For example, on an Ethernet network, the etherHistoryOctets counter could overflow in about one hour at the Ethernet's maximum utilization.

This object may not be modified if the associated historyControlStatus object is equal to valid(1)."

DEFVAL { 1800 }
 ::= { historyControlEntry 5 }

historyControlOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { historyControlEntry 6 }

historyControlStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status of this historyControl entry.

Each instance of the media-specific table associated with this historyControlEntry will be deleted by the agent if this historyControlEntry is not equal to valid(1)."

::= { historyControlEntry 7 }

-- Ether History table

etherHistoryTable OBJECT-TYPE

SYNTAX SEQUENCE OF EtherHistoryEntry

```

ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A list of Ethernet history entries."
 ::= { history 2 }

```

```

etherHistoryEntry OBJECT-TYPE
    SYNTAX EtherHistoryEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "An historical sample of Ethernet statistics on a
        particular Ethernet interface. This sample is
        associated with the historyControlEntry which set
        up the parameters for a regular collection of these
        samples."
    INDEX { etherHistoryIndex , etherHistorySampleIndex }
    ::= { etherHistoryTable 1 }

```

```

EtherHistoryEntry ::= SEQUENCE {
    etherHistoryIndex                INTEGER (1..65535),
    etherHistorySampleIndex          INTEGER,
    etherHistoryIntervalStart        TimeTicks,
    etherHistoryDropEvents           Counter,
    etherHistoryOctets               Counter,
    etherHistoryPkts                 Counter,
    etherHistoryBroadcastPkts        Counter,
    etherHistoryMulticastPkts        Counter,
    etherHistoryCRCAlignErrors       Counter,
    etherHistoryUndersizePkts        Counter,
    etherHistoryOversizePkts         Counter,
    etherHistoryFragments            Counter,
    etherHistoryJabbers              Counter,
    etherHistoryCollisions           Counter,
    etherHistoryUtilization          INTEGER (0..10000)
}

```

```

etherHistoryIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The history of which this entry is a part. The
        history identified by a particular value of this
        index is the same history as identified
        by the same value of historyControlIndex."
    ::= { etherHistoryEntry 1 }

```

etherHistorySampleIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "An index that uniquely identifies the particular sample this entry represents among all samples associated with the same historyControlEntry. This index starts at 1 and increases by one as each new sample is taken."
 ::= { etherHistoryEntry 2 }

etherHistoryIntervalStart OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The value of sysUpTime at the start of the interval over which this sample was measured. If the probe keeps track of the time of day, it should start the first sample of the history at a time such that when the next hour of the day begins, a sample is started at that instant. Note that following this rule may require the probe to delay collecting the first sample of the history, as each sample must be of the same interval. Also note that the sample which is currently being collected is not accessible in this table until the end of its interval."
 ::= { etherHistoryEntry 3 }

etherHistoryDropEvents OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of events in which packets were dropped by the probe due to lack of resources during this interval. Note that this number is not necessarily the number of packets dropped, it is just the number of times this condition has been detected."
 ::= { etherHistoryEntry 4 }

etherHistoryOctets OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The total number of octets of data (including

```
        those in bad packets) received on the
        network (excluding framing bits but including
        FCS octets)."
```

```
 ::= { etherHistoryEntry 5 }
```

```
etherHistoryPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of packets (including error packets)
        received during this sampling interval."
    ::= { etherHistoryEntry 6 }
```

```
etherHistoryBroadcastPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of good packets received during this
        sampling interval that were directed to the
        broadcast address."
    ::= { etherHistoryEntry 7 }
```

```
etherHistoryMulticastPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of good packets received during this
        sampling interval that were directed to a
        multicast address. Note that this number does not
        include packets addressed to the broadcast address."
    ::= { etherHistoryEntry 8 }
```

```
etherHistoryCRCAlignErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of packets received during this
        sampling interval that had a length (excluding
        framing bits but including FCS octets) between
        64 and 1518 octets, inclusive, but were not an
        integral number of octets in length or had a
        bad Frame Check Sequence (FCS)."
```

```
 ::= { etherHistoryEntry 9 }
```

etherHistoryUndersizePkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets received during this interval that were less than 64 octets long (excluding framing bits but including FCS octets) and were otherwise well formed."

::= { etherHistoryEntry 10 }

etherHistoryOversizePkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets received during this interval that were longer than 1518 octets (excluding framing bits but including FCS octets) but were otherwise well formed."

::= { etherHistoryEntry 11 }

etherHistoryFragments OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The total number of packets received during this sampling interval that were not an integral number of octets in length or that had a bad Frame Check Sequence (FCS), and were less than 64 octets in length (excluding framing bits but including FCS octets)."

::= { etherHistoryEntry 12 }

etherHistoryJabbers OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets received during this interval that were longer than 1518 octets (excluding framing bits but including FCS octets), and were not an integral number of octets in length or had a bad Frame Check Sequence (FCS)."

::= { etherHistoryEntry 13 }

etherHistoryCollisions OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The best estimate of the total number of collisions on this Ethernet segment during this interval."

::= { etherHistoryEntry 14 }

etherHistoryUtilization OBJECT-TYPE

SYNTAX INTEGER (0..10000)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The best estimate of the mean physical layer network utilization on this interface during this interval, in hundredths of a percent."

::= { etherHistoryEntry 15 }

-- The Alarm Group

-- Implementation of the Alarm group is optional.

--

-- The Alarm Group requires the implementation of the Event group.

--

-- The Alarm group periodically takes statistical samples from variables in the probe and compares them to thresholds that have been configured. The alarm table stores configuration entries that each define a variable, polling period, and threshold parameters. If a sample is found to cross the threshold values, an event is generated. Only variables that resolve to an ASN.1 primitive type of INTEGER (INTEGER, Counter, Gauge, or TimeTicks) may be monitored in this way.

--

-- This function has a hysteresis mechanism to limit the generation of events. This mechanism generates one event as a threshold is crossed in the appropriate direction. No more events are generated for that threshold until the opposite threshold is crossed.

--

-- In the case of a sampling a deltaValue, a probe may implement this mechanism with more precision if it takes a delta sample twice per period, each time comparing the sum of the latest two samples to the threshold. This allows the detection of threshold crossings

```
-- that span the sampling boundary.  Note that this does not
-- require any special configuration of the threshold value.
-- It is suggested that probes implement this more precise
-- algorithm.
```

```
alarmTable OBJECT-TYPE
    SYNTAX SEQUENCE OF AlarmEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of alarm entries."
    ::= { alarm 1 }
```

```
alarmEntry OBJECT-TYPE
    SYNTAX AlarmEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of parameters that set up a periodic checking
        for alarm conditions."
    INDEX { alarmIndex }
    ::= { alarmTable 1 }
```

```
AlarmEntry ::= SEQUENCE {
    alarmIndex                INTEGER (1..65535),
    alarmInterval             INTEGER,
    alarmVariable             OBJECT IDENTIFIER,
    alarmSampleType           INTEGER,
    alarmValue                INTEGER,
    alarmStartupAlarm         INTEGER,
    alarmRisingThreshold      INTEGER,
    alarmFallingThreshold     INTEGER,
    alarmRisingEventIndex     INTEGER (1..65535),
    alarmFallingEventIndex    INTEGER (1..65535),
    alarmOwner                OwnerString,
    alarmStatus               INTEGER
}
```

```
alarmIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry in the
        alarm table.  Each such entry defines a
        diagnostic sample at a particular interval
        for an object on the device."
    ::= { alarmEntry 1 }
```

`alarmInterval OBJECT-TYPE``SYNTAX INTEGER``ACCESS read-write``STATUS mandatory``DESCRIPTION`

"The interval in seconds over which the data is sampled and compared with the rising and falling thresholds. When setting this variable, care should be given to ensure that the variable being monitored will not exceed $2^{31} - 1$ and roll over the alarmValue object during the interval.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."

::= { alarmEntry 2 }

`alarmVariable OBJECT-TYPE``SYNTAX OBJECT IDENTIFIER``ACCESS read-write``STATUS mandatory``DESCRIPTION`

"The object identifier of the particular variable to be sampled. Only variables that resolve to an ASN.1 primitive type of INTEGER (INTEGER, Counter, Gauge, or TimeTicks) may be sampled.

Because SNMP access control is articulated entirely in terms of the contents of MIB views, no access control mechanism exists that can restrict the value of this object to identify only those objects that exist in a particular MIB view. Because there is thus no acceptable means of restricting the read access that could be obtained through the alarm mechanism, the probe must only grant write access to this object in those views that have read access to all objects on the probe.

During a set operation, if the supplied variable name is not available in the selected MIB view, a badValue error must be returned. If at any time the variable name of an established alarmEntry is no longer available in the selected MIB view, the probe must change the status of this alarmEntry to invalid(4).

This object may not be modified if the associated alarmStatus object is equal to valid(1)."

::= { alarmEntry 3 }

alarmSampleType OBJECT-TYPE

```
SYNTAX INTEGER {  
    absoluteValue(1),  
    deltaValue(2)  
}
```

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The method of sampling the selected variable and calculating the value to be compared against the thresholds. If the value of this object is absoluteValue(1), the value of the selected variable will be compared directly with the thresholds at the end of the sampling interval. If the value of this object is deltaValue(2), the value of the selected variable at the last sample will be subtracted from the current value, and the difference compared with the thresholds.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."

```
::= { alarmEntry 4 }
```

alarmValue OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The value of the statistic during the last sampling period. The value during the current sampling period is not made available until the period is completed."

```
::= { alarmEntry 5 }
```

alarmStartupAlarm OBJECT-TYPE

```
SYNTAX INTEGER {  
    risingAlarm(1),  
    fallingAlarm(2),  
    risingOrFallingAlarm(3)  
}
```

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The alarm that may be sent when this entry is first set to valid. If the first sample after this entry becomes valid is greater than or equal to the risingThreshold and alarmStartupAlarm is equal to risingAlarm(1) or risingOrFallingAlarm(3), then a single rising alarm will be generated. If the first

sample after this entry becomes valid is less than or equal to the fallingThreshold and alarmStartupAlarm is equal to fallingAlarm(2) or risingOrFallingAlarm(3), then a single falling alarm will be generated.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."
 ::= { alarmEntry 6 }

alarmRisingThreshold OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"A threshold for the sampled statistic. When the current sampled value is greater than or equal to this threshold, and the value at the last sampling interval was less than this threshold, a single event will be generated.

A single event will also be generated if the first sample after this entry becomes valid is greater than or equal to this threshold and the associated alarmStartupAlarm is equal to risingAlarm(1) or risingOrFallingAlarm(3).

After a rising event is generated, another such event will not be generated until the sampled value falls below this threshold and reaches the alarmFallingThreshold.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."
 ::= { alarmEntry 7 }

alarmFallingThreshold OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"A threshold for the sampled statistic. When the current sampled value is less than or equal to this threshold, and the value at the last sampling interval was greater than this threshold, a single event will be generated.

A single event will also be generated if the first sample after this entry becomes valid is less than or equal to this threshold and the associated

alarmStartupAlarm is equal to fallingAlarm(2) or risingOrFallingAlarm(3).

After a falling event is generated, another such event will not be generated until the sampled value rises above this threshold and reaches the alarmRisingThreshold.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."
 ::= { alarmEntry 8 }

alarmRisingEventIndex OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The index of the eventEntry that is used when a rising threshold is crossed. The eventEntry identified by a particular value of this index is the same as identified by the same value of the eventIndex object. If there is no corresponding entry in the eventTable, then no association exists. In particular, if this value is zero, no associated event will be generated, as zero is not a valid event index.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."
 ::= { alarmEntry 9 }

alarmFallingEventIndex OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The index of the eventEntry that is used when a falling threshold is crossed. The eventEntry identified by a particular value of this index is the same as identified by the same value of the eventIndex object. If there is no corresponding entry in the eventTable, then no association exists. In particular, if this value is zero, no associated event will be generated, as zero is not a valid event index.

This object may not be modified if the associated alarmStatus object is equal to valid(1)."

```
 ::= { alarmEntry 10 }

alarmOwner OBJECT-TYPE
    SYNTAX OwnerString
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The entity that configured this entry and is
        therefore using the resources assigned to it."
    ::= { alarmEntry 11 }

alarmStatus OBJECT-TYPE
    SYNTAX EntryStatus
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The status of this alarm entry."
    ::= { alarmEntry 12 }

-- The Host Group

-- Implementation of the Host group is optional.
--
-- The host group discovers new hosts on the network by
-- keeping a list of source and destination MAC Addresses seen
-- in good packets. For each of these addresses, the host
-- group keeps a set of statistics. The hostControlTable
-- controls which interfaces this function is performed on,
-- and contains some information about the process. On
-- behalf of each hostControlEntry, data is collected on an
-- interface and placed both the hostTable and the
-- hostTimeTable. If the monitoring device finds itself
-- short of resources, it may delete entries as needed. It
-- is suggested that the device delete the least recently
-- used entries first.

-- The hostTable contains entries for each address
-- discovered on a particular interface. Each entry
-- contains statistical data about that host. This table
-- is indexed by the MAC address of the host, through
-- which a random access may be achieved.

-- The hostTimeTable contains data in the same format as the
-- hostTable, and must contain the same set of hosts, but is
-- indexed using hostTimeCreationOrder rather than hostAddress.
-- The hostTimeCreationOrder is an integer which reflects
-- the relative order in which a particular entry was
```

```
-- discovered and thus inserted into the table.  As this
-- order, and thus index, is among those entries currently
-- in the table, the index for a particular entry may change
-- if an (earlier) entry is deleted.  Thus the association
-- between hostTimeCreationOrder and hostTimeEntry may be
-- broken at any time.

-- The hostTimeTable has two important uses.  The first is the
-- fast download of this potentially large table.  Because the
-- index of this table runs from 1 to the size of the table,
-- inclusive, its values are predictable.  This allows very
-- efficient packing of variables into SNMP PDU's and allows
-- a table transfer to have multiple packets outstanding.
-- These benefits increase transfer rates tremendously.

-- The second use of the hostTimeTable is the efficient
-- discovery by the management station of new entries added
-- to the table.  After the management station has
-- downloaded the entire table, it knows that new entries
-- will be added immediately after the end of the current
-- table.  It can thus detect new entries there
-- and retrieve them easily.

-- Because the association between hostTimeCreationOrder and
-- hostTimeEntry may be broken at any time, the management
-- station must monitor the related hostControlLastDeleteTime
-- object.  When the management station thus detects a deletion,
-- it must assume that any such associations have been broken,
-- and invalidate any it has stored locally.  This includes
-- restarting any download of the hostTimeTable that may have
-- been in progress, as well as rediscovering the end of the
-- hostTimeTable so that it may detect new entries.  If the
-- management station does not detect the broken association,
-- it may continue to refer to a particular host by its
-- creationOrder while unwittingly retrieving the data
-- associated with another host entirely.  If this happens
-- while downloading the host table, the management station
-- may fail to download all of the entries in the table.
```

hostControlTable OBJECT-TYPE

SYNTAX SEQUENCE OF HostControlEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"A list of host table control entries."

::= { hosts 1 }

hostControlEntry OBJECT-TYPE

SYNTAX HostControlEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"A list of parameters that set up the discovery of hosts on a particular interface and the collection of statistics about these hosts."

INDEX { hostControlIndex }

::= { hostControlTable 1 }

HostControlEntry ::= SEQUENCE {

hostControlIndex INTEGER (1..65535),

hostControlDataSource OBJECT IDENTIFIER,

hostControlTableSize INTEGER,

hostControlLastDeleteTime TimeTicks,

hostControlOwner OwnerString,

hostControlStatus INTEGER

}

hostControlIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index that uniquely identifies an entry in the hostControl table. Each such entry defines a function that discovers hosts on a particular interface and places statistics about them in the hostTable and the hostTimeTable on behalf of this hostControlEntry."

::= { hostControlEntry 1 }

hostControlDataSource OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"This object identifies the source of the data for this instance of the host function. This source can be any interface on this device. In order to identify a particular interface, this object shall identify the instance of the ifIndex object, defined in [4,6], for the desired interface. For example, if an entry were to receive data from interface #1, this object would be set to ifIndex.1."

The statistics in this group reflect all packets

on the local network segment attached to the identified interface.

This object may not be modified if the associated hostControlStatus object is equal to valid(1)."
::= { hostControlEntry 2 }

hostControlTableSize OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of hostEntries in the hostTable and the hostTimeTable associated with this hostControlEntry."

::= { hostControlEntry 3 }

hostControlLastDeleteTime OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The value of sysUpTime when the last entry was deleted from the portion of the hostTable associated with this hostControlEntry. If no deletions have occurred, this value shall be zero."

::= { hostControlEntry 4 }

hostControlOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { hostControlEntry 5 }

hostControlStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status of this hostControl entry.

If this object is not equal to valid(1), all associated entries in the hostTable, hostTimeTable, and the hostTopNTable shall be deleted by the agent."

::= { hostControlEntry 6 }

```
hostTable OBJECT-TYPE
    SYNTAX SEQUENCE OF HostEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of host entries."
    ::= { hosts 2 }

hostEntry OBJECT-TYPE
    SYNTAX HostEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A collection of statistics for a particular host
        that has been discovered on an interface of this
        device."
    INDEX { hostIndex, hostAddress }
    ::= { hostTable 1 }

HostEntry ::= SEQUENCE {
    hostAddress          OCTET STRING,
    hostCreationOrder    INTEGER (1..65535),
    hostIndex            INTEGER (1..65535),
    hostInPkts           Counter,
    hostOutPkts          Counter,
    hostInOctets          Counter,
    hostOutOctets        Counter,
    hostOutErrors         Counter,
    hostOutBroadcastPkts Counter,
    hostOutMulticastPkts Counter
}

hostAddress OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The physical address of this host."
    ::= { hostEntry 1 }

hostCreationOrder OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that defines the relative ordering of
        the creation time of hosts captured for a
        particular hostControlEntry. This index shall
```

be between 1 and N, where N is the value of the associated hostControlTableSize. The ordering of the indexes is based on the order of each entry's insertion into the table, in which entries added earlier have a lower index value than entries added later.

It is important to note that the order for a particular entry may change as an (earlier) entry is deleted from the table. Because this order may change, management stations should make use of the hostControlLastDeleteTime variable in the hostControlEntry associated with the relevant portion of the hostTable. By observing this variable, the management station may detect the circumstances where a previous association between a value of hostCreationOrder and a hostEntry may no longer hold."

::= { hostEntry 2 }

hostIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The set of collected host statistics of which this entry is a part. The set of hosts identified by a particular value of this index is associated with the hostControlEntry as identified by the same value of hostControlIndex."

::= { hostEntry 3 }

hostInPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets without errors transmitted to this address since it was added to the hostTable."

::= { hostEntry 4 }

hostOutPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets including errors transmitted by this address since it was added to the hostTable."

::= { hostEntry 5 }

hostInOctets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of octets transmitted to this address since it was added to the hostTable (excluding framing bits but including FCS octets), except for those octets in packets that contained errors."

::= { hostEntry 6 }

hostOutOctets OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of octets transmitted by this address since it was added to the hostTable (excluding framing bits but including FCS octets), including those octets in packets that contained errors."

::= { hostEntry 7 }

hostOutErrors OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of error packets transmitted by this address since this host was added to the hostTable."

::= { hostEntry 8 }

hostOutBroadcastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of good packets transmitted by this address that were directed to the broadcast address since this host was added to the hostTable."

::= { hostEntry 9 }

hostOutMulticastPkts OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

```

        "The number of good packets transmitted by this
        address that were directed to a multicast address
        since this host was added to the hostTable.
        Note that this number does not include packets
        directed to the broadcast address."
 ::= { hostEntry 10 }

-- host Time Table

hostTimeTable OBJECT-TYPE
    SYNTAX SEQUENCE OF HostTimeEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of time-ordered host table entries."
    ::= { hosts 3 }

hostTimeEntry OBJECT-TYPE
    SYNTAX HostTimeEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A collection of statistics for a particular host
        that has been discovered on an interface of this
        device. This collection includes the relative
        ordering of the creation time of this object."
    INDEX { hostTimeIndex, hostTimeCreationOrder }
    ::= { hostTimeTable 1 }

HostTimeEntry ::= SEQUENCE {
    hostTimeAddress          OCTET STRING,
    hostTimeCreationOrder    INTEGER (1..65535),
    hostTimeIndex            INTEGER (1..65535),
    hostTimeInPkts           Counter,
    hostTimeOutPkts          Counter,
    hostTimeInOctets         Counter,
    hostTimeOutOctets        Counter,
    hostTimeOutErrors        Counter,
    hostTimeOutBroadcastPkts Counter,
    hostTimeOutMulticastPkts Counter
}

hostTimeAddress OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION

```

"The physical address of this host."
 ::= { hostTimeEntry 1 }

hostTimeCreationOrder OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index that uniquely identifies an entry in the hostTime table among those entries associated with the same hostControlEntry. This index shall be between 1 and N, where N is the value of the associated hostControlTableSize. The ordering of the indexes is based on the order of each entry's insertion into the table, in which entries added earlier have a lower index value than entries added later. Thus the management station has the ability to learn of new entries added to this table without downloading the entire table.

It is important to note that the index for a particular entry may change as an (earlier) entry is deleted from the table. Because this order may change, management stations should make use of the hostControlLastDeleteTime variable in the hostControlEntry associated with the relevant portion of the hostTimeTable. By observing this variable, the management station may detect the circumstances where a download of the table may have missed entries, and where a previous association between a value of hostTimeCreationOrder and a hostTimeEntry may no longer hold."

::= { hostTimeEntry 2 }

hostTimeIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The set of collected host statistics of which this entry is a part. The set of hosts identified by a particular value of this index is associated with the hostControlEntry as identified by the same value of hostControlIndex."

::= { hostTimeEntry 3 }

hostTimeInPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of packets without errors transmitted to
 this address since it was added to the hostTimeTable."
 ::= { hostTimeEntry 4 }

hostTimeOutPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of packets including errors transmitted
 by this address since it was added to the
 hostTimeTable."
 ::= { hostTimeEntry 5 }

hostTimeInOctets OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of octets transmitted to this address
 since it was added to the hostTimeTable (excluding
 framing bits but including FCS octets), except for
 those octets in packets that contained errors."
 ::= { hostTimeEntry 6 }

hostTimeOutOctets OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of octets transmitted by this address since
 it was added to the hostTimeTable (excluding framing
 bits but including FCS octets), including those
 octets in packets that contained errors."
 ::= { hostTimeEntry 7 }

hostTimeOutErrors OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of error packets transmitted by this
 address since this host was added to the

```
        hostTimeTable."
 ::= { hostTimeEntry 8 }

hostTimeOutBroadcastPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of good packets transmitted by this
        address that were directed to the broadcast address
        since this host was added to the hostTimeTable."
    ::= { hostTimeEntry 9 }

hostTimeOutMulticastPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of good packets transmitted by this
        address that were directed to a multicast address
        since this host was added to the hostTimeTable.
        Note that this number does not include packets
        directed to the broadcast address."
    ::= { hostTimeEntry 10 }

-- The Host Top "N" Group

-- Implementation of the Host Top N group is optional.
--
-- The Host Top N group requires the implementation of the
-- host group.
--
-- The Host Top N group is used to prepare reports that
-- describe the hosts that top a list ordered by one of
-- their statistics. The available statistics are samples
-- of one of their base statistics, over an interval
-- specified by the management station. Thus, these
-- statistics are rate based. The management
-- station also selects how many such hosts are reported.

-- The hostTopNControlTable is used to initiate the generation
-- of such a report. The management station may select the
-- parameters of such a report, such as which interface,
-- which statistic, how many hosts, and the start and stop
-- times of the sampling. When the report is prepared,
-- entries are created in the hostTopNTable associated with
-- the relevant hostTopNControlEntry. These entries are
```

```
-- static for each report after it has been prepared.

hostTopNControlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF HostTopNControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of top N host control entries."
    ::= { hostTopN 1 }

hostTopNControlEntry OBJECT-TYPE
    SYNTAX HostTopNControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of parameters that control the creation of a
        report of the top N hosts according to several
        metrics."
    INDEX { hostTopNControlIndex }
    ::= { hostTopNControlTable 1 }

HostTopNControlEntry ::= SEQUENCE {
    hostTopNControlIndex    INTEGER (1..65535),
    hostTopNHostIndex       INTEGER (1..65535),
    hostTopNRateBase        INTEGER,
    hostTopNTimeRemaining   INTEGER,
    hostTopNDuration        INTEGER,
    hostTopNRequestedSize   INTEGER,
    hostTopNGrantedSize     INTEGER,
    hostTopNStartTime       TimeTicks,
    hostTopNOwner            OwnerString,
    hostTopNStatus          INTEGER
}

hostTopNControlIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry
        in the hostTopNControl table.  Each such
        entry defines one top N report prepared for
        one interface."
    ::= { hostTopNControlEntry 1 }

hostTopNHostIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-write
```

STATUS mandatory

DESCRIPTION

"The host table for which a top N report will be prepared on behalf of this entry. The host table identified by a particular value of this index is associated with the same host table as identified by the same value of hostIndex.

This object may not be modified if the associated hostTopNStatus object is equal to valid(1)."

::= { hostTopNControlEntry 2 }

hostTopNRateBase OBJECT-TYPE

SYNTAX INTEGER {
 hostTopNInPkts(1),
 hostTopNOutPkts(2),
 hostTopNInOctets(3),
 hostTopNOutOctets(4),
 hostTopNOutErrors(5),
 hostTopNOutBroadcastPkts(6),
 hostTopNOutMulticastPkts(7)
 }

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The variable for each host that the hostTopNRate variable is based upon.

This object may not be modified if the associated hostTopNStatus object is equal to valid(1)."

::= { hostTopNControlEntry 3 }

hostTopNTimeRemaining OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The number of seconds left in the report currently being collected. When this object is modified by the management station, a new collection is started, possibly aborting a currently running report. The new value is used as the requested duration of this report, which is loaded into the associated hostTopNDuration object.

When this object is set to a non-zero value, any associated hostTopNEntries shall be made inaccessible by the monitor. While the value of this

object is non-zero, it decrements by one per second until it reaches zero. During this time, all associated hostTopNEntries shall remain inaccessible. At the time that this object decrements to zero, the report is made accessible in the hostTopNTable. Thus, the hostTopN table needs to be created only at the end of the collection interval."

DEFVAL { 0 }
::= { hostTopNControlEntry 4 }

hostTopNDuration OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of seconds that this report has collected during the last sampling interval, or if this report is currently being collected, the number of seconds that this report is being collected during this sampling interval.

When the associated hostTopNTimeRemaining object is set, this object shall be set by the probe to the same value and shall not be modified until the next time the hostTopNTimeRemaining is set.

This value shall be zero if no reports have been requested for this hostTopNControlEntry."

DEFVAL { 0 }
::= { hostTopNControlEntry 5 }

hostTopNRequestedSize OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The maximum number of hosts requested for the top N table.

When this object is created or modified, the probe should set hostTopNGrantedSize as closely to this object as is possible for the particular probe implementation and available resources."

DEFVAL { 10 }
::= { hostTopNControlEntry 6 }

`hostTopNGrantedSize OBJECT-TYPE``SYNTAX INTEGER``ACCESS read-only``STATUS mandatory``DESCRIPTION`

"The maximum number of hosts in the top N table.

When the associated `hostTopNRequestedSize` object is created or modified, the probe should set this object as closely to the requested value as is possible for the particular implementation and available resources. The probe must not lower this value except as a result of a set to the associated `hostTopNRequestedSize` object.

Hosts with the highest value of `hostTopNRate` shall be placed in this table in decreasing order of this rate until there is no more room or until there are no more hosts."

::= { `hostTopNControlEntry` 7 }

`hostTopNStartTime OBJECT-TYPE``SYNTAX TimeTicks``ACCESS read-only``STATUS mandatory``DESCRIPTION`

"The value of `sysUpTime` when this top N report was last started. In other words, this is the time that the associated `hostTopNTimeRemaining` object was modified to start the requested report."

::= { `hostTopNControlEntry` 8 }

`hostTopNOwner OBJECT-TYPE``SYNTAX OwnerString``ACCESS read-write``STATUS mandatory``DESCRIPTION`

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { `hostTopNControlEntry` 9 }

`hostTopNStatus OBJECT-TYPE``SYNTAX EntryStatus``ACCESS read-write``STATUS mandatory``DESCRIPTION`

"The status of this `hostTopNControl` entry.

If this object is not equal to valid(1), all associated hostTopNEntries shall be deleted by the agent."

::= { hostTopNControlEntry 10 }

hostTopNTable OBJECT-TYPE
 SYNTAX SEQUENCE OF HostTopNEntry
 ACCESS not-accessible
 STATUS mandatory
 DESCRIPTION
 "A list of top N host entries."
 ::= { hostTopN 2 }

hostTopNEntry OBJECT-TYPE
 SYNTAX HostTopNEntry
 ACCESS not-accessible
 STATUS mandatory
 DESCRIPTION
 "A set of statistics for a host that is part of a top N report."
 INDEX { hostTopNReport, hostTopNIndex }
 ::= { hostTopNTable 1 }

HostTopNEntry ::= SEQUENCE {
 hostTopNReport INTEGER (1..65535),
 hostTopNIndex INTEGER (1..65535),
 hostTopNAddress OCTET STRING,
 hostTopNRate INTEGER
 }

hostTopNReport OBJECT-TYPE
 SYNTAX INTEGER (1..65535)
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "This object identifies the top N report of which this entry is a part. The set of hosts identified by a particular value of this object is part of the same report as identified by the same value of the hostTopNControlIndex object."
 ::= { hostTopNEntry 1 }

hostTopNIndex OBJECT-TYPE
 SYNTAX INTEGER (1..65535)
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "An index that uniquely identifies an entry in

the hostTopN table among those in the same report. This index is between 1 and N, where N is the number of entries in this table. Increasing values of hostTopNIndex shall be assigned to entries with decreasing values of hostTopNRate until index N is assigned to the entry with the lowest value of hostTopNRate or there are no more hostTopNEntries."

```
::= { hostTopNEntry 2 }
```

```
hostTopNAddress OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The physical address of this host."
    ::= { hostTopNEntry 3 }
```

```
hostTopNRate OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The amount of change in the selected variable
        during this sampling interval. The selected
        variable is this host's instance of the object
        selected by hostTopNRateBase."
    ::= { hostTopNEntry 4 }
```

-- The Matrix Group

-- Implementation of the Matrix group is optional.

--

-- The Matrix group consists of the matrixControlTable,
-- matrixSDTable and the matrixDSTable. These tables
-- store statistics for a particular conversation between
-- two addresses. As the device detects a new conversation,
-- including those to a non-unicast address, it creates a
-- new entry in both of the matrix tables.
-- It must only create new entries based on information
-- received in good packets. If the monitoring device finds
-- itself short of resources, it may delete entries as needed.
-- It is suggested that the device delete the least recently
-- used entries first.

```
matrixControlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF MatrixControlEntry
    ACCESS not-accessible
```

STATUS mandatory

DESCRIPTION

"A list of information entries for the traffic matrix on each interface."

::= { matrix 1 }

matrixControlEntry OBJECT-TYPE

SYNTAX MatrixControlEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"Information about a traffic matrix on a particular interface."

INDEX { matrixControlIndex }

::= { matrixControlTable 1 }

MatrixControlEntry ::= SEQUENCE {

matrixControlIndex INTEGER (1..65535),

matrixControlDataSource OBJECT IDENTIFIER,

matrixControlTableSize INTEGER,

matrixControlLastDeleteTime TimeTicks,

matrixControlOwner OwnerString,

matrixControlStatus INTEGER

}

matrixControlIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index that uniquely identifies an entry in the matrixControl table. Each such entry defines a function that discovers conversations on a particular interface and places statistics about them in the matrixSDTable and the matrixDSTable on behalf of this matrixControlEntry."

::= { matrixControlEntry 1 }

matrixControlDataSource OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"This object identifies the source of the data from which this entry creates a traffic matrix. This source can be any interface on this device. In order to identify a particular interface, this object shall identify the instance of the ifIndex object,

defined in [4,6], for the desired interface. For example, if an entry were to receive data from interface #1, this object would be set to ifIndex.1.

The statistics in this group reflect all packets on the local network segment attached to the identified interface.

This object may not be modified if the associated matrixControlStatus object is equal to valid(1)."
::= { matrixControlEntry 2 }

matrixControlTableSize OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of matrixSDEntries in the matrixSDTable for this interface. This must also be the value of the number of entries in the matrixDSTable for this interface."

::= { matrixControlEntry 3 }

matrixControlLastDeleteTime OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The value of sysUpTime when the last entry was deleted from the portion of the matrixSDTable or matrixDSTable associated with this matrixControlEntry.
If no deletions have occurred, this value shall be zero."

::= { matrixControlEntry 4 }

matrixControlOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { matrixControlEntry 5 }

matrixControlStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status of this matrixControl entry.

If this object is not equal to valid(1), all associated entries in the matrixSDTable and the matrixDSTable shall be deleted by the agent."

::= { matrixControlEntry 6 }

matrixSDTable OBJECT-TYPE

SYNTAX SEQUENCE OF MatrixSDEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"A list of traffic matrix entries indexed by source and destination MAC address."

::= { matrix 2 }

matrixSDEntry OBJECT-TYPE

SYNTAX MatrixSDEntry

ACCESS not-accessible

STATUS mandatory

DESCRIPTION

"A collection of statistics for communications between two addresses on a particular interface."

INDEX { matrixSDIndex,
matrixSDSourceAddress, matrixSDDestAddress }

::= { matrixSDTable 1 }

MatrixSDEntry ::= SEQUENCE {

matrixSDSourceAddress	OCTET STRING,
matrixSDDestAddress	OCTET STRING,
matrixSDIndex	INTEGER (1..65535),
matrixSDPkts	Counter,
matrixSDOctets	Counter,
matrixSDErrors	Counter

}

matrixSDSourceAddress OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The source physical address."

::= { matrixSDEntry 1 }

matrixSDDestAddress OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The destination physical address."
 ::= { matrixSDEntry 2 }

matrixSDIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The set of collected matrix statistics of which
 this entry is a part. The set of matrix statistics
 identified by a particular value of this index
 is associated with the same matrixControlEntry
 as identified by the same value of matrixControlIndex."
 ::= { matrixSDEntry 3 }

matrixSDPkts OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of packets transmitted from the source
 address to the destination address (this number
 includes error packets)."
 ::= { matrixSDEntry 4 }

matrixSDOctets OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of octets (excluding framing bits but
 including FCS octets) contained in all packets
 transmitted from the source address to the
 destination address."
 ::= { matrixSDEntry 5 }

matrixSDErrors OBJECT-TYPE
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "The number of error packets transmitted from
 the source address to the destination address."

```

 ::= { matrixSDEntry 6 }

-- Traffic matrix tables from destination to source

matrixDSTable OBJECT-TYPE
    SYNTAX SEQUENCE OF MatrixDSEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of traffic matrix entries indexed by
         destination and source MAC address."
    ::= { matrix 3 }

matrixDSEntry OBJECT-TYPE
    SYNTAX MatrixDSEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A collection of statistics for communications between
         two address on a particular interface."
    INDEX { matrixDSIndex,
            matrixDSDestAddress, matrixDSSourceAddress }
    ::= { matrixDSTable 1 }

MatrixDSEntry ::= SEQUENCE {
    matrixDSSourceAddress      OCTET STRING,
    matrixDSDestAddress        OCTET STRING,
    matrixDSIndex              INTEGER (1..65535),
    matrixDSPkts               Counter,
    matrixDSOctets             Counter,
    matrixDSErrors             Counter
}

matrixDSSourceAddress OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The source physical address."
    ::= { matrixDSEntry 1 }

matrixDSDestAddress OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The destination physical address."

```

```
 ::= { matrixDSEntry 2 }

matrixDSIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The set of collected matrix statistics of which
        this entry is a part. The set of matrix statistics
        identified by a particular value of this index
        is associated with the same matrixControlEntry
        as identified by the same value of matrixControlIndex."
    ::= { matrixDSEntry 3 }

matrixDSPkts OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of packets transmitted from the source
        address to the destination address (this number
        includes error packets)."
    ::= { matrixDSEntry 4 }

matrixDSOctets OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of octets (excluding framing bits
        but including FCS octets) contained in all packets
        transmitted from the source address to the
        destination address."
    ::= { matrixDSEntry 5 }

matrixDSErrors OBJECT-TYPE
    SYNTAX Counter
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of error packets transmitted from
        the source address to the destination address."
    ::= { matrixDSEntry 6 }

-- The Filter Group

-- Implementation of the Filter group is optional.
```

```
--
-- The Filter group allows packets to be captured with an
-- arbitrary filter expression.  A logical data and
-- event stream or "channel" is formed by the packets
-- that match the filter expression.
--
-- This filter mechanism allows the creation of an arbitrary
-- logical expression with which to filter packets.  Each
-- filter associated with a channel is OR'ed with the others.
-- Within a filter, any bits checked in the data and status are
-- AND'ed with respect to other bits in the same filter.  The
-- NotMask also allows for checking for inequality.  Finally,
-- the channelAcceptType object allows for inversion of the
-- whole equation.
--
-- The channel can be turned on or off, and can also
-- generate events when packets pass through it.
```

```
filterTable OBJECT-TYPE
    SYNTAX SEQUENCE OF FilterEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of packet filter entries."
    ::= { filter 1 }
```

```
filterEntry OBJECT-TYPE
    SYNTAX FilterEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of parameters for a packet filter applied on a
        particular interface."
    INDEX { filterIndex }
    ::= { filterTable 1 }
```

```
FilterEntry ::= SEQUENCE {
    filterIndex                INTEGER (1..65535),
    filterChannelIndex         INTEGER (1..65535),
    filterPktDataOffset       INTEGER,
    filterPktData              OCTET STRING,
    filterPktDataMask          OCTET STRING,
    filterPktDataNotMask       OCTET STRING,
    filterPktStatus            INTEGER,
    filterPktStatusMask        INTEGER,
    filterPktStatusNotMask     INTEGER,
    filterOwner                 OwnerString,
    filterStatus                INTEGER
```

```
}

filterIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry
         in the filter table.  Each such entry defines
         one filter that is to be applied to every packet
         received on an interface."
    ::= { filterEntry 1 }

filterChannelIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "This object identifies the channel of which this
         filter is a part.  The filters identified by a
         particular value of this object are associated
         with the same channel as identified by the same
         value of the channelIndex object."
    ::= { filterEntry 2 }

filterPktDataOffset OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The offset from the beginning of each packet where
         a match of packet data will be attempted.  This offset
         is measured from the point in the physical layer
         packet after the framing bits, if any.  For example,
         in an Ethernet frame, this point is at the beginning
         of the destination MAC address.

         This object may not be modified if the associated
         filterStatus object is equal to valid(1)."
```

```
    DEFVAL { 0 }
    ::= { filterEntry 3 }

filterPktData OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The data that is to be matched with the input packet.
```

For each packet received, this filter and the accompanying filterPktDataMask and filterPktDataNotMask will be adjusted for the offset. The only bits relevant to this match algorithm are those that have the corresponding filterPktDataMask bit equal to one. The following three rules are then applied to every packet:

- (1) If the packet is too short and does not have data corresponding to part of the filterPktData, the packet will fail this data match.
- (2) For each relevant bit from the packet with the corresponding filterPktDataNotMask bit set to zero, if the bit from the packet is not equal to the corresponding bit from the filterPktData, then the packet will fail this data match.
- (3) If for every relevant bit from the packet with the corresponding filterPktDataNotMask bit set to one, the bit from the packet is equal to the corresponding bit from the filterPktData, then the packet will fail this data match.

Any packets that have not failed any of the three matches above have passed this data match.

This object may not be modified if the associated filterStatus object is equal to valid(1)."

::= { filterEntry 4 }

filterPktDataMask OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The mask that is applied to the match process. After adjusting this mask for the offset, only those bits in the received packet that correspond to bits set in this mask are relevant for further processing by the match algorithm. The offset is applied to filterPktDataMask in the same way it is applied to the filter. For the purposes of the matching algorithm, if the associated filterPktData object is longer than this mask, this mask is conceptually extended with '1' bits until it reaches the length of the filterPktData object.

This object may not be modified if the associated filterStatus object is equal to valid(1)."
::= { filterEntry 5 }

filterPktDataNotMask OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The inversion mask that is applied to the match process. After adjusting this mask for the offset, those relevant bits in the received packet that correspond to bits cleared in this mask must all be equal to their corresponding bits in the filterPktData object for the packet to be accepted. In addition, at least one of those relevant bits in the received packet that correspond to bits set in this mask must be different to its corresponding bit in the filterPktData object.

For the purposes of the matching algorithm, if the associated filterPktData object is longer than this mask, this mask is conceptually extended with '0' bits until it reaches the length of the filterPktData object.

This object may not be modified if the associated filterStatus object is equal to valid(1)."
::= { filterEntry 6 }

filterPktStatus OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status that is to be matched with the input packet. The only bits relevant to this match algorithm are those that have the corresponding filterPktStatusMask bit equal to one.

The following two rules are then applied to every packet:

- (1) For each relevant bit from the packet status with the corresponding filterPktStatusNotMask bit set to zero, if the bit from the packet status is not equal to the corresponding bit from the filterPktStatus, then the packet will

fail this status match.

- (2) If for every relevant bit from the packet status with the corresponding filterPktStatusNotMask bit set to one, the bit from the packet status is equal to the corresponding bit from the filterPktStatus, then the packet will fail this status match.

Any packets that have not failed either of the two matches above have passed this status match.

The value of the packet status is a sum. This sum initially takes the value zero. Then, for each error, E, that has been discovered in this packet, 2 raised to a value representing E is added to the sum. The errors and the bits that represent them are dependent on the media type of the interface that this channel is receiving packets from.

The errors defined for a packet captured off of an Ethernet interface are as follows:

bit #	Error
0	Packet is longer than 1518 octets
1	Packet is shorter than 64 octets
2	Packet experienced a CRC or Alignment error

For example, an Ethernet fragment would have a value of 6 ($2^1 + 2^2$).

As this MIB is expanded to new media types, this object will have other media-specific errors defined.

For the purposes of this status matching algorithm, if the packet status is longer than this object, filterPktStatus this object is conceptually extended with '0' bits until it reaches the size of the packet status.

This object may not be modified if the associated filterStatus object is equal to valid(1)."

::= { filterEntry 7 }

filterPktStatusMask OBJECT-TYPE
 SYNTAX INTEGER
 ACCESS read-write

STATUS mandatory

DESCRIPTION

"The mask that is applied to the status match process. Only those bits in the received packet that correspond to bits set in this mask are relevant for further processing by the status match algorithm. For the purposes of the matching algorithm, if the associated filterPktStatus object is longer than this mask, this mask is conceptually extended with '1' bits until it reaches the size of the filterPktStatus. In addition, if a packet status is longer than this mask, this mask is conceptually extended with '0' bits until it reaches the size of the packet status.

This object may not be modified if the associated filterStatus object is equal to valid(1)."

::= { filterEntry 8 }

filterPktStatusNotMask OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The inversion mask that is applied to the status match process. Those relevant bits in the received packet status that correspond to bits cleared in this mask must all be equal to their corresponding bits in the filterPktStatus object for the packet to be accepted. In addition, at least one of those relevant bits in the received packet status that correspond to bits set in this mask must be different to its corresponding bit in the filterPktStatus object for the packet to be accepted.

For the purposes of the matching algorithm, if the associated filterPktStatus object or a packet status is longer than this mask, this mask is conceptually extended with '0' bits until it reaches the longer of the lengths of the filterPktStatus object and the packet status.

This object may not be modified if the associated filterStatus object is equal to valid(1)."

::= { filterEntry 9 }

filterOwner OBJECT-TYPE

SYNTAX OwnerString

```

ACCESS read-write
STATUS mandatory
DESCRIPTION
    "The entity that configured this entry and is
    therefore using the resources assigned to it."
::= { filterEntry 10 }

filterStatus OBJECT-TYPE
    SYNTAX EntryStatus
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The status of this filter entry."
    ::= { filterEntry 11 }

channelTable OBJECT-TYPE
    SYNTAX SEQUENCE OF ChannelEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of packet channel entries."
    ::= { filter 2 }

channelEntry OBJECT-TYPE
    SYNTAX ChannelEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of parameters for a packet channel applied on a
        particular interface."
    INDEX { channelIndex }
    ::= { channelTable 1 }

ChannelEntry ::= SEQUENCE {
    channelIndex                INTEGER (1..65535),
    channelIfIndex              INTEGER (1..65535),
    channelAcceptType            INTEGER,
    channelDataControl           INTEGER,
    channelTurnOnEventIndex      INTEGER (0..65535),
    channelTurnOffEventIndex     INTEGER (0..65535),
    channelEventIndex            INTEGER (0..65535),
    channelEventStatus           INTEGER,
    channelMatches               Counter,
    channelDescription            DisplayString (SIZE (0..127)),
    channelOwner                 OwnerString,
    channelStatus                INTEGER
}

```

channelIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index that uniquely identifies an entry in the channel table. Each such entry defines one channel, a logical data and event stream."

::= { channelEntry 1 }

channelIfIndex OBJECT-TYPE

SYNTAX INTEGER (1..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The value of this object uniquely identifies the interface on this remote network monitoring device to which the associated filters are applied to allow data into this channel. The interface identified by a particular value of this object is the same interface as identified by the same value of the ifIndex object, defined in [4,6]. The filters in this group are applied to all packets on the local network segment attached to the identified interface."

This object may not be modified if the associated channelStatus object is equal to valid(1)."

::= { channelEntry 2 }

channelAcceptType OBJECT-TYPE

SYNTAX INTEGER {
 acceptMatched(1),
 acceptFailed(2)
}

ACCESS read-write

STATUS mandatory

DESCRIPTION

"This object controls the action of the filters associated with this channel. If this object is equal to acceptMatched(1), packets will be accepted to this channel if they are accepted by both the packet data and packet status matches of an associated filter. If this object is equal to acceptFailed(2), packets will be accepted to this channel only if they fail either the packet data match or the packet status match of each of the associated filters."

This object may not be modified if the associated
channelStatus object is equal to valid(1)."
::= { channelEntry 3 }

channelDataControl OBJECT-TYPE

SYNTAX INTEGER {
 on(1),
 off(2)
}

ACCESS read-write

STATUS mandatory

DESCRIPTION

"This object controls the flow of data through this
channel. If this object is on(1), data, status and
events flow through this channel. If this object is
off(2), data, status and events will not flow through
this channel."

DEFVAL { off }

::= { channelEntry 4 }

channelTurnOnEventIndex OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The value of this object identifies the event
that is configured to turn the associated
channelDataControl from off to on when the event is
generated. The event identified by a particular value
of this object is the same event as identified by the
same value of the eventIndex object. If there is no
corresponding entry in the eventTable, then no
association exists. In fact, if no event is intended
for this channel, channelTurnOnEventIndex must be
set to zero, a non-existent event index."

This object may not be modified if the associated
channelStatus object is equal to valid(1)."

::= { channelEntry 5 }

channelTurnOffEventIndex OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The value of this object identifies the event
that is configured to turn the associated
channelDataControl from on to off when the event is

generated. The event identified by a particular value of this object is the same event as identified by the same value of the eventIndex object. If there is no corresponding entry in the eventTable, then no association exists. In fact, if no event is intended for this channel, channelTurnOffEventIndex must be set to zero, a non-existent event index.

This object may not be modified if the associated channelStatus object is equal to valid(1)."

::= { channelEntry 6 }

channelEventIndex OBJECT-TYPE

SYNTAX INTEGER (0..65535)

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The value of this object identifies the event that is configured to be generated when the associated channelDataControl is on and a packet is matched. The event identified by a particular value of this object is the same event as identified by the same value of the eventIndex object. If there is no corresponding entry in the eventTable, then no association exists. In fact, if no event is intended for this channel, channelEventIndex must be set to zero, a non-existent event index.

This object may not be modified if the associated channelStatus object is equal to valid(1)."

::= { channelEntry 7 }

channelEventStatus OBJECT-TYPE

SYNTAX INTEGER {
 eventReady(1),
 eventFired(2),
 eventAlwaysReady(3)
}

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The event status of this channel.

If this channel is configured to generate events when packets are matched, a means of controlling the flow of those events is often needed. When this object is equal to eventReady(1), a single event may be generated, after which this object

will be set by the probe to eventFired(2). While in the eventFired(2) state, no events will be generated until the object is modified to eventReady(1) (or eventAlwaysReady(3)). The management station can thus easily respond to a notification of an event by re-enabling this object.

If the management station wishes to disable this flow control and allow events to be generated at will, this object may be set to eventAlwaysReady(3). Disabling the flow control is discouraged as it can result in high network traffic or other performance problems."

```
DEFVAL { eventReady }  
::= { channelEntry 8 }
```

channelMatches OBJECT-TYPE

SYNTAX Counter

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of times this channel has matched a packet. Note that this object is updated even when channelDataControl is set to off."

```
::= { channelEntry 9 }
```

channelDescription OBJECT-TYPE

SYNTAX DisplayString (SIZE (0..127))

ACCESS read-write

STATUS mandatory

DESCRIPTION

"A comment describing this channel."

```
::= { channelEntry 10 }
```

channelOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

```
::= { channelEntry 11 }
```

channelStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

```

        "The status of this channel entry."
 ::= { channelEntry 12 }

-- The Packet Capture Group

-- Implementation of the Packet Capture group is optional.
--
-- The Packet Capture Group requires implementation of the
-- Filter Group.
--
-- The Packet Capture group allows packets to be captured
-- upon a filter match. The bufferControlTable controls
-- the captured packets output from a channel that is
-- associated with it. The captured packets are placed
-- in entries in the captureBufferTable. These entries are
-- associated with the bufferControlEntry on whose behalf they
-- were stored.

bufferControlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF BufferControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of buffers control entries."
    ::= { capture 1 }

bufferControlEntry OBJECT-TYPE
    SYNTAX BufferControlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of parameters that control the collection of
        a stream of packets that have matched filters."
    INDEX { bufferControlIndex }
    ::= { bufferControlTable 1 }

BufferControlEntry ::= SEQUENCE {
    bufferControlIndex                INTEGER (1..65535),
    bufferControlChannelIndex         INTEGER (1..65535),
    bufferControlFullStatus            INTEGER,
    bufferControlFullAction            INTEGER,
    bufferControlCaptureSliceSize     INTEGER,
    bufferControlDownloadSliceSize    INTEGER,
    bufferControlDownloadOffset       INTEGER,
    bufferControlMaxOctetsRequested    INTEGER,
    bufferControlMaxOctetsGranted     INTEGER,
    bufferControlCapturedPackets     INTEGER,

```

```

    bufferControlTurnOnTime      TimeTicks,
    bufferControlOwner           OwnerString,
    bufferControlStatus          INTEGER
}

bufferControlIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry
        in the bufferControl table. The value of this
        index shall never be zero. Each such
        entry defines one set of packets that is
        captured and controlled by one or more filters."
    ::= { bufferControlEntry 1 }

```

```

bufferControlChannelIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "An index that identifies the channel that is the
        source of packets for this bufferControl table.
        The channel identified by a particular value of this
        index is the same as identified by the same value of
        the channelIndex object.

        This object may not be modified if the associated
        bufferControlStatus object is equal to valid(1)."
    ::= { bufferControlEntry 2 }

```

```

bufferControlFullStatus OBJECT-TYPE
    SYNTAX INTEGER {
        spaceAvailable(1),
        full(2)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This object shows whether the buffer has room to
        accept new packets or if it is full.

        If the status is spaceAvailable(1), the buffer is
        accepting new packets normally. If the status is
        full(2) and the associated bufferControlFullAction
        object is wrapWhenFull, the buffer is accepting new
        packets by deleting enough of the oldest packets

```

to make room for new ones as they arrive. Otherwise, if the status is full(2) and the bufferControlFullAction object is lockWhenFull, then the buffer has stopped collecting packets.

When this object is set to full(2) the probe must not later set it to spaceAvailable(1) except in the case of a significant gain in resources such as an increase of bufferControlOctetsGranted. In particular, the wrap-mode action of deleting old packets to make room for newly arrived packets must not affect the value of this object."

::= { bufferControlEntry 3 }

bufferControlFullAction OBJECT-TYPE

SYNTAX INTEGER {
 lockWhenFull(1),
 wrapWhenFull(2) -- FIFO
}

ACCESS read-write

STATUS mandatory

DESCRIPTION

"Controls the action of the buffer when it reaches the full status. When in the lockWhenFull(1) state a packet is added to the buffer that fills the buffer, the bufferControlFullStatus will be set to full(2) and this buffer will stop capturing packets."

::= { bufferControlEntry 4 }

bufferControlCaptureSliceSize OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The maximum number of octets of each packet that will be saved in this capture buffer. For example, if a 1500 octet packet is received by the probe and this object is set to 500, then only 500 octets of the packet will be stored in the associated capture buffer. If this variable is set to 0, the capture buffer will save as many octets as is possible.

This object may not be modified if the associated bufferControlStatus object is equal to valid(1)."

DEFVAL { 100 }

::= { bufferControlEntry 5 }

bufferControlDownloadSliceSize OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The maximum number of octets of each packet in this capture buffer that will be returned in an SNMP retrieval of that packet. For example, if 500 octets of a packet have been stored in the associated capture buffer, the associated bufferControlDownloadOffset is 0, and this object is set to 100, then the captureBufferPacket object that contains the packet will contain only the first 100 octets of the packet.

A prudent manager will take into account possible interoperability or fragmentation problems that may occur if the download slice size is set too large. In particular, conformant SNMP implementations are not required to accept messages whose length exceeds 484 octets, although they are encouraged to support larger datagrams whenever feasible."

DEFVAL { 100 }

::= { bufferControlEntry 6 }

bufferControlDownloadOffset OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The offset of the first octet of each packet in this capture buffer that will be returned in an SNMP retrieval of that packet. For example, if 500 octets of a packet have been stored in the associated capture buffer and this object is set to 100, then the captureBufferPacket object that contains the packet will contain bytes starting 100 octets into the packet."

DEFVAL { 0 }

::= { bufferControlEntry 7 }

bufferControlMaxOctetsRequested OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The requested maximum number of octets to be saved in this captureBuffer, including any

implementation-specific overhead. If this variable is set to -1, the capture buffer will save as many octets as is possible.

When this object is created or modified, the probe should set `bufferControlMaxOctetsGranted` as closely to this object as is possible for the particular probe implementation and available resources. However, if the object has the special value of -1, the probe must set `bufferControlMaxOctetsGranted` to -1."

```
DEFVAL { -1 }  
::= { bufferControlEntry 8 }
```

`bufferControlMaxOctetsGranted` OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The maximum number of octets that can be saved in this `captureBuffer`, including overhead. If this variable is -1, the capture buffer will save as many octets as possible.

When the `bufferControlMaxOctetsRequested` object is created or modified, the probe should set this object as closely to the requested value as is possible for the particular probe implementation and available resources. However, if the request object has the special value of -1, the probe must set this object to -1. The probe must not lower this value except as a result of a modification to the associated `bufferControlMaxOctetsRequested` object.

When this maximum number of octets is reached and a new packet is to be added to this capture buffer and the corresponding `bufferControlFullAction` is set to `wrapWhenFull(2)`, enough of the oldest packets associated with this capture buffer shall be deleted by the agent so that the new packet can be added. If the corresponding `bufferControlFullAction` is set to `lockWhenFull(1)`, the new packet shall be discarded. In either case, the probe must set `bufferControlFullStatus` to `full(2)`.

When the value of this object changes to a value less than the current value, entries are deleted from the `captureBufferTable` associated with this

bufferControlEntry. Enough of the oldest of these captureBufferEntries shall be deleted by the agent so that the number of octets used remains less than or equal to the new value of this object.

When the value of this object changes to a value greater than the current value, the number of associated captureBufferEntries may be allowed to grow."

::= { bufferControlEntry 9 }

bufferControlCapturedPackets OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of packets currently in this captureBuffer."

::= { bufferControlEntry 10 }

bufferControlTurnOnTime OBJECT-TYPE

SYNTAX TimeTicks

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The value of sysUpTime when this capture buffer was first turned on."

::= { bufferControlEntry 11 }

bufferControlOwner OBJECT-TYPE

SYNTAX OwnerString

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The entity that configured this entry and is therefore using the resources assigned to it."

::= { bufferControlEntry 12 }

bufferControlStatus OBJECT-TYPE

SYNTAX EntryStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

"The status of this buffer Control Entry."

::= { bufferControlEntry 13 }

captureBufferTable OBJECT-TYPE

SYNTAX SEQUENCE OF CaptureBufferEntry

ACCESS not-accessible

STATUS mandatory
 DESCRIPTION
 "A list of packets captured off of a channel."
 ::= { capture 2 }

captureBufferEntry OBJECT-TYPE
 SYNTAX CaptureBufferEntry
 ACCESS not-accessible
 STATUS mandatory
 DESCRIPTION
 "A packet captured off of an attached network."
 INDEX { captureBufferControlIndex, captureBufferIndex }
 ::= { captureBufferTable 1 }

CaptureBufferEntry ::= SEQUENCE {
 captureBufferControlIndex INTEGER (1..65535),
 captureBufferIndex INTEGER,
 captureBufferPacketID INTEGER,
 captureBufferPacketData OCTET STRING,
 captureBufferPacketLength INTEGER,
 captureBufferPacketTime INTEGER,
 captureBufferPacketStatus INTEGER
 }

captureBufferControlIndex OBJECT-TYPE
 SYNTAX INTEGER (1..65535)
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "The index of the bufferControlEntry with which
 this packet is associated."
 ::= { captureBufferEntry 1 }

captureBufferIndex OBJECT-TYPE
 SYNTAX INTEGER
 ACCESS read-only
 STATUS mandatory
 DESCRIPTION
 "An index that uniquely identifies an entry
 in the captureBuffer table associated with a
 particular bufferControlEntry. This index will
 start at 1 and increase by one for each new packet
 added with the same captureBufferControlIndex."
 ::= { captureBufferEntry 2 }

captureBufferPacketID OBJECT-TYPE
 SYNTAX INTEGER
 ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index that describes the order of packets that are received on a particular interface. The packetID of a packet captured on an interface is defined to be greater than the packetID's of all packets captured previously on the same interface. As the captureBufferPacketID object has a maximum positive value of $2^{31} - 1$, any captureBufferPacketID object shall have the value of the associated packet's packetID mod 2^{31} ."

::= { captureBufferEntry 3 }

captureBufferPacketData OBJECT-TYPE

SYNTAX OCTET STRING

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The data inside the packet, starting at the beginning of the packet plus any offset specified in the associated bufferControlDownloadOffset, including any link level headers. The length of the data in this object is the minimum of the length of the captured packet minus the offset, the length of the associated bufferControlCaptureSliceSize minus the offset, and the associated bufferControlDownloadSliceSize. If this minimum is less than zero, this object shall have a length of zero."

::= { captureBufferEntry 4 }

captureBufferPacketLength OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The actual length (off the wire) of the packet stored in this entry, including FCS octets."

::= { captureBufferEntry 5 }

captureBufferPacketTime OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The number of milliseconds that had passed since this capture buffer was first turned on when this packet was captured."

::= { captureBufferEntry 6 }

captureBufferPacketStatus OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"A value which indicates the error status of this packet.

The value of this object is defined in the same way as filterPacketStatus. The value is a sum. This sum initially takes the value zero. Then, for each error, E, that has been discovered in this packet, 2 raised to a value representing E is added to the sum.

The errors defined for a packet captured off of an Ethernet interface are as follows:

bit #	Error
0	Packet is longer than 1518 octets
1	Packet is shorter than 64 octets
2	Packet experienced a CRC or Alignment error
3	First packet in this capture buffer after it was detected that some packets were not processed correctly.

For example, an Ethernet fragment would have a value of 6 ($2^1 + 2^2$).

As this MIB is expanded to new media types, this object will have other media-specific errors defined."

::= { captureBufferEntry 7 }

-- The Event Group

-- Implementation of the Event group is optional.

--

-- The Event group controls the generation and notification of events from this device. Each entry in the eventTable describes the parameters of the event that can be triggered. Each event entry is fired by an associated condition located elsewhere in the MIB. An event entry may also be associated with a function elsewhere in the MIB that will be executed when the event is generated. For example, a channel may be turned on or off by the firing of an event.

--

-- Each eventEntry may optionally specify that a log entry

```
-- be created on its behalf whenever the event occurs.
-- Each entry may also specify that notification should
-- occur by way of SNMP trap messages.  In this case, the
-- community for the trap message is given in the associated
-- eventCommunity object.  The enterprise and specific trap
-- fields of the trap are determined by the condition that
-- triggered the event.  Three traps are defined in a companion
-- document: risingAlarm, fallingAlarm, and packetMatch.
-- If the eventTable is triggered by a condition specified
-- elsewhere, the enterprise and specific trap fields
-- must be specified for traps generated for that condition.
```

```
eventTable OBJECT-TYPE
    SYNTAX SEQUENCE OF EventEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of events to be generated."
    ::= { event 1 }
```

```
eventEntry OBJECT-TYPE
    SYNTAX EventEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of parameters that describe an event to be
        generated when certain conditions are met."
    INDEX { eventIndex }
    ::= { eventTable 1 }
```

```
EventEntry ::= SEQUENCE {
    eventIndex          INTEGER (1..65535),
    eventDescription    DisplayString (SIZE (0..127)),
    eventType           INTEGER,
    eventCommunity      OCTET STRING (SIZE (0..127)),
    eventLastTimeSent   TimeTicks,
    eventOwner          OwnerString,
    eventStatus         INTEGER
}
```

```
eventIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry in the
        event table.  Each such entry defines one event that
        is to be generated when the appropriate conditions
```

```
        occur."
 ::= { eventEntry 1 }

eventDescription OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..127))
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "A comment describing this event entry."
 ::= { eventEntry 2 }

eventType OBJECT-TYPE
    SYNTAX INTEGER {
        none(1),
        log(2),
        snmp-trap(3),      -- send an SNMP trap
        log-and-trap(4)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The type of notification that the probe will make
         about this event.  In the case of log, an entry is
         made in the log table for each event.  In the case of
         snmp-trap, an SNMP trap is sent to one or more
         management stations."
 ::= { eventEntry 3 }

eventCommunity OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE (0..127))
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "If an SNMP trap is to be sent, it will be sent to
         the SNMP community specified by this octet string.
         In the future this table will be extended to include
         the party security mechanism.  This object shall be
         set to a string of length zero if it is intended that
         that mechanism be used to specify the destination of
         the trap."
 ::= { eventEntry 4 }

eventLastTimeSent OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of sysUpTime at the time this event
```

```
        entry last generated an event.  If this entry has
        not generated any events, this value will be
        zero."
 ::= { eventEntry 5 }

eventOwner OBJECT-TYPE
    SYNTAX OwnerString
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The entity that configured this entry and is therefore
        using the resources assigned to it.

        If this object contains a string starting with 'monitor'
        and has associated entries in the log table, all
        connected management stations should retrieve those
        log entries, as they may have significance to all
        management stations connected to this device"
 ::= { eventEntry 6 }

eventStatus OBJECT-TYPE
    SYNTAX EntryStatus
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The status of this event entry.

        If this object is not equal to valid(1), all associated
        log entries shall be deleted by the agent."
 ::= { eventEntry 7 }

--
logTable OBJECT-TYPE
    SYNTAX SEQUENCE OF LogEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A list of events that have been logged."
 ::= { event 2 }

logEntry OBJECT-TYPE
    SYNTAX LogEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A set of data describing an event that has been
        logged."
    INDEX { logEventIndex, logIndex }
```

```
 ::= { logTable 1 }

LogEntry ::= SEQUENCE {
    logEventIndex      INTEGER (1..65535),
    logIndex           INTEGER,
    logTime            TimeTicks,
    logDescription     DisplayString (SIZE (0..255))
}

logEventIndex OBJECT-TYPE
    SYNTAX INTEGER (1..65535)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The event entry that generated this log
        entry. The log identified by a particular
        value of this index is associated with the same
        eventEntry as identified by the same value
        of eventIndex."
    ::= { logEntry 1 }

logIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An index that uniquely identifies an entry
        in the log table amongst those generated by the
        same eventEntries. These indexes are
        assigned beginning with 1 and increase by one
        with each new log entry. The association
        between values of logIndex and logEntries
        is fixed for the lifetime of each logEntry.
        The agent may choose to delete the oldest
        instances of logEntry as required because of
        lack of memory. It is an implementation-specific
        matter as to when this deletion may occur."
    ::= { logEntry 2 }

logTime OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of sysUpTime when this log entry was
        created."
    ::= { logEntry 3 }
```

```
logDescription OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "An implementation dependent description of the
         event that activated this log entry."
    ::= { logEntry 4 }
```

END

7. Acknowledgments

This document was produced by the IETF Remote Network Monitoring Working Group.

In addition, the comments of the following individuals are also acknowledged:

Anne Ambler	Spider Systems
Steve Bostock	Novell
Gigi Chu	Hewlett-Packard
Chuck Davin	MIT
Gary Ellis	Hewlett-Packard
Mike Erlinger	Lexcel
Stephen Grau	Novell
Martin Gray	Spider Systems
Mark Hoerth	Hewlett-Packard
Tim Lee-Thorp	Network General
Anil Singhal	Frontier Software
David Stevens	Network General
Gerard White	Concord Communications

8. 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", 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", 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", 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] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.
- [10] Rose, M., Editor, "A Convention for Defining Traps for use with the SNMP", RFC 1215, Performance Systems International, March 1991.

Security Considerations

Security issues are not discussed in this memo.

Author's Address

Steven Waldbusser
Carnegie Mellon University
4910 Forbes Ave.
Pittsburgh, PA 15213

Phone: (412) 268-6628

EMail: waldbusser@andrew.cmu.edu