CHAPTER 4

Remote Monitoring MIB

The network management module (NMM) SNMP agent implements a remote monitoring probe to support the four basic groups of RFC-1757 RMON MIB. The probe, or agent code, only monitors the FastHub backplane as represented by the in-band Ethernet management interface.

The SNMP RMON agent implements the first four groups and the Event group in support of the Alarm group. The following group definitions are taken from RFC 1757.

- Ethernet Statistics Group: The Ethernet statistics group contains statistics measured by the probe for each monitored Ethernet interface on this device. This group consists of the etherStatsTable.
- History Control Group: The history control group controls the periodic statistical sampling of data from various types of networks. This group consists of the historyControlTable.
- Ethernet History Group: The Ethernet history group records periodic statistical samples from an Ethernet network and stores them for later retrieval. This group consists of the etherHistoryTable.
- 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.
- Event Group: The event group controls the generation and notification of events from this device. This group consists of the eventTable and the logTable.

Ethernet Statistics Group

etherStatsTable

At startup, the FastHub creates and maintains a default etherStatsTable entry for the repeater. The owner string for the entry contains *monitor.* A management application requiring the running Ethernet statistics of the FastHub can perform get-next operations on this table for the applicable etherStatsDataSource.

etherStatsIndex (integer [1 to 65535])

The value of this read-only MIB object uniquely identifies this etherStats entry.

Initial Value: Unique value assigned by owner (creator) of this entry.

etherStatsDataSource (object identifier)

This read-write MIB object identifies the source of the data that this etherStats entry is configured to analyze. This source can be any Ethernet interface on the FastHub. To identify a particular interface, this object identifies the instance of the ifIndex object, defined in RFC 1213 and RFC 1573, for the desired interface. For example, if an entry were to receive data from interface number 1 (such as port 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.

An agent might not be able to tell if fundamental changes to the media of the interface have occurred and necessitate an invalidation of this entry. For example, a hot-insertable Ethernet card could be pulled out and replaced by a Token Ring card. In such a case, if the agent has knowledge of the change, it is recommended that the entry be invalidated.

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

Initial Value: Applicable interface object identifier for the in-band management interface.

etherStatsDropEvents (counter)

This read-only MIB object displays the total number of events in which packets were dropped by the probe due to lack of resources.

Note This number is not necessarily the number of packets dropped; it is the number of times this condition has been detected.

Initial Value: 0

etherStatsOctets (counter)

This read-only MIB object displays the total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including frame check sequence [FCS] octets).

This object provides a reasonable estimate of Ethernet utilization. If greater precision is desired, sample the etherStatsPkts and etherStatsOctets objects before and after a common interval. The differences in the sampled values are Pkts and Octets, respectively, and the number of seconds in the interval is Interval. These values are used to calculate the utilization as follows:

$$\text{Utilization} = \frac{\text{Pkts} * (9.6 + 6.4) + (\text{Octets} * .8)}{\text{Interval} * 10{,}000}$$

The result of this equation is the value Utilization, which is the percent of utilization of the Ethernet segment on a scale of 0 to 100 percent.

Initial Value: 0

etherStatsPkts (counter)

This read-only MIB object displays the total number of packets (including bad packets, broadcast packets, and multicast packets) received.

Initial Value: 0

etherStatsBroadcastPkts (counter)

This read-only MIB object displays the total number of good packets received that were directed to the broadcast address (this does not include multicast packets).

Initial Value: 0

etherStatsMulticastPkts (counter)

This read-only MIB object displays the total number of good packets received that were directed to a multicast address (this number does not include packets directed to the broadcast address).

Initial Value: 0

etherStatsCRCAlignErrors (counter)

This read-only MIB object displays 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 had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Initial Value: 0

etherStatsUndersizePkts (counter)

This read-only MIB object displays 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.

Initial Value: 0

etherStatsOversizePkts (counter)

This read-only MIB object displays the total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.

Initial Value: 0

etherStatsFragments (counter)

This read-only MIB object displays the total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Note It is normal for etherStatsFragments to increment. This is because it counts both runts (normal occurrences because of collisions) and noise hits.

Initial Value: 0

etherStatsJabbers (counter)

This read-only MIB object displays the total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Note This definition of jabber is different than the definition in IEEE 802.3, section 8.2.1.5 (10Base5) and section 10.3.1.4 (10Base2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowable range to detect jabber is between 20 ms and 150 ms.

Initial Value: 0

etherStatsCollisions (counter)

This read-only MIB object displays the best estimate of the total number of collisions on this Ethernet segment.

The value returned depends on the location of the RMON probe. IEEE 802.3, section 8.2.1.3 (10Base5) and section 10.3.1.3 (10Base2), states that a station must detect a collision in the receive mode if three or more stations are transmitting simultaneously. A repeater port must detect a collision when two or more stations are transmitting simultaneously. Thus a probe placed on a repeater port could record more collisions than a probe connected to a station on the same segment.

Probe location plays a much smaller role when considering 10BaseT. IEEE 802.3, section 14.2.1.4 (10BaseT), defines a collision as the simultaneous presence of signals on the DO and RD circuits (transmitting and receiving at the same time). A 10BaseT station can only detect collisions when it is transmitting. Thus probes placed on a station and a repeater should report the same number of collisions.

Note An RMON probe inside a repeater should ideally report collisions between the repeater and one or more other hosts (transmit collisions as defined by IEEE 802.3k) and receiver collisions observed on any coaxial segments to which the repeater is connected.

Initial Value: 0

etherStatsPkts64Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were 64 octets in length excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsPkts65to127Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were between 65 and 127 octets in length, inclusive, excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsPkts128to255Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were between 128 and 255 octets in length, inclusive, excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsPkts256to511Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were between 256 and 511 octets in length, inclusive, excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsPkts512to1023Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were between 512 and 1023 octets in length, inclusive, excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsPkts1024to1518Octets (counter)

This read-only MIB object displays the total number of packets (including bad packets) received that were between 1024 and 1518 octets in length, inclusive, excluding framing bits but including FCS octets.

Initial Value: 0

etherStatsOwner (ownerstring)

This read-write MIB object specifies the entity that configured this entry and is therefore using the resources assigned to it.

Initial Value: String assigned by owner of this entry.

etherStatsStatus (EntryStatus)

This read-write MIB object specifies the status of this etherStats entry.

Initial Value: Assigned by owner of this entry.

History Control Group

From RFC 1757:

“The history control group controls the periodic statistical sampling of data from various types of networks. The historyControlTable stores configuration entries that 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 entry defines one sample and is associated with the historyControlEntry that caused the sample to be taken. Each counter in the etherHistoryEntry counts the same event as its similarly-named counterpart in the etherStatsEntry, except that each value here is a cumulative sum during a sampling period.

“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 probe 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."

The FastHub RMON agent creates two history control entries as described above.

historyControlTable

historyControlIndex (integer [1 to 65535])

This read-only MIB object displays an index that uniquely identifies an entry in the historyControl table. Each entry defines a set of samples at a particular interval for an interface on the device.

Initial Value: Unique value assigned by owner (creator) of this entry.

historyControlDataSource (object identifier)

This read-write MIB object identifies the source of the data from 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 RFC 1213 and RFC 1573, for the desired interface. For example, if an entry were to receive data from interface number 1 (such as port 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.

An agent might not be able to tell if fundamental changes to the media of the interface have occurred and necessitate an invalidation of this entry. For example, a hot-insertable Ethernet card could be pulled out and replaced by a token-ring card. In such a case, if the agent has knowledge of the change, it is recommended that the entry be invalidated.

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

Initial Value: Applicable interface object identifier for the in-band management interface.

historyControlBucketsRequested (integer [1 to 65535])

This read-write MIB object specifies the requested number of discrete time intervals over which data is saved in the part of the media-specific table associated with this historyControlEntry.

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

Default Value: 50

historyControlBucketsGranted (integer [1 to 65535])

This read-only MIB object displays the number of discrete sampling intervals over which data is saved in the part of the media-specific table associated with this historyControlEntry.

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 are 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 is 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 is 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. The agent deletes enough of the older entries 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 is allowed to grow.

Initial Value: 50

historyControlInterval (integer [1 to 3600])

This read-write MIB object specifies the interval in seconds over which the data is sampled for each bucket in the part of the media-specific table associated with this historyControlEntry. This interval can be set to any number of seconds between 1 and 3600 (1 hour).

Because the counters in a bucket can overflow at their maximum value with no indication, you must 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 of importance 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 cannot be modified if the associated historyControlStatus object is equal to valid (1).

Initial Value: 1800 (30 minutes), 30 for the first *monitor* owned entry.

historyControlOwner (ownerstring)

This read-write MIB object specifies the entity that configured this entry and is therefore using the resources assigned to it.

Initial Value: Unique value assigned by owner (creator) of this entry.

historyControlStatus (EntryStatus)

This read-write MIB object specifies the status of this historyControl entry.

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

Initial Value: Assigned by owner of this entry.

Ethernet History Group

From RFC 1757:

“The Ethernet History group records periodic statistical samples from a network and stores them for later retrieval. Once samples are taken, their data is stored in an entry in a media-specific table. Each entry defines one sample and is associated with the historyControlEntry that caused the sample to be taken. This group defines the etherHistoryTable for Ethernet networks.”

etherHistoryTable

etherHistoryIndex (integer [1 to 65535])

The history of which this read-only MIB object 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.

Initial Value: Applicable etherHistoryControlindex

etherHistorySampleIndex (integer [1 to 2147483647])

This read-only MIB object is 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.

Initial Value: Variable

etherHistoryIntervalStart (time tick)

This read-only MIB object is 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 the start of an hour. Following this rule might 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 that is currently being collected is not accessible in this table until the end of its interval.

Initial Value: Applicable value of sysUpTime

etherHistoryDropEvents (counter)

This read-only MIB object displays the total number of events in which packets were dropped by the probe due to lack of resources during this sampling interval. This number is not necessarily the number of packets dropped; it is the number of times this condition has been detected.

Initial Value: 0

etherHistoryOctets (counter)

This read-only MIB object displays the total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).

Initial Value: 0

etherHistoryPkts (counter)

This read-only MIB object displays the number of packets (including bad packets) received during this sampling interval.

Initial Value: 0

etherHistoryBroadcastPkts (counter)

This read-only MIB object displays the number of good packets received during this sampling interval that were directed to the broadcast address.

Initial Value: 0

etherHistoryMulticastPkts (counter)

This read-only MIB object displays the number of good packets received during this sampling interval that were directed to a multicast address. This number does not include packets addressed to the broadcast address.

Initial Value: 0

etherHistoryCRCAlignErrors (counter)

This read-only MIB object displays 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 had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Initial Value: 0

etherHistoryUndersizePkts (counter)

This read-only MIB object displays the number of packets received during this sampling interval that were less than 64 octets long (excluding framing bits but including FCS octets) and were otherwise well-formed.

Initial Value: 0

etherHistoryOversizePkts (counter)

This read-only MIB object displays the number of packets received during this sampling interval that were longer than 1518 octets (excluding framing bits but including FCS octets) but were otherwise well-formed.

Initial Value: 0

etherHistoryFragments (counter)

This read-only MIB object displays the total number of packets received during this sampling interval that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Note It is normal for etherHistoryFragments to increment. This is because it counts both runts (normal occurrences because of collisions) and noise hits.

Initial Value: 0

etherHistoryJabbers (counter)

This read-only MIB object displays the number of packets received during this sampling interval that were longer than 1518 octets (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error).

Note This definition of jabber is different than the definition in IEEE 802.3, section 8.2.1.5 (10Base5) and section 10.3.1.4 (10Base2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowable range to detect jabber is between 20 ms and 150 ms.

Initial Value: 0

etherHistoryCollisions (counter)

This read-only MIB object displays the best estimate of the total number of collisions on this Ethernet segment during this sampling interval.

The value returned depends on the location of the RMON probe. IEEE 802.3, section 8.2.1.3 (10Base5) and section 10.3.1.3 (10Base2), states that a station must detect a collision in the receive mode if three or more stations are transmitting simultaneously. A repeater port must detect a collision when two or more stations are transmitting simultaneously. Thus a probe placed on a repeater port could record more collisions than a probe connected to a station on the same segment.

Probe location plays a much smaller role when considering 10BaseT. IEEE 802.3, section 14.2.1.4 (10BaseT), defines a collision as the simultaneous presence of signals on the DO and RD circuits (transmitting and receiving at the same time). A 10BaseT station can only detect collisions when it is transmitting. Thus probes placed on a station and a repeater should report the same number of collisions.

Note An RMON probe inside a repeater should ideally report collisions between the repeater and one or more other hosts (transmit collisions as defined by IEEE 802.3k), plus receiver collisions observed on any coaxial segments to which the repeater is connected.

Initial Value: 0

etherHistoryUtilization (integer [0 to 10000])

This read-only MIB object displays the best estimate, in hundredths of a percent, of the mean physical layer network utilization on this interface during this sampling interval.

Initial Value: 0

Alarm Group

From RFC 1757:

“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 time tick) can 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 of a deltaValue, a probe can 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. This does not require any special configuration of the threshold value. It is suggested that probes implement this more precise algorithm." The FastHub RMON agent implements the more precise algorithm.

alarmTable

alarmIndex (integer [1 to 65535])

This read-only MIB object is 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.

alarmInterval (integer)

This read-write MIB object specifies 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 taken in the case of deltaValue sampling—the interval should be set short enough that the sampled variable is very unlikely to increase or decrease by more than 2^31 to 1 during a single sampling interval.

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

Initial Value: 0

alarmVariable (object identifier)

This read-write MIB object specifies 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 time tick) can 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. Therefore, with 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 cannot be modified if the associated alarmStatus object is equal to valid (1).

Initial Value: 0 0

alarmSampleType (integer)

This read-write MIB object samples the selected variable and calculates the value to be compared against the thresholds. If the value of this object is absoluteValue (1), the value of the selected variable is compared 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 is subtracted from the current value, and the difference is compared with the thresholds.

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

Valid Values:	absoluteValue	(1)
	deltaValue	(2)
Initial Value:	deltaValue	(2)

alarmValue (integer)

This read-only MIB object displays the value of the statistic during the last sampling period. For example, if the sample type is deltaValue, this value will be the difference between the samples at the beginning and end of the period. If the sample type is absoluteValue, this value is the sampled value at the end of the period.

This is the value that is compared with the rising and falling thresholds.

The value during the current sampling period is not available until the period is completed and remains available until the next period completes.

Initial Value: 0

alarmStartupAlarm (integer)

This read-write MIB object specifies the alarm that can be sent when this entry is first set to valid. If the first sample after this entry becomes valid and is greater than or equal to the risingThreshold and alarmStartupAlarm is equal to risingAlarm (1) or risingOrFallingAlarm (3), then a single rising alarm is generated. If the first sample after this entry becomes valid and is less than or equal to the fallingThreshold and alarmStartupAlarm is equal to fallingAlarm (2) or risingOrFallingAlarm (3), then a single falling alarm is generated.

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

Valid Values:	risingAlarm	(1)
	fallingAlarm	(2)
	risingOrFallingAlarm	(3)
Initial Value:	risingOrFallingAlarm	(3)

alarmRisingThreshold (integer)

This read-write MIB object specifies 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 is generated. A single event is also generated if the first sample after this entry becomes valid and 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 is not generated until the sampled value falls below this threshold and reaches the alarmFallingThreshold.

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

Initial Value: 0

alarmFallingThreshold (integer)

This read-write MIB object specifies 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 is generated. A single event is also generated if the first sample after this entry becomes valid and 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 is not generated until the sampled value rises above this threshold and reaches the alarmRisingThreshold.

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

Initial Value: 0

alarmRisingEventIndex (integer [0 to 65535])

This read-write MIB object is 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 is generated because zero is not a valid event index.

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

Initial Value: Applicable eventEntry index

alarmFallingEventIndex (integer [0 to 65535])

This read-write MIB object is 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. If this value is zero, no associated event is generated because zero is not a valid event index.

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

Initial Value: Applicable eventEntry index

alarmOwner (OwnerString)

This read-write MIB object is the entity that configured this entry and is therefore using the resources assigned to it.

Initial Value: Assigned by creator of this entry

alarmStatus (EntryStatus)

This read-write MIB object is the status of this alarm entry.

Initial Value: Assigned by creator of this entry

Event Group

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 can also be associated with a function elsewhere in the MIB that is executed when the event is generated. For example, a channel might be turned on or off by the firing of an event.

Each eventEntry can optionally specify that a log entry be created on its behalf whenever the event occurs. Each entry can 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. Two traps are defined: risingAlarm and fallingAlarm. 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

eventIndex (integer [1 to 65535])

This read-only MIB object is an index that uniquely identifies an entry in the event table. Each entry defines one event to be generated when the appropriate conditions occur.

eventDescription (DisplayString [0 to 127])

This read-write MIB object is a comment describing this event entry.

Initial Value: String of length 0

eventType (integer)

This read-write MIB object specifies the type of notification that the probe makes about this event. In the case of log (2), an entry is made in the log table for each event. In the case of snmp-trap (3), an SNMP trap is sent to one or more management station.

Valid Values:	none	(1)	
	log	(2)	
	snmp-trap	(3)	send an SNMP trap
	log-and-trap	(4)	
Initial Value:	snmp-trap	(3)	

eventCommunity (octet string [0 to 127])

If an SNMP trap is to be sent, it is sent to the SNMP community specified by this octet string. This read-write MIB object should be set to a string of length zero if it is intended that the mechanism be used to specify the destination of the trap.

Initial Value: String of length 0

eventLastTimeSent (time tick)

This read-only MIB object displays the value of sysUpTime when this event entry last generated an event. If this entry has not generated any events, this value is zero.

Initial Value: 0

eventOwner (OwnerString)

This read-write MIB object is 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 workstations should retrieve those log entries, as they can have significance to all management workstations connected to this device.

Initial Value: Assigned by creator of this entry

eventStatus (EntryStatus)

This read-write MIB object is 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.

Initial Value: Assigned by creator of this entry

logTable

logEventIndex (integer [1 to 65535])

This read-only MIB object is 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.

Initial Value: Applicable eventEntry index

logIndex (integer [1 to 2147483647])

This read-only MIB object is an index that uniquely identifies an entry in the log table among those generated by the same eventEntries. These assigned indexes begin 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 can 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 can occur.

Initial Value: variable

logTime (time tick)

This read-only MIB object is the value of sysUpTime when this log entry was created.

Initial Value: Applicable value of sysUpTime

logDescription (DisplayString [0 to 255])

This read-only MIB object is an implementation-dependent description of the event that activated this log entry.

Remote Network Monitoring Traps

risingAlarm

This SNMP trap is generated when an alarm entry crosses its rising threshold and generates an event that is configured for sending SNMP traps.

Variables: alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold

fallingAlarm

This SNMP trap is generated when an alarm entry crosses its falling threshold and generates an event that is configured for sending SNMP traps.

Variables: alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold