Understanding Network Device Traps

Thursday, October 08, 1998

V1.4

Digital Equipment Corporation

Table of Contents

Tables:

Introduction

Effective network management requires detailed knowledge of the current state of each device in the network. To ensure the timeliness of this information, it must be supplied by the devices themselves at their own initiative, rather than in response to manager requests. Specifically, each device must be responsible for informing a management station whenever it detects a significant change in its operations or environment.

The SNMP standard, which is implemented in DIGITAL networks and devices, provides a mechanism whereby devices can asynchronously notify managers of changes in their operations or environment. SNMP (Simple Network Management Protocol) is a set of network management standards for IP-based internetworks. It includes a protocol, a database-structure specification, and a set of management data objects. SNMP implementations typically consist of a management application, running on one or more Network Management Systems (NMSs), and agent applications, usually executing in firmware on various network devices.

The specific SNMP mechanism that allows a device to asynchronously notify a hub of changes is known as a *trap*.

What Is a Trap?

Trap, along with get, getnext, and set, is one of the four SNMP-supported operations. It designates the action whereby an SNMP agent sends an unsolicited message to one or more SNMP managers. (Each device can have up to eight addresses defined for trap destinations.) Agents use traps to notify their managers of changes or faults that have been detected in the agent.

As SNMP protocol data units (PDUs), traps exhibit a distinctive format. The fields in the trap PDU format are as follows:

PDU type
: An identifier that indicates this is a trap PDU (as opposed to a get or set request PDU).

enterprise
: The network management subsystem that generated the trap. The type of object/agent that generated the trap. Very often, it is the sysObjectID of the agent/object.

agent-addr
: The IP address of the agent that generated the trap.

generic-trap
: An identifier indicating that this is one of the standard, predefined trap types.

specific-trap
: An identifier indicating that this is a device-specific trap, such as an RMON-generated trap or a DIGITAL enterprise-specific trap. This is

useful only when a generic trap is set or equal to enterpriseSpecific (6).

time-stamp
The time between the most recent initialization of the agent and the generation of the trap.

variable-bindings
Additional, implementation-specific information about the trap.

Receiving and Viewing Traps

To receive and view traps sent by SNMP agents, use the capabilities provided by your Network Management System.

NOTE
DIGITAL also provides a tool in the clearVISN kit for Windows NT that accepts traps and puts them into the Windows NT event viewer. The directions are available on the clearVISN CD.

Each trap that you receive can be distinguished by its object identifier (OID). An OID indicates the Management Information Base (MIB) that contains its definition. The MIB, in turn, indicates the object's location within the SNMP management tree.

Note that some MIBs are public and industry standard MIBs, while other MIBs are private and enterprise specific. Examples of public and industry-standard MIBs are the TCP/IP-Based MIB II (RFC 1213) and the Remote Network Monitoring (RMON) MIB (RFC 1757). Examples of private and enterprise-specific MIBs are DIGITAL's proprietary, device-specific MIBs, such as the DECHUB900-HUBMGR-MIB-V3-0 MIB.

Interpreting a trap consists of using its enterprise OID to look up its definition in the associated MIB. For example, when you receive a trap using your Network Management System, the NMS will attempt to perform the definition lookup for you automatically. To successfully access the definitions, however, you must make sure that you have pre-loaded all the relevant MIBs into your NMS.

NOTE
Some Network Management Systems do not interpret the trap from the MIB. In that case, you must define the traps in the NMS yourself.

SNMP Trap Types

DIGITAL supports the following four types of SNMP traps:

- **Generic traps** – These are traps defined in the public and industry-standard Simple Network Management Protocol (SNMP RFC 1157).
- **RMON-generated traps** - These are traps that are produced with the Remote Network Monitoring (RMON) alarm and event mechanism, as specified in the public and industry-standard Remote Network Monitoring Management Information Base (RFC 1757).
- **DIGITAL RMON-like traps** - These are DIGITAL-defined traps that are implemented as extensions to the basic RMON alarm-event mechanism. These exist only on 600 modules. They are defined in the DIGITAL private and enterprise-specific DECHUB900-HUBMGR-MIB-V3-0 MIB.

- **DIGITAL enterprise-specific traps** - These are DIGITAL-defined traps that are specific to DIGITAL device types. They are defined in DIGITAL proprietary MIBs, such as the DECHUB900-HUBMGR-MIB-V3-0 MIB.

Each trap type is described in detail later in this paper.

Configuring for SNMP Traps

To receive and interpret SNMP traps, you must ensure that all required MIBs are loaded into your Network Management System.

As a rule, most Network Management Systems come with the following public and industry-standard MIBs pre-loaded:

- Simple Network Monitoring Protocol (SNMP RFC 1157)
- TCP/IP Based Internets MIB II (RFC 1213)

If your Network Management System does not provide these public MIBs, you must acquire and load them yourself.

NOTE

Some Network Management Systems do not interpret the trap from the MIB. In that case, you must define the traps in the NMS yourself.

The following standard MIBs must be loaded, but are not usually loaded automatically by an NMS:

- Remote Network Monitoring (RMON) MIB (RFC 1757)
- IETF Repeater MIB (RFC 1516)
- IETF Bridge MIB (RFC 1493)

In addition, for any DIGITAL device types that are managed by your system, you should load the appropriate DEC private and enterprise-specific MIB. The available DEC MIBs are:

- DECHUB900-CHASSIS-MIB-V3-0
- DECHUB900-COMMON-MIB-V3-0
- DECHUB900-HUBMGR-MIB-V3-0
- DECHUB900-ERPTR-MIB-V3-0
- DEC-ELAN-MIB

The following MIBs can be found at DIGITAL's web site or on the clearVISN V2.1 (or later) CD:

http://www.networks.digital.com/dr/hubs/mibs

- TCP/IP Based Internets MIB II (RFC 1213)
- RMON MIB (RFC 1757)
- IETF Repeater MIB (RFC 1516)
- IETF Bridge MIB (RFC 1493)

- DIGITAL private and enterprise-specific MIBs

Note that the only standard MIB you cannot find at this site is Simple Network Monitoring Protocol (SNMP RFC 1157).

Enabling and Disabling Traps

To enable traps for a given device, the address for at least one Network Management System must be entered in the device's trap address table. Each device can have up to eight trap addresses assigned, with no duplicates allowed. When a trap occurs, the SNMP agent sends the trap PDU to each address listed in the device's trap table.

For Generic traps and DIGITAL enterprise-specific traps, all that is necessary is to define one or more trap addresses. For RMON-generated traps and DIGITAL RMON-like traps, an appropriate Alarm table and associated Event table entry must be installed on the agent. The alarm/event table mechanism by itself will generate a trap if an alarm condition occurs. However, if no trap addresses have been defined, the trap can not be sent.

NOTE

In the case of RMON-generated traps and DIGITAL RMON-like traps, you should set the trap addresses before setting the Alarm and Event table entries. The chassis and repeater network devices have factory default alarm and event table entries pre-configured. In that case, you cannot set the trap address first.

To disable Generic traps and DIGITAL enterprise-specific traps, simply remove all addresses from the agent's trap address table. To disable RMON-generated traps and DIGITAL RMON-like traps, remove either all trap addresses or trap-generating entries from the Alarm table.

You can add or remove trap addresses from the command line on the device console or by using the capabilities provided by your Network Management System or clearVISN.

SNMP Generic Traps

Generic traps are standard traps that are common to all SNMP implementations. They are defined in the Simple Network Monitoring Protocol (SNMP RFC 1157). In the case of a generic trap, the agent indicates which generic trap it is by supplying the appropriate integer code in the *generic-trap* field in the trap PDU. The seven possible generic traps (with their associated codes) are as follows:

coldStart [0]
: The agent is reinitializing itself in such a way that the objects in its view may be altered. This trap typically indicates an unexpected restart resulting from a crash or major fault.

warmStart [1]
: The agent is reinitializing itself, but without any alteration to the objects in its view. This trap typically indicates a routine restart.

linkDown [2]
: A failure has occurred in one of the agent's communications links. The *variable-bindings* field in the trap PDU contains a pointer to the affected link.

linkUp [3]
: One of the agent's communications links has come up. The *variable-bindings* field in the trap PDU contains a pointer to the affected link.

authenticationFailure [4]
: The agent has received a message that has failed authentication. This trap typically indicates an invalid access attempt.

egpNeighborLoss [5]
: An EGP (Exterior Gateway Protocol) peer for the agent has transitioned to the down state. This is a router-specific trap.

enterpriseSpecific [6]
: The agent has detected an enterprise-specific event. Examples of such an event might include a module status change, a port status change, or a change in configuration limits. The *specific-trap* field in the trap PDU indicates the type of trap involved.

Since generic traps are self-explanatory, there is usually no need for further interpretation.

NOTE

You may need to manually define traps in the NMS, because some vendors do not put generic traps in the MIB or not all Network Management Systems interpret generic traps.

Note that if a code of 6 appears in the generic-trap field, it indicates that the trap is not a generic trap, but an enterprise-specific trap. RMON traps, RMON-like traps, and DIGITAL enterprise-specific traps are all examples of enterprise-specific traps. From the point of view of SNMP, an enterprise-specific trap is simply a trap that is defined in a MIB other than the standard Simple Network Monitoring Protocol (SNMP RFC 1157).

Table 1 lists the traps that are supported by each DIGITAL device module.

Table 1: Traps Supported by DIGITAL Device Modules

*** - Indicates available in a future release.**

Device	cold Start(0)	warm Start(1)	link Down(2)	link Up(3)	authentication Failure(4)	egpNeighbor Loss(5)	enterprise Specific(6)	RMON(6)	RMON-Like(6)
DECagent 90	X	X			X		X		
DEChub 900	X		X	X	X			X	X
MultiSwitch 600 Stack Director	X		X	X	X			X	X
MultiSwitch WorkGroup	X		X	X	X			X	X
MultiSwitch 300	X	X	X	X	X	X	X	X	
DECrepeater 900 TM/GM/CP PORTswitch 900 TP/TP12/CP/FP	X		X	X	X			X	
DECrepeater 90TPlus	X		X	X	X				
MultiSwitch Hub 624T/612TX	X			X					X
GIGAswitch/FDDI		X			X		X	*X	
GIGAswitch/ATM			X	X			*X	*X	
DECswitch 900EE/EF/EF-MP/FO/ET PEswitch 900TX DECbridge 900MX	X		X	X	X			X	
VNswitch 900 EA/EE/EF-MX/EF-MM/EX/FA/LL/XA/XX/ *FF/*FX/ *GV/*GC/*CC	X	X	X	X	X	X	X	*X (refer to VN type)	
MultiSwitch 612EX/624EX	X			X					X
ATMswitch 900T/F			X	X				*X	
DECconcentrator 900MX/900TH/900FH	X		X	X	X		X		
DECserver 900TM, 900GM, 900MC DECserver 700-08, 700-16 DECserver 90TL, 90M	X		X	X	X		X		
RouteAbout Central EW/EI	X	X	X	X	X		X		
RouteAbout Access EW/ES/EI/ISDN/TW	X	X	X	X	X	X	X		

Table 2: Enterprise SpecificTraps(6) Supported by DIGITAL Device Modules

*** - Indicates available in a future release.**

Device	enterprise Specific(6)	Trap Variable Arguments	MIB Location
DECagent 90	consolePasswordFailure(1)	*none*	DECHUB90-MIB
	nonVolatileRamError(2)	*none*	
	configurationExceeded(3)	*none*	
	populationChange(4)	*none*	
	moduleStatusChange(5)	*none*	
	rptrPortStatusChange(6)	*none*	
	srvrPortStatusChange(7)	*none*	
	brdgPortStatusChange(8)	*none*	
DEChub 900	risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
MultiSwitch 600 Stack Director	fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
MultiSwitch WorkGroup	hubRisingAlarm(1)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmRisingThreshold	DECHUB900-HUBMGR-MIB-V3-0
	hubFallingAlarm(2)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmFallingThreshold	
MultiSwitch 300	risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
	newRoot(1)	*none*	BRIDGE-MIB (RFC1493)
	topologyChange(2)	*none*	
**DECrepeater 900 TM/GM/CP	risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
**PORTswitch 900 TP/TP12/CP/FP	fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
MultiSwitch Hub 624T/612TX	hubRisingAlarm(1)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmRisingThreshold	DECHUB900-HUBMGR-MIB-V3-0
	hubFallingAlarm(2)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmFallingThreshold	
GIGAswitch/FDDI	*risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	*fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
	newRoot(1)	*none*	BRIDGE-MIB (RFC1493)
	topologyChange(2)	*none*	
	lineCardFailureTrap(3)	*none*	DEC-ELAN-MIB
GIGAswitch/ATM	*risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	*fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
	*decAtmKeyTurned(1)	decAtmKeyswitchPosition	DEC-ATM-CHASSIS-MIB
	*decAtmKeyDown(2)	decAtmKeyswitchPosition	
	*decAtmKeyUp(3)	decAtmKeyswitchPosition	
	*decAtmCardSwap(4)	decAtmSlotIndex, decAtmCardStatus	
	*decAtmCardDown(5)	decAtmSlotIndex, decAtmCardStatus	
	*decAtmCardUp(6)	decAtmSlotIndex, decAtmCardStatus	
	*decAtmPscSwap(7)	decAtmPscStatus, decAtmPscFwImageStatus,	

****** - indicates NO support for repeater TRAP-TYPEs: (rptrHealth, rptrGroupChange, rptrResetEvent) in RFC 1516.

	*decAtmPscDown(8)	decAtmPscBackplaneStatus decAtmPscStatus, decAtmPscFwImageStatus,	
	*decAtmPscUp(9)	decAtmPscBackplaneStatus decAtmPscStatus, decAtmPscFwImageStatus,	
	*decAtmPowerSupplySwap(10)	decAtmPscBackplaneStatus	
	*decAtmPowerSupplyDown(11)	decAtmPowerIndex, decAtmPowerStatus	
	*decAtmPowerSupplyUp(12)	decAtmPowerIndex, decAtmPowerStatus	
	*decAtmBatterySwap(13)	decAtmPowerIndex, decAtmPowerStatus	
	*decAtmBatteryDown(14)	decAtmBatteryStatus, decAtmBatteryCharge	
	*decAtmBatteryUp(15)	decAtmBatteryStatus, decAtmBatteryCharge	
	*decAtmTempYellowAlert(16)	decAtmBatteryStatus, decAtmBatteryCharge decAtmCabinetTemperature,	
	*decAtmTempRedAlert(17)	decAtmTemperatureWarning decAtmCabinetTemperature,	
	*decAtmTempSensorDown(18)	decAtmTemperatureWarning	
	*decAtmTempSensorUp(19)	decAtmCabinetTemperature	
	*decAtmFanSwap(20)	decAtmCabinetTemperature	
	*decAtmFanDown(21)	decAtmFanIndex, decAtmFanStatus	
	*decAtmFanUp(22)	decAtmFanIndex, decAtmFanStatus	
	*uniMisconfigured(1)	decAtmFanIndex, decAtmFanStatus ifIndex, ifDescr	DEC-ATM-MIB
***DECswitch 900EE/EF/EF-MP/FO/ET ***PEswitch 900TX ***DECbridge 900MX	risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
***VNswitch 900 EA/EE/EF-MX/EF-MM/EX/FA/LL/XA/XX /*FF/*FX/*GV/*GC/*CC	*risingAlarm(1) (refer to type)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	*fallingAlarm(2) (refer to type)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
	proElsTrapV1(1)	proElsTrapSeqs, proElsTrapSubSystem, proElsTrapEvent	DEC-COMET-MIB-V1-2
	proElsTrapV2(2)	proElsSubSysEventMsg	
MultiSwitch 612EX/624EX	hubRisingAlarm(1)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmRisingThreshold	DECHUB900-HUBMGR-MIB-V3-0
	hubFallingAlarm(2)	hubAlarmSlotNumber, hubAlarmIndex, hubAlarmVariable, hubAlarmSampleType, hubAlarmValue, hubAlarmFallingThreshold	
ATMswitch 900T/F	*risingAlarm(1)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmRisingThreshold	RMON-MIB (RFC1757)
	*fallingAlarm(2)	alarmIndex, alarmVariable, alarmSampleType, alarmValue, alarmFallingThreshold	
DECconcentrator 900MX/900TH/900FH	****fddimibPORTConnectState	1.3.6.1.2.1.10.15.73.5.2.1.20.*smtIndex.portvalue*	FDDI-MIB (RFC1512)
DECserver 900TM, 900GM, 900MC DECserver 700-08, 700-16 DECserver 90TL, 90M	acctThresholdExceeded(1)	sysUpTime, acctThreshold	DECserver-Accounting-MIB (V1.0)
RouteAbout Central EW/EI	proElsTrapV1(1)	proElsTrapSeqs, proElsTrapSubSystem, proElsTrapEvent	DEC-COMET-MIB-V1-2
RouteAbout Access EW/ES/EI/ISDN/TW	proElsTrapV2(2)	proElsSubSysEventMsg	

******* - indicates NO support for bridge TRAP-TYPEs: (newRoot and TopologyChange) in RFC 1493.

******** - indicates a special case trap that is NOT defined as a TRAP-TYPE definition in the MIB.
(In the clearVISN Integration definitions, this trap is named fddimibPORTConnectStateChanged.)

RMON-Generated Traps

The RMON (Remote Network Monitoring) framework represents an addition to the

basic set of SNMP standards. RMON has its own specification and is defined in the RMON MIB (RFC 1757).

RMON extends the monitoring capabilities of SNMP (which is focused on local devices) to the network as a whole. RMON supports the implementation of special agents, typically called *monitors* or *probes*. In the DIGITAL implementation of RMON, agent probes can be the devices themselves or they can execute in firmware on network device modules. By sampling various variables in the device's firmware, they can provide up-to-date information on the state of the device and the network activity associated with that device.

The RMON MIB supports a full set of sampling, data capture, and analysis functions.

How RMON Alarms and Events Work

The RMON alarm-event mechanism is similar to a conditional statement in a programming language. The alarm part of the mechanism defines a condition for a variable in the device's firmware. The associated event specifies the action to take when the condition is met. For example, each alarm defines a threshold for a variable. The agent probe samples the alarm variable at a prescribed interval. When the agent detects that the threshold has been crossed (in either a rising or falling direction), it may issue a trap (in standard SNMP PDU format) to notify the manager(s) of the alarm condition, or just log the event, or both.

Unlike the seven standard SNMP generic traps, there is no limit on the number of RMON-generated traps. The number of traps that can be generated is determined solely by the number of alarms and associated events that have been defined for each device type in the network. Most device types have a limit to the number of alarm and event entries that can be configured in a device.

The RMON alarm-event mechanism is implemented in device firmware as two separate tables: an alarm table and an event table. For each device type, the alarm and event tables contain a unique set of values. For example, the alarm-event values for repeaters are different from the alarm-event values for switches.

Alarm Table

Each entry (row) in the alarm table contains the following objects:

alarmIndex
: An integer that uniquely identifies the entry's position in the table.

alarmInterval
: The interval, in seconds, between samples.

alarmVariable
: The object ID of the variable to be sampled. The variable object must be a MIB variable known to the device. Basically, only integer object types can be sampled.

alarmSampleType
: The method for calculating the value to be compared to the threshold. Two methods are supported. absoluteValue compares the variable's current value directly with the threshold(s). deltaValue subtracts the last

sample value from the current sample value and compares the difference with the threshold(s).

alarmValue
: The current value of the variable, as detected by the most recent sample.

alarmStartupAlarm
: An indicator that specifies how the first alarm will be generated after startup: if the first sample is greater than the RisingThreshold (1); if the first sample is less than the FallingThreshold (2); or if the first sample is either greater than the RisingThreshold or less than the FallingThreshold (3).

alarmRisingThreshold
: A value that specifies the rising threshold for the target variable. With the absoluteValue method, an alarm occurs when a sampled value is greater than or equal to this value.

alarmFallingThreshold
: A value that specifies the falling threshold for the target variable. With the absoluteValue method, an alarm occurs when a sampled value is less than or equal to this value.

alarmRisingEventIndex
: The index of the entry in the event table to use when the RisingThreshold is crossed.

alarmFallingEventIndex
: The index of the entry in the event table to use when the FallingThreshold is crossed.

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

AlarmStatus
: The status of this alarm entry.

DIGITAL provides factory default alarm (and matching event) table entries for chassis and repeater network device types.

Each alarm entry defines a combination of variable, sampling interval, and threshold parameters. An alarm may be signaled by crossing either the RisingThreshold or the FallingThreshold. The RisingThreshold is crossed if the value detected by the most recent sample is greater than or equal to the specified threshold value. The FallingThreshold is crossed if the value detected by the most recent sample is less than or equal to the specified threshold value.

alarmRisingEventIndex and alarmFallingEventIndex point to an associated event depending on whether the alarm resulted from crossing the RisingThreshold or the FallingThreshold. A given alarm can specify an event for either or both cases. Event entries are defined in the event table, which is described next.

Event Table

Trap events map to conditions defined in the alarm table.

Each entry (row) in the event table contains the following objects:

eventIndex
: An integer that uniquely identifies the entry's position in the table. An alarm entry indicates which event to use by referencing this value for the alarmRisingEventIndex or alarmFallingEventIndex object.

eventDescription
: A text description of the event.

eventType
: An integer indicator that identifies the type of event. Posssible values are: 1 (none), 2 (log), 3 (SNMP trap), and 4 (log and trap). All default event entries in the DIGITAL RMON implementation are log and trap (type 4) entries.

eventCommunity
: For trap entries only, the community string of the NMS is included so the trap can be accepted by the appropriate Network Management Systems.

eventLastTimeSent
: The time that the entry last generated an event (a trap in this case).

eventOwner
: The entity responsible for configuring this event entry.

eventStatus
: The status of this event entry.

Default RMON Alarms

Table 3 lists the default RMON Alarm Device Defaults (DD) and Added Value (AV) definitions. The first column indicates the device type to which the associated entries apply. MIB varialbes are preceded with 1.3.6.1.4.1.36.2.18.11…, except where indicated by an asterisk (*).

Table 3: RMON Alarm Device Default and DIGITAL Added Value Definitions

g = Repeater Group Instance **n** = Port Instance

	AIndex	AInterval	AVariable	Asample Type	AValue	AStart upA	ARise Thresh	AFall Thresh	ARise Event	AFall Event	AOwner	AStatus
Chassis (DD)	1	1	chasNumSlotsOccupied .1.1.1.1.2.0	2- Delta Value	0	3=rising OrFallingAlarm	0	-1	0	1	monitor	1
(DD)	2	1	chasNumSlotsOccupied .1.1.1.1.2.0	2- Delta Value	0	3=rising OrFallingAlarm	1	0	2	0	monitor	1
(DD)	3	5	chasEnvironChanges .1.1.1.10.1.0	2- Delta Value	0	1=rising Alarm	1	0	3	0	monitor	1
(DD)	4	2	chasPowerConfigNumSupplies .1.1.1.7.2.0	2- Delta Value	0	3=rising OrFallingAlarm	1	0	4	0	monitor	1
(DD)	5	2	chasPowerConfigNumSupplies .1.1.1.7.2.0	2- Delta Value	0	3=rising OrFallingAlarm	0	-1	0	5	monitor	1
(DD)	6	2	chasPowerConfigRedundancyState .1.1.1.7.4.0	2- Delta Value Value	0	3=rising OrFallingAlarm	1	-1	6	7	monitor	1
(DD)	7	2	chasConnChanges .1.1.1.5.2.0	2- Delta Value	0	1=rising Alarm	1	0	8	0	monitor	1
(DD)	8	30	pcomEsysNVRAMavailableOctets .2.7.6.0	1- Absolute Value	19561	1=rising Alarm	0	0	9	0	monitor	1
(DD)	9	30	pcomEsysNVRAMfailedFlag .2.7.4.0	1- Absolute Value	2	1=rising Alarm	0	1	0	10	monitor	1
Rptr Group (AV)	12	1	rptrMonitorGroupTotalFrames *1.3.6.1.2.1.22.2.2.1.1.2.***g**	2- Delta Value	0	1=rising Alarm	1	0	14	0	monitor cv	1
(AV)	13	1	rptrMonitorGroupTotalOctets *1.3.6.1.2.1.22.2.2.1.1.3.**g**	2- Delta Value	0	1=rising Alarm	1	0	15	0	monitor cv	1
(AV)	14	1	rptrMonitorGroupTotalErrors *1.3.6.1.2.1.22.2.2.1.1.4.**g**	2- Delta Value	0	1=rising Alarm	1	0	16	0	monitor cv	1
Rptr Ports (AV)	9	1	rptrMonitorPortReadableFrames *1.3.6.1.2.1.22.2.3.1.1.3.***g.n**	2- Delta Value	0	1=rising Alarm	1	0	11	0	monitor cv	1
(AV)	10	1	rptrMonitorPortReadable Octets *1.3.6.1.2.1.22.2.3.1.1.4.***g.n**	2- Delta Value	0	1=rising Alarm	1	0	12	0	monitor cv	1
(AV)	11	1	rptrMonitorPortTotalErrors *1.3.6.1.2.1.22.2.3.1.1.15.***g.n**	2- Delta Value	0	1=rising Alarm	1	0	13	0	monitor cv	1
(AV)	15	1	rptrMonitorPortCollisions *1.3.6.1.2.1.22.2.3.1.1.10.***g.n**	2- Delta Value	0	1=rising Alarm	1	0	17	0	monitor cv	1

Table 2: RMON Alarm Device Default and DIGITAL Added Value Definitions continued

g = Repeater Group Instance **n** = Port Instance

	AIndex	AInterval	AVariable	ASample Type	AValue	AStart upA	ARise Thresh	AFall Thresh	ARise Event	AFall Event	AOwner	AStatus
Rptr (DD)	1	2	rptrTotalPartitionedPorts *1.3.6.1.2.1.22.1.1.6.0	2- Delta Value	*0*	1=rising Alarm	1	1	1	2	monitor	1
(DD)	2	30	pcomEsysNVRAMavailableOctets .2.7.6.0	1- Absolute Value	*0*	1=rising Alarm	0	0	3	0	monitor	1
(DD)	3	2	erptrHealthTextChanges .5.1.1.1.1.4.0	2- Delta Value	*0*	1=rising Alarm	1	0	4	0	monitor	1
(DD)	4	2	erptrTotalPortsEvents .5.1.1.1.1.5.0	2- Delta Value	*0*	1=rising Alarm	1	0	5	0	monitor	1
(DD)	5	2	erptrTotalRptrErrors .5.1.1.1.1.6.0	2- Delta Value	*0*	1=rising Alarm	1	0	6	0	monitor	1
(DD)	6	10	erptrDprTotalStateChanges .5.1.1.3.1.1.0	2- Delta Value	*0*	1=rising Alarm	1	0	7	0	monitor	1
(DD)	7	30	erptrSecurityRptrSecurityViolations .5.1.1.4.1.1.0	2- Delta Value	*0*	1=rising Alarm	1	0	8	0	monitor	1
(DD)	8	2	erptrMauTotalMediaUnavailable .5.1.1.5.1.1.0	2- Delta Value	*0*	3=rising OrFalling Alarm	1	1	9	10	monitor	1
Switch (AV)	1	1	dot1dStpTopChanges *1.3.6.1.2.1.17.2.4.0	2- Delta Value	0	3=rising OrFalling Alarm	1	0	1	0	monitor cv	1
(AV)	2	1	dot1dStpPortState *1.3.6.1.2.1.17.2.15.1.3***.n***	2- Delta Value	0	3=rising OrFalling Alarm	1	0	2	0	monitor cv	1
(AV)	3	1	ebrIfBadHelloLimitExceeded *1.3.6.1.4.1.36.2.18.1.4.5.1.1.20***.n***	2- Delta Value	0	3=rising OrFalling Alarm	1	0	3	0	monitor cv	1
(AV)	4	30	ifOutDiscards *1.3.6.1.2.1.2.2.1.19***.n***	2- Delta Value	0	3=rising OrFalling Alarm	5	0	4	0	monitor cv	1
(AV)	5	1	dot1dTpLearnedEntryDiscards *1.3.6.1.2.1.17.4.1.0	2- Delta Value	0	3=rising OrFalling Alarm	1	0	5	0	monitor cv	1
(AV)	6	1	esysUnsolicitedResets *1.3.6.1.4.1.36.2.18.1.2.3.3.0	2- Delta Value	0	3=rising OrFalling Alarm	1	0	6	0	monitor cv	1

Default RMON Events

Table 4 lists the RMON Event Device Defaults (DD) and DIGITAL Added Value (AV) definitions.

Table 4: RMON Event Device Default and DIGITAL Added Value Definitons

	EIndex	EDescription	EType	ECommunity	ELastTimeSent	EOwner	EStatus
Chassis (DD)	1	A network module was removed.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	2	A network module was inserted.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	3	An environmental change occurred.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	4	A power supply was inserted.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	5	A power supply was removed.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	6	The DEChub 900 no longer has N-Plus-1 power redundancy.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	7	The DEChub 900 now has N-plus-1 power redundancy.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	8	A backplane connection change has occurred.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	9	Nonvolatile RAM cannot accept any additional parameters.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	10	The nonvolatile memory has failed.	4-log-and -trap	public	*0days:0hours*	monitor	1
Rptr Group (AV)	14	Total number of repeater group readable frames.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	15	Total number of repeater group readable octets.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	16	Total number of repeater group errors.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
Rptr Port (AV)	11	Total number of repeater port readable frames.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	12	Total number of repeater port readable octets.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	13	Total number of repeater port errors.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	17	Total number of repeater port collisions.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
Rptr (DD)	1	One or more ports have autopartitioned.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	2	One or more autopartitioned ports are now operational.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	3	There is no more space for non-volatile parameters.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	4	The repeater's 'rptrHealthText' has changed.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	5	The total number of times a port has become not-operational, autopartitioned, or unavailable.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	6	The total number of errors for this repeater.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	7	A Dual Port Redundancy state change occurred.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	8	A repeater security violation occurred.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	9	One or more media has become unavailable.	4-log-and -trap	public	*0days:0hours*	monitor	1
(DD)	10	One or more formerly unavailable media are now available.	4-log-and -trap	public	*0days:0hours*	monitor	1

Table 3: RMON Event Device Default and DIGITAL Added Value Definitions continued

	EIndex	EDescription	EType	ECommunity	ELastTimeSent	EOwner	EStatus
Switch (AV)	1	The topology of the Spanning Tree has changed.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	2	The Spanning Tree state of a port on this switch has changed.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	3	Rogue bridge on extended LAN. A bridge seen on the interface corresponding to this counter is propagating bad spanning tree information.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	4	Frame dropped due to excessive delay caused by congestion on LAN to which this port is connected.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	5	An address was not learned by this switch because the number of addresses seen on the extended LAN exceeded the capacity of the forwarding database.	4-log-and -trap	public	*0days:0hours*	monitor cv	1
(AV)	6	An unexpected failure caused the device to crash. Error log contains information for use by DIGITAL support to analyze failure.	4-log-and -trap	public	*0days:0hours*	monitor cv	1

Interpreting RMON-Generated Traps

When you receive an RMON-generated trap, the trap information will indicate that it is either an RMON Rising Alarm or an RMON Falling Alarm. In addition, the trap will indicate the target variable for which the alarm is defined. For example, a sample RMON-generated trap might appear in Netview as follows:

```
Description: RMON Rising Alarm:
erptrRptrInfo.erptrTotalRptrErrors.0exceeded
        threshold 1 value = 4165484 ( Sample type = 2 alarm index 5)

Information:
    Node: offsb2_tp5.domainname
    Enterprise: 1.3.6.1.2.1.16 (rmon)
    Trap: RMON_ALARM - #1
    Logged Time: Mon Jan 27 06:36:01 1997
    Severity: Critical
    Category: Threshold Events
    Source: Agent
```

In this case, the target variable is `erptrRptrInfo.erptrTotalRptrErrors.` The actual contents of the variable at the time of the sample are shown by `value =`. `Sample type =` indicates whether the absolute (1) or delta (2) method was used to calculate the value to be compared with the threshold. The name of the variable indicates that the associated alarm is a Repeater alarm and is defined in the Repeaters alarm table. To determine the meaning of the trap:

1. Use your NMS's MIB Browser to access the Repeaters alarm table (see default tables, above). Look up the entry (AIndex 5 for alarm index 5) that references the target variable (`erptrRptrInfo.erptrTotalRptrErrors`). In this case, the alarm entry is defined as follows:

	AIndex	AInterval	AVariable	ASample Type	AValue	AStart upA	ARise Thresh	AFall Thresh	ARise Event	AFall Event	AOwner	AStatus
(DD)	5	2	erptrTotalRptr Errors .5.1.1.1.1.6.0	2- Delta Value	*0*	1=rising Alarm	1	0	6	0	monitor	1

2. Since this is a rising alarm, find the index number for the corresponding rising event entry. In this case, the ARiseEvent value is 6.

3. Use your NMS's MIB Browser to access the Repeaters event table (see default tables, above). Look up the entry whose index number was referenced in the alarm entry. In this case, the event entry (index 6) is defined as follows:

	EIndex	EDescription	EType	ECommunity	ELastTimeSent	EOwner	EStatus
(DD)	6	The total number of errors for this repeater.	4-log-and -trap	public	*0days:0hours*	monitor	1

The EDescription object provides some information about the trap. In this case, the information ("The total number of errors for this repeater.") is not too meaningful. To get more information, you can consult the definition of the target variable in the associated DIGITAL Repeaters MIB. The definition for `erptrTotalRptrErrors` in the DECHUB900-ERPTR-MIB-V3-0 MIB appears as follows:

```
erptrTotalRptrErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
          "The total number of errors which have occurred on all
           the groups in a repeater. This object is a summation
           of the values of the rptMonitorGroupTotalErrors as
           defined in RFC 1516 for all the groups in a
           repeater."
        REFERENCE
            "Reference RFC 1516 repeater MIB"
        ::= { erptrRptrInfo 6 }
```

The referenced `rptrMonitorGroupTotalErrors` object is defined in the standard IETF Repeaters MIB (RFC 1516) as follows:

```
rptrMonitorGroupTotalErrors OBJECT-TYPE
        SYNTAX    Counter
        ACCESS    read-only
        STATUS    mandatory
        DESCRIPTION
                "The total number of errors which have occurred on
                all the ports in this group.  This counter is
                the summation of the values of the
                rptrMonitorPortTotalErrors counters for all the
                ports in the group."
        ::= { rptrMonitorGroupEntry 4 }
```

Finally, the definition for the referenced `rptrMonitorPortTotalErrors` object appears in the standard IETF Repeater MIB (RFC 1516) as follows:

```
rptrMonitorPortTotalErrors OBJECT-TYPE
        SYNTAX    Counter
        ACCESS    read-only
        STATUS    mandatory
        DESCRIPTION
                "The total number of errors which have occurred on
                this port.  This counter is the summation of the
                values of other error counters (for the same
                port), namely:

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

                This counter is redundant in the sense that it is
                the summation of information already available
                through other objects.  However, it is included
                specifically because the regular retrieval of this
                object as a means of tracking the health of a port
                provides a considerable optimization of network
                management traffic over the otherwise necessary
                retrieval of the summed counters."
        ::= { rptrMonitorPortEntry 15 }
```

The net result of this investigation indicates that you should examine all the error counters on all the repeater ports to find out which port(s) and which error(s) are causing the traps.

NOTE

clearVISN 2.1 contains an RMON Trap application currently available for NetView only that automatically performs the preceding manual steps for you. Refer to the README on the CD \nmsintegration\netview\cvnvreadme.txt.

DIGITAL RMON-Like Traps

DIGITAL has implemented RMON-like traps for the DIGITAL MultiSwitch 900 (previously called the DEChub 900 MultiSwitch) and MultiSwitch 600 chassis devices. From an SNMP point of view, these devices consist of two parts: the overall device itself and the actual modules (slots) that are installed within the device. These devices present a special case in that the manager must be able to distinguish between a trap that is associated with the device itself and a trap that is associated with a module within the device.

Standard RMON-generated traps are used for the DIGITAL MultiSwitch 900 and the MultiSwitch 600 chassis devices themselves. The RMON-like traps are used exclusively for MultiSwitch 600 modules within these chassis devices. The RMON-like traps are supported only for 600-class modules.

To associate a trap with a specific 600 module in a DIGITAL MultiSwitch 900 or the MultiSwitch 600 chassis device, DIGITAL has extended the standard RMON definitions to include an additional field, called the *hubAlarmSlotNumber*. This field provides the number of the slot in which the offending module resides.

Apart from this additional field, the DIGITAL RMON-like traps operate the same as

RMON-generated traps and must be interpreted in the same way – that is, by performing a lookup in the appropriate alarm table and then in the associated event table. See "Interpreting RMON-Generated Traps" above.

The DIGITAL extensions to the standard RMON MIB (RFC 1757) are defined in the DECHUB900-HUBMGR-MIB-V3-0 MIB. The definitions for rising and falling alarms appear as follows:

```
hubRisingAlarm TRAP-TYPE
    ENTERPRISE mamPrivate
    VARIABLES { hubAlarmSlotNumber,
                hubAlarmIndex,
                hubAlarmVariable,
                hubAlarmSampleType,
                hubAlarmValue,
                hubAlarmRisingThreshold }
    DESCRIPTION
        "The SNMP trap which is generated when an alarm entry
         crosses its rising threshold and generates an event
         that is configured for sending SNMP traps.

         The enterprise Object Identifier for this trap is:


iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).
             dec(36).ema(2).decMIBextension(18).decHub900(11).
             mgmtAgent(1).mgmtAgentVersion2(2).mamPrivate(2)
         "
    ::= 1

hubFallingAlarm TRAP-TYPE
    ENTERPRISE mamPrivate
    VARIABLES { hubAlarmSlotNumber,
                hubAlarmIndex,
                hubAlarmVariable,
                hubAlarmSampleType,
                hubAlarmValue,
                hubAlarmRisingThreshold }
    DESCRIPTION
        "The SNMP trap which is generated when an alarm entry
         crosses its falling threshold and generates an event
         that is configured for sending SNMP traps.

         The enterprise Object Identifier for this trap is:


iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).
             dec(36).ema(2).decMIBextension(18).decHub900(11)
             mgmtAgent(1).mgmtAgentVersion2(2).mamPrivate(2)
         "
    ::= 2
```

DIGITAL Enterprise-Specific Traps

DIGITAL enterprise-specific traps are defined in DIGITAL private MIBs, such as the DECHUB900-HUBMGR-MIB-V3-0 MIB. There is no special procedure for interpreting DIGITAL enterprise-specific traps. If you have loaded the required DIGITAL MIBs and defined the DIGITAL trap definitions, you can use your Network Management System to look up the definitions associated with the received trap OIDs.