APPENDIX A

Using MIBs

This chapter describes how to perform tasks on the Cisco ASR 903 Series Aggregation Services Routers:

Cisco Unique Device Identifier Support

The ENTITY-MIB now supports the Cisco compliance effort for a Cisco unique device identifier (UDI) standard which is stored in IDPROM.

The Cisco UDI provides a unique identity for every Cisco product. The UDI is composed of three separate data elements which must be stored in the entPhysicalTable:

- Orderable product identifier (PID)—Product Identifier (PID). PID is the alphanumeric identifier used by customers to order Cisco products. Two examples include NM-1FE-TX or CISCO3745. PID is limited to 18 characters and must be stored in the entPhysicalModelName object.
- Version identifier (VID)—Version Identifier (VID). VID is the version of the PID. The VID indicates the number of times a product has versioned in ways that are reported to a customer. For example, the product identifier NM-1FE-TX may have a VID of V04. VID is limited to three alphanumeric characters and must be stored in the entPhysicalHardwareRev object.
- Serial number (SN)—Serial number is the 11-character identifier used to identify a specific part within a product and must be stored in the entPhysicalSerialNum object. Serial number content is defined by manufacturing part number 7018060-0000. The SN is accessed at the following website by searching on the part number 701806-0000:

  https://mco.cisco.com/servlet/mco.ecm.inbiz

  Serial number format is defined in four fields:

  – Location (L)
  – Year (Y)
  – Workweek (W)

  - Sequential serial ID (S)

  The SN label is represented as: LLLYYWWSSS.

Note The Version ID returns NULL for those old or existing cards whose IDPROMs do not have the Version ID field. Therefore, corresponding entPhysicalHardwareRev returns NULL for cards that do not have the Version ID field in IDPROM.

Cisco Redundancy Features

Redundancy creates a duplication of data elements and software functions to provide an alternative in case of failure. The goal of Cisco redundancy features is to cut over without affecting the link and protocol states associated with each interface and continue packet forwarding. The state of the interfaces and subinterfaces is maintained, along with the state of line cards and various packet processing hardware.

The levels of redundancy, redundancy verification, and related informatin is covered in these sections:

- Levels of Redundancy, page A-2
  - Route Switch Processor Redundancy (RPR) Mode, page A-3
  - Cisco Nonstop Forwarding and Stateful Switchover (NSF/SSO), page A-3
- Verifying Cisco ASR 903 Series Router Redundancy
- Verifying Cisco ASR 903 Series Router Redundancy, page A-3
- Related Information and Useful Links, page A-4

Levels of Redundancy

Cisco ASR 903 Series Router supports fault resistance by allowing a Cisco redundant Supervisor Engine (SE) to take over if the active SE fails. Redundancy prevents equipment failures from causing service outages, and supports hitless maintenance and upgrade activities. The state of the interfaces and subinterfaces are maintained along with the state of line cards and various packet processing hardware.

Redundant systems support two route switch processors. One acts as the active route switch processor while the other acts as the standby.

The route switch processor redundancy feature provides high availability for Cisco routers by switching over to the standby route switch processor when one of the following conditions occur:

- Cisco IOS XE software failure
- Cisco ASR 903 Series Route Switch Processor (RSP) hardware failure
- Software upgrade
- Maintenance procedure

Cisco ASR 903 Series Router can operate in one of two redundancy modes:

- Route Switch Processor Redundancy (RPR) Mode
- Cisco Nonstop Forwarding and Stateful Switchover (NSF/SSO)

In all modes, the standby RSP will take over when the active RSP fails.

Route Switch Processor Redundancy (RPR) Mode

This section describes the Route Processor Redundancy (RPR) mode for the Cisco ASR 903 Series Router.

When the switch is powered on, RPR runs between two Cisco SEs. The supervisor engine that boots first becomes the RPR active SE.

Cisco ASR 903 Series Router supports fault resistance by allowing a redundant SE to take over if the active SE fails.

Cisco Nonstop Forwarding and Stateful Switchover (NSF/SSO)

This section describes the Cisco Nonstop Forwarding and Stateful switchover mode. With Cisco NSF/SSO, Cisco ASR 903 Series Router can fail over from the active to the standby route switch processor almost immediately while continuing to forward packets. Cisco IOS XE software with Cisco NSF/SSO support on this platform enables immediate failover.

In networking devices running Cisco NSF/SSO, both RSPs must be running the same configuration so that the standby RSP is always ready to assume control following a fault on the active RSP. The configuration information is synchronized from the active RSP to the standby RSP at startup and each time when changes to the active RSP configuration occur.

Following an initial synchronization between the two processors, NFS/SSO maintains RSP state information between them, including forwarding information.

Cisco NSF works with the SSO to minimize the amount of time a network is unavailable to its users following a RSP failover in a router with dual RSPs. The Cisco NSF/SSO capability allows routers to detect a switchover and take the necessary actions to continue forwarding network traffic and to recover route information from peer devices.

Note For detailed information about the Cisco Nonstop Forwarding feature, go to: http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fsnsf20s.html

Note For detailed information about the Stateful Switchover feature, go to: http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fssso20s.html

Verifying Cisco ASR 903 Series Router Redundancy

To display information about the active and standby SE installed on a Cisco ASR 903 Series Router, use the **show redundancy** and **show redundancy states** commands. For RSP in R0 slot, the value of Unit ID is 48, same as ASCII "0" (hex 30). The value of Unit ID is 49, ASCII "1" (hex 31), for RSP in R1 slot.

Example A-1 Displaying Redundancy States from Active Processor

```
Router#show redundancy states
```

```
my state = 13 -ACTIVE
     peer state = 8  -STANDBY HOT
           Mode = Duplex
           Unit = Primary
        Unit ID = 48

Redundancy Mode (Operational) = sso
Redundancy Mode (Configured)  = sso
Redundancy State              = sso
    Manual Swact = enabled
 Communications = Up

   client count = 87
 client_notification_TMR = 30000 milliseconds
           RF debug mask = 0x0


Router#exit
```

Example A-2 Displaying Redundancy States from Standby Processor

```
Router#show redundancy state
my state = 8  -STANDBY HOT
     peer state = 13 -ACTIVE
           Mode = Duplex
           Unit = Secondary
        Unit ID = 49

Redundancy Mode (Operational) = sso
Redundancy Mode (Configured)  = sso
Redundancy State              = sso
    Manual Swact = cannot be initiated from this the standby unit
 Communications = Up

   client count = 87
 client_notification_TMR = 30000 milliseconds
           RF debug mask = 0x0

Router#
```

Related Information and Useful Links

The following URLs provide access to helpful information about the Cisco redundancy feature:

- Detailed information about Cisco Nonstop Forwarding:
  http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fsnsf20s.html
- Detailed information about the Stateful Switchover Feature:
  http://www.cisco.com/en/US/docs/ios/12_2s/feature/guide/fssso20s.html
- Detailed information about the Route Processor Redundancy Feature:
  http://www.cisco.com/en/US/docs/ios/12_1/12_1ex/feature/guide/12e_rpr.html

Managing Physical Entities

This section describes how to use SNMP to manage the physical entities (components) in the router by:

- Performing Inventory Management, page A-6

Purpose and Benefits

The physical entity management feature of the Cisco ASR 903 Series Router SNMP implementation does the following:

- Monitors and configures the status of field replaceable units (FRUs)
- Provides information about physical port to interface mappings
- Provides asset information for asset tagging
- Provides firmware and software information for chassis components

MIBs Used for Physical Entity Management

- CISCO-ENTITY-FRU-CONTROL-MIB—Contains objects used to monitor and configure the administrative and operational status of field replaceable units (FRUs), such as power supplies and line cards, that are listed in the entPhysicalTable of the ENTITY-MIB.
- CISCO-ENTITY-EXT-MIB - Contains Cisco defined extensions to the entPhysicalTable of the ENTITY-MIB to provide information for entities with an entPhysicalClass value of 'module' that have a CPU, RAM/NVRAM, and/or a configuration register.
- CISCO-ENTITY-SENSOR-MIB and ENTITY-SENSOR-MIB—Contain information about entities in the entPhysicalTable with an entPhysicalClass value of 'sensor'.
- CISCO-ENTITY-VENDORTYPE-OID-MIB—Contains the object identifiers (OIDs) for all physical entities in the router.
- ENTITY-MIB—Contains information for managing physical entities on the router. It also organizes the entities into a containment tree that depicts their hierarchy and relationship to each other. The MIB contains the following tables:
  - The entPhysicalTable describes each physical component (entity) in the router. The table contains an entry for the top-level entity (the chassis) and for each entity in the chassis. Each entry provides information about that entity: its name, type, vendor, and a description, and describes how the entity fits into the hierarchy of chassis entities.

    Each entity is identified by a unique index (*entPhysicalIndex*) that is used to access information about the entity in this and other MIBs.
  - The entAliasMappingTable maps each physical port's entPhysicalIndex value to its corresponding ifIndex value in the IF-MIB ifTable.
  - The entPhysicalContainsTable shows the relationship between physical entities in the chassis. For each physical entity, the table lists the entPhysicalIndex for each of the entity's child objects.
  - The entPhysicalIsFRU indicates whether or not a physical entity is considered a Field Replaceable Unit (FRU). For an entity identified as FRU, the physical entity contains the following device-specific information:
  - entPhysicalModelName- Product Identification (PID), same as orderable part number.
  - entPhysicalHardwareRev- Version Identification (VID)
  - entPhysicalSerialNum- Serial Number (SN)

- Cisco Unique Device Identifier (UDI)- Composed of PID, VID and SN, it provides a unique identity for all Cisco hardware products on which it has been enabled.

Performing Inventory Management

To obtain information about entities in the router, perform a MIB walk on the ENTITY-MIB entPhysicalTable.

As you examine sample entries in the ENTITY-MIB entPhysicalTable, consider the following:

- entPhysicalIndex—Uniquely identifies each entity in the chassis. This index is also used to access information about the entity in other MIBs.
- entPhysicalContainedIn—Indicates the entPhysicalIndex of a component's parent entity.
- entPhysicalParentRelPos—Shows the relative position of same-type entities that have the same entPhysicalContainedIn value (for example, chassis slots, and line card ports).

Note The container is applicable if the physical entity class is capable of containing one or more removable physical entities. For example, each (empty or full) slot in a chassis is modeled as a container. All removable physical entities should be modeled within a container entity, such as field-replaceable modules, fans, or power supplies.

ENTITY-MIB

The Entity physical table contains information for managing physical entities on the router. It also organizes the entities into a containment tree that depicts their hierarchy, and relationship with each other. Refer to the Appendix A, "Entity Containment Tree" section for the entity hierarchy. The following sample output contains the information for the ASR 903 AC power supply in power supply bay 0:

```
sw-mrrbu-nms-1:/opt/ats/ats5.1.0>getmany -v2c 10.86.0.50 public entityMIB | grep "\.11 ="
| more
entPhysicalDescr.11 = ASR 900 500W DC Power Supply
entPhysicalVendorType.11 = cevPowerSupply.332
entPhysicalContainedIn.11 = 10
entPhysicalClass.11 = powerSupply(6)
entPhysicalParentRelPos.11 = 0
entPhysicalName.11 = Power Supply Module 0
entPhysicalHardwareRev.11 = V00
entPhysicalFirmwareRev.11 =
entPhysicalSoftwareRev.11 =
entPhysicalSerialNum.11 =
entPhysicalMfgName.11 = Cisco Systems Inc
entPhysicalModelName.11 = A900-PWR550-D
entPhysicalAlias.11 =
entPhysicalAssetID.11 =
entPhysicalIsFRU.11 = true(1)
entPhysicalMfgDate.11 = 00 00  00 00   00 00  00 00
entPhysicalUris.11 = URN:CLEI:UNASSIGNED
entPhysicalChildIndex.10.11 = 11
sw-mrrbu-nms-1:/opt/ats/ats5.1.0>
```

For more information on this MIB, refer to ENTITY-MIB (RFC 4133), page 3-50.

Entity Containment Tree

The following is sample entity hierarchy for a ASR 903 device, with Mib Variables printed : <entPhysicalName entPhysicalClass>

```
ENTITY-MIB containment tree:
    |
|
    \-1 (cevChassis.1120) : Chassis : ASR 903 Series Router Chassis : ASR-903 : chassis
         |
         +-2 (cevContainer.246) : slot R0 : RSP Slot : {} : container
         |    |
         |    \-100 (cevModule.87.1) : module R0 : ASR 900 Route Switch Processor 1,
55Gbps throughput : A900-RSP1A-55 : module
         |         |
         |         +-101 (cevSensorModuleDeviceVoltage) : VNILE: VX1 R0/0 : VNILE: VX1 :
{} : sensor
         |         |
         |         +-102 (cevSensorModuleDeviceVoltage) : VNILE: VX2 R0/1 : VNILE: VX2 :
{} : sensor
         |         |
         |         +-103 (cevSensorModuleDeviceVoltage) : VNILE: VX3 R0/2 : VNILE: VX3 :
{} : sensor
         |         |
         |         +-104 (cevSensorModuleDeviceVoltage) : VNILE: VX4 R0/3 : VNILE: VX4 :
{} : sensor
         |         |
         |         +-105 (cevSensorModuleDeviceVoltage) : VNILE: VP1 R0/4 : VNILE: VP1 :
{} : sensor
         |         |
         |         +-106 (cevSensorModuleDeviceVoltage) : VNILE: VP2 R0/5 : VNILE: VP2 :
{} : sensor
         |         |
         |         +-107 (cevSensorModuleDeviceVoltage) : VNILE: VP3 R0/6 : VNILE: VP3 :
{} : sensor
         |         |
         |         +-108 (cevSensorModuleDeviceVoltage) : VNILE: VH R0/7 : VNILE: VH : {}
: sensor
         |         |
        |         +-109 (cevSensorModuleDeviceTemp) : Temp: Nile 0   R0/8 : Temp: Nile 0
: {} : sensor
         |         |
        |         +-110 (cevSensorModuleDeviceTemp) : Temp: N-Inlet  R0/9 : Temp: N-Inlet
: {} : sensor
         |         |
        |         +-111 (cevSensorModuleDeviceTemp) : Temp: Nile 1   R0/10 : Temp: Nile 1
: {} : sensor
         |         |
         |         +-112 (cevSensorModuleDeviceTemp) : Temp: N-Outlet R0/11 : Temp:
N-Outlet : {} : sensor
         |         |
         |         +-113 (cevSensorModuleDeviceVoltage) : VCPU : VX1 R0/12 : VCPU : VX1 :
{} : sensor
         |         |
         |         +-114 (cevSensorModuleDeviceVoltage) : VCPU : VX2 R0/13 : VCPU : VX2 :
{} : sensor
         |         |
         |         +-115 (cevSensorModuleDeviceVoltage) : VCPU : VX3 R0/14 : VCPU : VX3 :
{} : sensor
         |         |
         |         +-116 (cevSensorModuleDeviceVoltage) : VCPU : VX4 R0/15 : VCPU : VX4 :
{} : sensor
         |         |
```

```
         |         +-117 (cevSensorModuleDeviceVoltage) : VCPU : VP1 R0/16 : VCPU : VP1 :
{} : sensor
         |         |
         |         +-118 (cevSensorModuleDeviceVoltage) : VCPU : VP2 R0/17 : VCPU : VP2 :
{} : sensor
         |         |
         |         +-119 (cevSensorModuleDeviceVoltage) : VCPU : VP3 R0/18 : VCPU : VP3 :
{} : sensor
         |         |
         |         +-120 (cevSensorModuleDeviceVoltage) : VCPU : VP4 R0/19 : VCPU : VP4 :
{} : sensor
         |         |
         |         +-121 (cevSensorModuleDeviceVoltage) : VCPU : VH R0/20 : VCPU : VH : {}
: sensor
         |         |
         |        +-122 (cevSensorModuleDeviceTemp) : Temp: CPU      R0/21 : Temp: CPU
: {} : sensor
         |         |
         |         +-123 (cevSensorModuleDeviceTemp) : Temp: C-Inlet  R0/22 : Temp:
C-Inlet  : {} : sensor
         |         |
         |         +-124 (cevSensorModuleDeviceTemp) : Temp: PCIe Sw  R0/23 : Temp: PCIe
Sw  : {} : sensor
         |         |
         |         +-125 (cevSensorModuleDeviceTemp) : Temp: C-Outlet R0/24 : Temp:
C-Outlet : {} : sensor
         |         |
         |         +-131 (cevModuleCpuType) : cpu R0/0 : CPU 0 of module R0 : {} : cpu
         |         |
         |         +-132 (cevPortUSB) : usb R0/0 : USB Port : {} : port
         |         |    |
         |         |    \-133 (cevUsbFlash) : usb0 : USB Flash : {} : module
         |         |
         |         +-134 (cevPortUSB) : usb R0/1 : USB Port : {} : port
         |         |
         |         \-136 (cevPortGe) : NME R0 : Network Management Ethernet : {} : port
         |
         +-3 (cevContainer.246) : slot R1 : RSP Slot : {} : container
         |
         +-4 (cevContainer.249) : subslot 0/0 : IM Bay : {} : container
         |
         +-5 (cevContainer.249) : subslot 0/1 : IM Bay : {} : container
         |    |
         |    \-550 (cevModuleCommonCards.334) : IM subslot 0/1 : 8-port Gigabit Ethernet
Interface Module : A900-IM8T : module
         |         |
         |         +-551 (cevPortGe) : GigabitEthernet0/1/0 : A900-IM8T : {} : port
         |         |
         |         +-552 (cevPortGe) : GigabitEthernet0/1/1 : A900-IM8T : {} : port
         |         |
         |         +-553 (cevPortGe) : GigabitEthernet0/1/2 : A900-IM8T : {} : port
         |         |
         |         +-554 (cevPortGe) : GigabitEthernet0/1/3 : A900-IM8T : {} : port
         |         |
         |         +-555 (cevPortGe) : GigabitEthernet0/1/4 : A900-IM8T : {} : port
         |         |
         |         +-556 (cevPortGe) : GigabitEthernet0/1/5 : A900-IM8T : {} : port
         |         |
         |         +-557 (cevPortGe) : GigabitEthernet0/1/6 : A900-IM8T : {} : port
         |         |
         |         +-558 (cevPortGe) : GigabitEthernet0/1/7 : A900-IM8T : {} : port
         |         |
         |         +-567 (cevSensorModuleDeviceTemp) : subslot 0/1 temperature Sensor 0 :
subslot 0/1 temperature Sensor 0 : {} : s+
```

```
         |         |
         |         +-568 (cevSensorModuleDeviceTemp) : subslot 0/1 temperature Sensor 1 : 
subslot 0/1 temperature Sensor 1 : {} : s+
         |         |
         |         +-569 (cevSensorModuleDeviceTemp) : subslot 0/1 temperature Sensor 2 : 
subslot 0/1 temperature Sensor 2 : {} : s+
         |         |
         |         +-570 (cevSensorModuleDeviceTemp) : subslot 0/1 temperature Sensor 3 : 
subslot 0/1 temperature Sensor 3 : {} : s+
         |         |
         |         +-571 (cevSensorModuleDeviceTemp) : subslot 0/1 temperature Sensor 4 : 
subslot 0/1 temperature Sensor 4 : {} : s+
         |         |
         |         +-579 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 0 : 
subslot 0/1 voltage Sensor 0 : {} : sensor
         |         |
         |         +-580 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 1 : 
subslot 0/1 voltage Sensor 1 : {} : sensor
         |         |
         |         +-581 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 2 : 
subslot 0/1 voltage Sensor 2 : {} : sensor
         |         |
         |         +-582 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 3 : 
subslot 0/1 voltage Sensor 3 : {} : sensor
         |         |
         |         +-583 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 4 : 
subslot 0/1 voltage Sensor 4 : {} : sensor
         |         |
         |         \-584 (cevSensorModuleDeviceVoltage) : subslot 0/1 voltage Sensor 5 : 
subslot 0/1 voltage Sensor 5 : {} : sensor
         |
         +-6 (cevContainer.249) : subslot 0/2 : IM Bay : {} : container
         |    |
         |    \-800 (cevModuleCommonCards.336) : IM subslot 0/2 : 16 port T1/E1 IM : 
A900-IMA16D : module
         |         |
         |         +-817 (cevSensorModuleDeviceTemp) : subslot 0/2 temperature Sensor 0 : 
subslot 0/2 temperature Sensor 0 : {} : s+
         |         |
         |         +-818 (cevSensorModuleDeviceTemp) : subslot 0/2 temperature Sensor 1 : 
subslot 0/2 temperature Sensor 1 : {} : s+
         |         |
         |         +-819 (cevSensorModuleDeviceTemp) : subslot 0/2 temperature Sensor 2 : 
subslot 0/2 temperature Sensor 2 : {} : s+
         |         |
         |         +-829 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 0 : 
subslot 0/2 voltage Sensor 0 : {} : sensor
         |         |
         |         +-830 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 1 : 
subslot 0/2 voltage Sensor 1 : {} : sensor
         |         |
         |         +-831 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 2 : 
subslot 0/2 voltage Sensor 2 : {} : sensor
         |         |
         |         +-832 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 3 : 
subslot 0/2 voltage Sensor 3 : {} : sensor
         |         |
         |         +-833 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 4 : 
subslot 0/2 voltage Sensor 4 : {} : sensor
         |         |
         |         \-834 (cevSensorModuleDeviceVoltage) : subslot 0/2 voltage Sensor 5 : 
subslot 0/2 voltage Sensor 5 : {} : sensor
         |
         +-7 (cevContainer.249) : subslot 0/3 : IM Bay : {} : container
```

```
         |    |
         |    \-1050 (cevModuleCommonCards.334) : IM subslot 0/3 : 8-port Gigabit Ethernet
Interface Module : A900-IM8T : module
         |         |
         |         +-1051 (cevPortGe) : GigabitEthernet0/3/0 : A900-IM8T : {} : port
         |         |
         |         +-1052 (cevPortGe) : GigabitEthernet0/3/1 : A900-IM8T : {} : port
         |         |
         |         +-1053 (cevPortGe) : GigabitEthernet0/3/2 : A900-IM8T : {} : port
         |         |
         |         +-1054 (cevPortGe) : GigabitEthernet0/3/3 : A900-IM8T : {} : port
         |         |
         |         +-1055 (cevPortGe) : GigabitEthernet0/3/4 : A900-IM8T : {} : port
         |         |
         |         +-1056 (cevPortGe) : GigabitEthernet0/3/5 : A900-IM8T : {} : port
         |         |
         |         +-1057 (cevPortGe) : GigabitEthernet0/3/6 : A900-IM8T : {} : port
         |         |
         |         +-1058 (cevPortGe) : GigabitEthernet0/3/7 : A900-IM8T : {} : port
         |         |
         |        +-1067 (cevSensorModuleDeviceTemp) : subslot 0/3 temperature Sensor 0 :
subslot 0/3 temperature Sensor 0 : {} : +
         |         |
         |        +-1068 (cevSensorModuleDeviceTemp) : subslot 0/3 temperature Sensor 1 :
subslot 0/3 temperature Sensor 1 : {} : +
         |         |
         |        +-1069 (cevSensorModuleDeviceTemp) : subslot 0/3 temperature Sensor 2 :
subslot 0/3 temperature Sensor 2 : {} : +
         |         |
         |        +-1070 (cevSensorModuleDeviceTemp) : subslot 0/3 temperature Sensor 3 :
subslot 0/3 temperature Sensor 3 : {} : +
         |         |
         |        +-1071 (cevSensorModuleDeviceTemp) : subslot 0/3 temperature Sensor 4 :
subslot 0/3 temperature Sensor 4 : {} : +
         |         |
         |         +-1079 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 0 :
subslot 0/3 voltage Sensor 0 : {} : sensor
         |         |
         |         +-1080 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 1 :
subslot 0/3 voltage Sensor 1 : {} : sensor
         |         |
         |         +-1081 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 2 :
subslot 0/3 voltage Sensor 2 : {} : sensor
         |         |
         |         +-1082 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 3 :
subslot 0/3 voltage Sensor 3 : {} : sensor
         |         |
         |         +-1083 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 4 :
subslot 0/3 voltage Sensor 4 : {} : sensor
         |         |
         |         \-1084 (cevSensorModuleDeviceVoltage) : subslot 0/3 voltage Sensor 5 :
subslot 0/3 voltage Sensor 5 : {} : sensor
         |
         +-8 (cevContainer.249) : subslot 0/4 : IM Bay : {} : container
         |
         +-9 (cevContainer.249) : subslot 0/5 : IM Bay : {} : container
         |
         +-10 (cevContainer.247) : Power Supply Bay 0 : Power Supply Bay : {} : container
         |
         +-30 (cevContainer.247) : Power Supply Bay 1 : Power Supply Bay : {} : container
         |
         \-50 (cevContainer.248) : Fan Tray Bay 0 : Fan Tray Bay : {} : container
              |
              \-51 (cevFan.178) : Fan Tray : ASR 903 FAN Tray : A903-FAN : fan
                   |
```

```
                    +-52 (cevSensorModuleDeviceTemp) : Temp: FC PWM1 P2/0 : Temp: FC PWM1 :
{} : sensor
                     |
                     +-62 (cevFan.179) : Fan 2/0 : Fan : {} : fan
                     |
                     +-63 (cevFan.179) : Fan 2/1 : Fan : {} : fan
                     |
                     +-64 (cevFan.179) : Fan 2/2 : Fan : {} : fan
                     |
                     +-65 (cevFan.179) : Fan 2/3 : Fan : {} : fan
                     |
                     +-66 (cevFan.179) : Fan 2/4 : Fan : {} : fan
                     |
                     +-67 (cevFan.179) : Fan 2/5 : Fan : {} : fan
                     |
                     +-68 (cevFan.179) : Fan 2/6 : Fan : {} : fan
                     |
                     +-69 (cevFan.179) : Fan 2/7 : Fan : {} : fan
                     |
                     +-70 (cevFan.179) : Fan 2/8 : Fan : {} : fan
                     |
                     +-71 (cevFan.179) : Fan 2/9 : Fan : {} : fan
                     |
                     +-72 (cevFan.179) : Fan 2/10 : Fan : {} : fan
                     |
                     \-73 (cevFan.179) : Fan 2/11 : Fan : {} : fan
Line length limited to: <132>
Mib Variables printed : <entPhysicalName entPhysicalDescr entPhysicalModelName
entPhysicalClass>
```

Sample of ENTITY-MIB entPhysicalTable Entries

The samples in this section show how information is stored in the entPhysicalTable. An asset inventory can be performed by examining entPhysicalTable entries.

Note The sample outputs and values that appear throughout this chapter are examples of data that is displayed when using MIBs.

The following is a sample output that shows the ENTITY-MIB entPhysicalTable sample entries for a 8-port Gigabit Ethernet Interface Module card installed in a router chassis, and the IM inserted into the card.

ENTITY-MIB entPhysicalTable Entries

```
eentPhysicalDescr.1050 = 8-port Gigabit Ethernet Interface Module
entPhysicalDescr.1051 = A900-IM8T
entPhysicalDescr.1052 = A900-IM8T
entPhysicalDescr.1053 = A900-IM8T
entPhysicalDescr.1054 = A900-IM8T
entPhysicalDescr.1055 = A900-IM8T
entPhysicalDescr.1056 = A900-IM8T
entPhysicalDescr.1057 = A900-IM8T
entPhysicalDescr.1058 = A900-IM8T
entPhysicalDescr.1067 = subslot 0/3 temperature Sensor 0
entPhysicalDescr.1068 = subslot 0/3 temperature Sensor 1
entPhysicalDescr.1069 = subslot 0/3 temperature Sensor 2
entPhysicalDescr.1070 = subslot 0/3 temperature Sensor 3
entPhysicalDescr.1071 = subslot 0/3 temperature Sensor 4
entPhysicalDescr.1079 = subslot 0/3 voltage Sensor 0
entPhysicalDescr.1080 = subslot 0/3 voltage Sensor 1
```

```
entPhysicalDescr.1081 = subslot 0/3 voltage Sensor 2
entPhysicalDescr.1082 = subslot 0/3 voltage Sensor 3
entPhysicalDescr.1083 = subslot 0/3 voltage Sensor 4
entPhysicalDescr.1084 = subslot 0/3 voltage Sensor 5
....

entPhysicalVendorType.1050 = cevIM8pGeCu
entPhysicalVendorType.1051 = cevPortGe
entPhysicalVendorType.1052 = cevPortGe
entPhysicalVendorType.1053 = cevPortGe
entPhysicalVendorType.1054 = cevPortGe
entPhysicalVendorType.1055 = cevPortGe
entPhysicalVendorType.1056 = cevPortGe
entPhysicalVendorType.1057 = cevPortGe
entPhysicalVendorType.1058 = cevPortGe
entPhysicalVendorType.1067 = cevSensorModuleDeviceTemp
entPhysicalVendorType.1068 = cevSensorModuleDeviceTemp
entPhysicalVendorType.1069 = cevSensorModuleDeviceTemp
entPhysicalVendorType.1070 = cevSensorModuleDeviceTemp
entPhysicalVendorType.1071 = cevSensorModuleDeviceTemp
entPhysicalVendorType.1079 = cevSensorModuleDeviceVoltage
entPhysicalVendorType.1080 = cevSensorModuleDeviceVoltage
entPhysicalVendorType.1081 = cevSensorModuleDeviceVoltage
entPhysicalVendorType.1082 = cevSensorModuleDeviceVoltage
entPhysicalVendorType.1083 = cevSensorModuleDeviceVoltage
entPhysicalVendorType.1084 = cevSensorModuleDeviceVoltage
```

where **entPhysicalVendorType** identifies the unique vendor-specific hardware type of the physical entity.

```
entPhysicalContainedIn.1050 = 7
entPhysicalContainedIn.1051 = 1050
entPhysicalContainedIn.1052 = 1050
entPhysicalContainedIn.1053 = 1050
entPhysicalContainedIn.1054 = 1050
entPhysicalContainedIn.1055 = 1050
entPhysicalContainedIn.1056 = 1050
entPhysicalContainedIn.1057 = 1050
entPhysicalContainedIn.1058 = 1050
entPhysicalContainedIn.1067 = 1050
entPhysicalContainedIn.1068 = 1050
entPhysicalContainedIn.1069 = 1050
entPhysicalContainedIn.1070 = 1050
entPhysicalContainedIn.1071 = 1050
entPhysicalContainedIn.1079 = 1050
entPhysicalContainedIn.1080 = 1050
entPhysicalContainedIn.1081 = 1050
entPhysicalContainedIn.1082 = 1050
entPhysicalContainedIn.1083 = 1050
entPhysicalContainedIn.1084 = 1050
```

where **entPhysicalContainedIn** indicates the entPhysicalIndex of parent entity of the component.

```
entPhysicalClass.1050 = module(9)
entPhysicalClass.1051 = port(10)
entPhysicalClass.1052 = port(10)
entPhysicalClass.1053 = port(10)
entPhysicalClass.1054 = port(10)
entPhysicalClass.1055 = port(10)
entPhysicalClass.1056 = port(10)
entPhysicalClass.1057 = port(10)
entPhysicalClass.1058 = port(10)
```

```
entPhysicalClass.1067 = sensor(8)
entPhysicalClass.1068 = sensor(8)
entPhysicalClass.1069 = sensor(8)
entPhysicalClass.1070 = sensor(8)
entPhysicalClass.1071 = sensor(8)
entPhysicalClass.1079 = sensor(8)
entPhysicalClass.1080 = sensor(8)
entPhysicalClass.1081 = sensor(8)
entPhysicalClass.1082 = sensor(8)
entPhysicalClass.1083 = sensor(8)
entPhysicalClass.1084 = sensor(8)
```

where **entPhysicalClass** indicates the general type of hardware device.

```
entPhysicalParentRelPos.1050 = 0
entPhysicalParentRelPos.1051 = 0
entPhysicalParentRelPos.1052 = 1
entPhysicalParentRelPos.1053 = 2
entPhysicalParentRelPos.1054 = 3
entPhysicalParentRelPos.1055 = 4
entPhysicalParentRelPos.1056 = 5
entPhysicalParentRelPos.1057 = 6
entPhysicalParentRelPos.1058 = 7
entPhysicalParentRelPos.1067 = 0
entPhysicalParentRelPos.1068 = 1
entPhysicalParentRelPos.1069 = 2
entPhysicalParentRelPos.1070 = 3
entPhysicalParentRelPos.1071 = 4
entPhysicalParentRelPos.1079 = 12
entPhysicalParentRelPos.1080 = 13
entPhysicalParentRelPos.1081 = 14
entPhysicalParentRelPos.1082 = 15
entPhysicalParentRelPos.1083 = 16
entPhysicalParentRelPos.1084 = 17
```

where **entPhysicalParentRelPos** indicates the relative position of this child among the other entities.

```
entPhysicalName.1050 = IM subslot 0/3
entPhysicalName.1051 = GigabitEthernet0/3/0
entPhysicalName.1052 = GigabitEthernet0/3/1
entPhysicalName.1053 = GigabitEthernet0/3/2
entPhysicalName.1054 = GigabitEthernet0/3/3
entPhysicalName.1055 = GigabitEthernet0/3/4
entPhysicalName.1056 = GigabitEthernet0/3/5
entPhysicalName.1057 = GigabitEthernet0/3/6
entPhysicalName.1058 = GigabitEthernet0/3/7
entPhysicalName.1067 = subslot 0/3 temperature Sensor 0
entPhysicalName.1068 = subslot 0/3 temperature Sensor 1
entPhysicalName.1069 = subslot 0/3 temperature Sensor 2
entPhysicalName.1070 = subslot 0/3 temperature Sensor 3
entPhysicalName.1071 = subslot 0/3 temperature Sensor 4
entPhysicalName.1079 = subslot 0/3 voltage Sensor 0
entPhysicalName.1080 = subslot 0/3 voltage Sensor 1
entPhysicalName.1081 = subslot 0/3 voltage Sensor 2
entPhysicalName.1082 = subslot 0/3 voltage Sensor 3
entPhysicalName.1083 = subslot 0/3 voltage Sensor 4
entPhysicalName.1084 = subslot 0/3 voltage Sensor 5
```

where **entPhysicalName** provides the textual name of the physical entity.

```
entPhysicalHardwareRev.1050 = V00
```

```
entPhysicalHardwareRev.1051 =
entPhysicalHardwareRev.1052 =
entPhysicalHardwareRev.1053 =
entPhysicalHardwareRev.1054 =
entPhysicalHardwareRev.1055 =
entPhysicalHardwareRev.1056 =
entPhysicalHardwareRev.1057 =
entPhysicalHardwareRev.1058 =
entPhysicalHardwareRev.1067 =
entPhysicalHardwareRev.1068 =
entPhysicalHardwareRev.1069 =
entPhysicalHardwareRev.1070 =
entPhysicalHardwareRev.1071 =
entPhysicalHardwareRev.1079 =
entPhysicalHardwareRev.1080 =
entPhysicalHardwareRev.1081 =
entPhysicalHardwareRev.1082 =
entPhysicalHardwareRev.1083 =
entPhysicalHardwareRev.1084 =
```

where **entPhysicalHardware** provides the vendor-specific hardware revision number (string) for the physical entity.

```
entPhysicalSerialNum.1050 = N/A
entPhysicalSerialNum.1051 =
entPhysicalSerialNum.1052 =
entPhysicalSerialNum.1053 =
entPhysicalSerialNum.1054 =
entPhysicalSerialNum.1055 =
entPhysicalSerialNum.1056 =
entPhysicalSerialNum.1057 =
entPhysicalSerialNum.1058 =
entPhysicalSerialNum.1067 =
entPhysicalSerialNum.1068 =
entPhysicalSerialNum.1069 =
entPhysicalSerialNum.1070 =
entPhysicalSerialNum.1071 =
entPhysicalSerialNum.1079 =
entPhysicalSerialNum.1080 =
entPhysicalSerialNum.1081 =
entPhysicalSerialNum.1082 =
entPhysicalSerialNum.1083 =
entPhysicalSerialNum.1084 =
```

where **entPhysicalSerialNumber** provides the vendor-specific serial number (string) for the physical entity.

```
entPhysicalMfgName.1050 = Cisco Systems Inc
entPhysicalMfgName.1051 =
entPhysicalMfgName.1052 =
entPhysicalMfgName.1053 =
entPhysicalMfgName.1054 =
entPhysicalMfgName.1055 =
entPhysicalMfgName.1056 =
entPhysicalMfgName.1057 =
entPhysicalMfgName.1058 =
entPhysicalMfgName.1067 =
entPhysicalMfgName.1068 =
entPhysicalMfgName.1069 =
entPhysicalMfgName.1070 =
entPhysicalMfgName.1071 =
entPhysicalMfgName.1079 =
```

```
entPhysicalMfgName.1080 =
entPhysicalMfgName.1081 =
entPhysicalMfgName.1082 =
entPhysicalMfgName.1083 =
entPhysicalMfgName.1084 =
```

where **entPhysicalMfgName** provides the name of the manufacturer for the physical component.

```
entPhysicalModelName.1050 = A900-IM8T
entPhysicalModelName.1051 =
entPhysicalModelName.1052 =
entPhysicalModelName.1053 =
entPhysicalModelName.1054 =
entPhysicalModelName.1055 =
entPhysicalModelName.1056 =
entPhysicalModelName.1057 =
entPhysicalModelName.1058 =
entPhysicalModelName.1067 =
entPhysicalModelName.1068 =
entPhysicalModelName.1069 =
entPhysicalModelName.1070 =
entPhysicalModelName.1071 =
entPhysicalModelName.1079 =
entPhysicalModelName.1080 =
entPhysicalModelName.1081 =
entPhysicalModelName.1082 =
entPhysicalModelName.1083 =
entPhysicalModelName.1084 =
```

where **entPhysicalModelName** provides the vendor-specific model name string for the physical component.

```
entPhysicalIsFRU.1050 = true(1)
entPhysicalIsFRU.1051 = false(2)
entPhysicalIsFRU.1052 = false(2)
entPhysicalIsFRU.1053 = false(2)
entPhysicalIsFRU.1054 = false(2)
entPhysicalIsFRU.1055 = false(2)
entPhysicalIsFRU.1056 = false(2)
entPhysicalIsFRU.1057 = false(2)
entPhysicalIsFRU.1058 = false(2)
entPhysicalIsFRU.1067 = false(2)
entPhysicalIsFRU.1068 = false(2)
entPhysicalIsFRU.1069 = false(2)
entPhysicalIsFRU.1070 = false(2)
entPhysicalIsFRU.1071 = false(2)
entPhysicalIsFRU.1079 = false(2)
entPhysicalIsFRU.1080 = false(2)
entPhysicalIsFRU.1081 = false(2)
entPhysicalIsFRU.1082 = false(2)
entPhysicalIsFRU.1083 = false(2)
entPhysicalIsFRU.1084 = false(2)
```

where **entPhysicalIsFRU** indicates whether or not this physical entity is considered a FRU.

Note the following about the sample configuration:

- All chassis slots and IM ports have the same entPhysicalContainedIn value:
  - For chassis slots, entPhysicalContainedIn = 1 (the entPhysicalIndex of the chassis).
  - For IM ports, the entPhysicalContainedIn = 1050 (the entPhysicalIndex of the IM card).
- Each chassis slot and IM card port has a different entPhysicalParentRelPos to show its relative position within the parent object.

Determining the ifIndex Value for a Physical Port

The ENTITY-MIB **entAliasMappingIdentifier** maps a physical port to an interface by mapping the port's entPhysicalIndex to its corresponding ifIndex value in the IF-MIB ifTable. The following sample shows that the physical port whose entPhysicalIndex is 35 is associated with the interface whose ifIndex value is 4. (See the MIB for detailed descriptions of possible MIB values.)

```
entAliasMappingIdentifer.1813.0 = ifIndex.4
```

Monitoring and Configuring FRU Status

View objects in the CISCO-ENTITY-FRU-CONTROL-MIB cefcModuleTable to determine the administrative and operational status of FRUs, such as power supplies and line cards:

- cefcModuleAdminStatus—The administrative state of the FRU. Use cefcModuleAdminStatus to enable or disable the FRU.
- cefcModuleOperStatus—The current operational state of the FRU.

Figure A-1 shows a cefcModuleTable entry for a IM card whose entPhysicalIndex is 1000.

Figure A-1 Sample cefcModuleTable Entry

```
cefcModuleAdminStatus.1000 = enabled(1)
cefcModuleOperStatus.1000 = ok(2)
cefcModuleResetReason.1000 = unknown(1)
cefcModuleStatusLastChangeTime.1000 =
15865
```

See the "FRU Status Changes" section on page A-25 for information about how the router generates notifications to indicate changes in FRU status.

Using ENTITY-ALARM-MIB to Monitor Entity Alarms

CISCO-ENTITY-ALARM-MIB

CISCO-ENTITY-ALARM-MIB supports the monitoring of alarms generated by physical entities contained by the system, including chassis, slots, modules, ports, power supplies, etc. In order to monitor alarms generated by a physical entity, it must be represented by a row in the entPhysicalTable.

For more information on this MIB, refer to CISCO-ENTITY-ALARM-MIB, page 3-22.

Alarm Description Map Table

For each type of entity (represented by entPhysicalVendorType OID), this table contains a mapping between a unique ceAlarmDescrIndex and entPhysicalvendorType OID.

The ceAlarmDescrMapEntry is indexed by the CeAlarmDescrMapEntry.

Note The mapping between the ceAlarmDescrIndex and entPhysicalvendorType OID will exist only if the type of entity supports alarms monitoring, and it is in the device since device boot-up.

The following is a sample output of the alarm description map tables:

```
ptolemy:24> getmany 10.86.0.50 ceAlarmDescrMapTable
ceAlarmDescrVendorType.1 = cevContainerSFP
ceAlarmDescrVendorType.2 = cevContainerSlot
ceAlarmDescrVendorType.3 = cevContainer.246
ceAlarmDescrVendorType.4 = cevContainer.249
ceAlarmDescrVendorType.5 = cevContainer.247
ceAlarmDescrVendorType.6 = cevContainer.248
ceAlarmDescrVendorType.7 = cevSensorModuleDeviceTemp
ceAlarmDescrVendorType.8 = cevSensorModuleDeviceVoltage
ceAlarmDescrVendorType.9 = cevSensorModuleDeviceCurrent
ceAlarmDescrVendorType.10 = cevSensor
ceAlarmDescrVendorType.11 = cevModule.87.1
ceAlarmDescrVendorType.12 = cevPortUSB
ceAlarmDescrVendorType.13 = cevPortGe
ceAlarmDescrVendorType.14 = cevFan.178
ceAlarmDescrVendorType.15 = cevModuleCommonCards.334
ceAlarmDescrVendorType.16 = cevModuleCommonCards.336
```

The temperature sensor in ASR903 modules (RSP and PEM) contain cevSensorModuleDeviceTemp as entPhysicalvendorType OID. From the sample output, the index (ceAlarmDescrIndex) 7 is mapped to cevSensorModuleDeviceTemp, and the index 14 is mapped to the ASR 903 FAN module, which has cevFan.178 as entity physical vendor type OID.

Note The generic vendor OID, cevSenor, is used in case the Cisco ASR 903 SNMP agent is not able to determine the sensor type.

Alarm Description Table

The Alarm Description Table contains a description for each alarm type, defined by each vendor type employed by the system. Each alarm description entry (ceAlarmDescrEntry) is indexed by ceAlarmDescrIndex and ceAlarmDescrAlarmType.

The following is the sample output for all alarm types defined for all temperature types of entities in the Cisco ASR 903 modules. The index 9 is obtained from the ceAlarmDescrMapTable in the previous section:

```
ptolemy:26> getmany 10.86.0.50 ceAlarmDescrTable | grep "\.9\."
ceAlarmDescrSeverity.9.0 = 1
ceAlarmDescrSeverity.9.1 = 1
ceAlarmDescrSeverity.9.2 = 1
ceAlarmDescrSeverity.9.3 = 2
ceAlarmDescrSeverity.9.4 = 3
ceAlarmDescrSeverity.9.5 = 1
ceAlarmDescrSeverity.9.6 = 1
ceAlarmDescrSeverity.9.7 = 2
ceAlarmDescrSeverity.9.8 = 3
ceAlarmDescrText.9.0 = Faulty Ampere Sensor
ceAlarmDescrText.9.1 = Ampere Above Normal (Shutdown)
ceAlarmDescrText.9.2 = Ampere Above Normal
ceAlarmDescrText.9.3 = Ampere Above Normal
ceAlarmDescrText.9.4 = Ampere Above Normal
ceAlarmDescrText.9.5 = Ampere Below Normal (Shutdown)
ceAlarmDescrText.9.6 = Ampere Below Normal
ceAlarmDescrText.9.7 = Ampere Below Normal
ceAlarmDescrText.9.8 = Ampere Below Normal
```

Refer to the Bellcore Technical Reference TR-NWT-000474 Issue 4, December 1993, OTGR Section 4. Network Maintenance: Alarm and Control - Network Element. The severity is defined as follows:

- critical(1)
- major(2)
- minor(3)
- info(4)

The following is the list of alarms defined for the sensor:

```
Alarm type 1 is for crossing the shutdow threshold (above normal range).
Alarm type 2 is for crossing the critical threshold (above normal range).
Alarm type 3 is for crossing the major threshold (above normal range).
Alarm type 4 is for crossing the minor threshold (above normal range).
Alarm type 5 is for crossing the shutdow threshold (below normal range).
Alarm type 6 is for crossing the critical threshold (below normal range).
Alarm type 7 is for crossing the major threshold (below normal range).
Alarm type 8 is for crossing the minor threshold (below normal range).
```

These alarm types are defined for all sensor physical entity type. The only difference is that different sensor physical type have different ceAlarmDescrText. The temperature sensor has "TEMP" and the voltage sensor has "Volt" in the alarm description text.

The following is the sample output of all alarm types. It is defined for the Cisco ASR903 Router fan module which has cevFan.178 as vendor type OID and is mapped to the ceAlarmDescrIndex

```
ptolemy:27> getmany 10.86.0.50 ceAlarmDescrTable | grep "\.14\."
ceAlarmDescrSeverity.14.0 = 1
ceAlarmDescrSeverity.14.1 = 1
ceAlarmDescrSeverity.14.2 = 1
ceAlarmDescrSeverity.14.3 = 2
ceAlarmDescrSeverity.14.4 = 2
ceAlarmDescrSeverity.14.5 = 2
ceAlarmDescrSeverity.14.6 = 2
ceAlarmDescrSeverity.14.7 = 2
ceAlarmDescrSeverity.14.8 = 2
```

```
ceAlarmDescrSeverity.14.9 = 2
ceAlarmDescrSeverity.14.10 = 2
ceAlarmDescrSeverity.14.11 = 2
ceAlarmDescrSeverity.14.12 = 2
ceAlarmDescrSeverity.14.13 = 2
ceAlarmDescrSeverity.14.14 = 2
ceAlarmDescrText.14.0 = Fan Tray/Module Failure
ceAlarmDescrText.14.1 = All Fans Failed
ceAlarmDescrText.14.2 = Multiple Fan Failures
ceAlarmDescrText.14.3 = Fan 0 Failure
ceAlarmDescrText.14.4 = Fan 1 Failure
ceAlarmDescrText.14.5 = Fan 2 Failure
ceAlarmDescrText.14.6 = Fan 3 Failure
ceAlarmDescrText.14.7 = Fan 4 Failure
ceAlarmDescrText.14.8 = Fan 5 Failure
ceAlarmDescrText.14.9 = Fan 6 Failure
ceAlarmDescrText.14.10 = Fan 7 Failure
ceAlarmDescrText.14.11 = Fan 8 Failure
ceAlarmDescrText.14.12 = Fan 9 Failure
ceAlarmDescrText.14.13 = Fan 10 Failure
ceAlarmDescrText.14.14 = Fan 11 Failure
```

Alarm Table

The Alarm Table specifies alarm control and status information related to each physical entity contained by the system. The table includes the alarms currently being asserted by each physical entity that is capable of generating alarms. Each physical entity in entity physical table that is capable of generating alarms has an entry in this table.The alarm entry (ceAlarmEntry) is indexed by the entity physical index (entPhysicalIndex). The following is a list of MIB objects in the alarm entry:

- **ceAlarmFilterProfile**
  The alarm filter profile object contains an integer value that uniquely identifies an alarm filter profile associated with the corresponding physical entity. An alarm filter profile controls which alarm types the agent will monitor and signal for the corresponding physical entity. The default value of this object is 0, the agent monitors and signals all alarms associated with the corresponding physical entity.
- **ceAlarmSeverity**
  This object specifies the highest severity alarm currently being asserted by the corresponding physical entity.
  A value of '0' indicates that the corresponding physical entity is not currently asserting any alarms.
- **ceAlarmList**
  This object specifies those alarms currently being asserted by the corresponding physical entity. If an alarm is being asserted by the physical entity, then the corresponding bit in the alarm list is set to a one. The alarm list is defined as octet string and its size ranges from 0 to 32.
  - If the physical entity is not currently asserting any alarms, then the list will have a length of zero, otherwise it will have a length of 32.
  - An OCTET STRING represents an alarm list, in which each bit represents an alarm type:

octet 1:

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
```

```
| | | | | | | +- Alarm type 0
| | | | | | +--- Alarm type 1
| | | | | +----- Alarm type 2
| | | | +------- Alarm type 3
| | | +--------- Alarm type 4
| | +----------- Alarm type 5
| +------------- Alarm type 6
+--------------- Alarm type 7
```

octet 2:

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
| | | | | | | +- Alarm type 8
| | | | | | +--- Alarm type 9
| | | | | +----- Alarm type 10
| | | | +------- Alarm type 11
| | | +--------- Alarm type 12
| | +----------- Alarm type 13
| +------------- Alarm type 14
+--------------- Alarm type 15
```

octet xx

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| |
+-+-+-+-+-+-+-+-+
| | | | | | | | |
| | | | | | | +- Alarm type 248
| | | | | | +--- Alarm type 249
| | | | | +----- Alarm type 250
| | | | +------- Alarm type 251
| | | +--------- Alarm type 252
| | +----------- Alarm type 253
| +------------- Alarm type 254
+--------------- Alarm type 255
```

The entity physical table (entPhysicalTable in ENTITY-MIB), indicates that the Cisco ASR903 Router AC power supply in power supply bay 0 has 4 as entPhysicalIndex .

The following is the sample output of the alarm list for the power supply in PS bay 0:

```
ptolemy-248->getone -v2c 9.0.0.56 public ceAlarmList.4
ceAlarmList.10 =
01 00  00 00   00 00  00 00    00 00  00 00   00 00  00 00
00 00  00 00   00 00  00 00    00 00  00 00   00 00  00 00
```

octet 1: 09

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
0 0 0 0 1 0 0 1
+-+-+-+-+-+-+-+-+
| | | | | | | | |
```

```
| | | | | | | +- Alarm type 0
| | | | | | +--- Alarm type 1
| | | | | +----- Alarm type 2
| | | | +------- Alarm type 3
| | | +--------- Alarm type 4
| | +----------- Alarm type 5
| +------------- Alarm type 6
+--------------- Alarm type 7
```

From the sample output in the Alarm Description Table section and the alarm mapping table, the ASR903 AC power supply in the bay 0 has the following alarms asserted:

Alarm type 0 : Power Supply Failure

Alarm type 3 : Fan 0 Failure

The following is the output of the **show facility-alarm status** command; it displays all alarms currently asserted in the device:

```
System Totals  Critical: 11  Major: 0  Minor: 0

Source                  Severity      Description [Index]
------                  --------      -------------------
Power Supply Bay 0          CRITICAL      Power Supply/FAN Module Missing [0]
TenGigabitEthernet0/1/0     CRITICAL      Physical Port Link Down [35]
TenGigabitEthernet0/2/0     CRITICAL      Physical Port Link Down [35]
xcvr container 0/4/0        CRITICAL      Transceiver Missing - Link Down [1]
GigabitEthernet0/4/4        CRITICAL      Physical Port Link Down [1]
GigabitEthernet0/4/5        CRITICAL      Physical Port Link Down [1]
GigabitEthernet0/4/6        CRITICAL      Physical Port Link Down [1]
xcvr container 0/5/0        CRITICAL      Transceiver Missing - Link Down [1]
GigabitEthernet0/5/1        CRITICAL      Physical Port Link Down [1]
xcvr container 0/5/2        CRITICAL      Transceiver Missing - Link Down [1]
GigabitEthernet0/5/3        CRITICAL      Physical Port Link Down [1]
```

Alarm History Table

The Alarm History Table, ceAlarmHistTable, contains history of alarms both asserted and cleared generated by the agent. The ceAlarmHistTableSize is used to control the size of the alarm history table. A value of 0 prevents any history from being retained in this table. If the capacity of the ceAlarmHistTable has reached the value specified by this object, then the agent deletes the oldest entity in order to accommodate a new entry.

The ceAlarmHistLastIndex object contains the last index corresponding to the last entry added to the table by the snmp agent in the device. If the management client uses notifications listed in the Appendix A, "Alarm Notifications" defined in CISCO-ENTITY-ALARM-MIB module, then it can poll this object to determine whether it has missed a notification sent by the agent.

The following is a list of MIB objects defined in the ceAlarmHistEntry, which is indexed by the ceAlarmHistIndex:

- **ceAlarmHistIndex**
  This is an integer value uniquely identifying the entry in the table. The value of this object starts at '1' and monotonically increases for each alarm (asserted or cleared) added to the alarm history table. If the value of this object is '4294967295', it will be reset to '1', upon monitoring the next alarm condition transition.
- **ceAlarmHistType**
  This object indicates that the entry is added as a result of of an alarm being asserted or cleared.

- **ceAlarmHistEntPhysicalIndex**
  This object contains the entPhysicalIndex of the physical entity that generated the alarm.
- **ceAlarmHistAlarmType**
  This object specifies the type of alarm generated.
- **ceAlarmHistSeverity**
  This object specifies the severity of the alarm generated.
- **ceAlarmHistTimeStamp**
  This object specifies the value of the sysUpTime object at the time the alarm is generated.

Example A-3 Displaying Sample Output for the Alarm History

```
ptolemy:33> getnext 10.86.0.50 ceAlarmHistory
ceAlarmHistTableSize.0 = 200  -> the size of alarm history table
ptolemy:34> getnext 10.86.0.50 ceAlarmHistTableSize.0
ceAlarmHistLastIndex.0 = 21  -> the index for the last alarm added
```

Example A-4 Displaying the Last Alarm Action (asserted or cleared) Added to the Alarm History Table

```
ptolemy:37> getmany 10.86.0.50 ceAlarmHistTable | grep "\.99"
ceAlarmHistType.99 = cleared(2)
ceAlarmHistEntPhysicalIndex.99 = 800
ceAlarmHistAlarmType.99 = 0
ceAlarmHistSeverity.99 = major(2)
ceAlarmHistTimeStamp.99 = 20033972
```

At this point, the EMS application should already have all information regarding the physical entity and the entity alarm type defined for the physical entity.

Example A-5 Displaying the Physical Entity with Value 51 as entPhysicalIndex

```
entPhysicalDescr.51 = ASR 903 FAN Tray
entPhysicalVendorType.51 = cevFan.178
entPhysicalContainedIn.51 = 50
entPhysicalClass.51 = fan(7)
entPhysicalParentRelPos.51 = 0
entPhysicalName.51 = Fan Tray
entPhysicalHardwareRev.51 = V00
entPhysicalFirmwareRev.51 =
entPhysicalSoftwareRev.51 =
entPhysicalSerialNum.51 =
entPhysicalMfgName.51 = Cisco Systems Inc
entPhysicalModelName.51 = A903-FAN
```

Example A-6 Displaying the Alarm Type Defined for cevFan.178

```
ceAlarmDescrSeverity.18.0 = 1
ceAlarmDescrSeverity.18.1 = 1
ceAlarmDescrSeverity.18.2 = 1
ceAlarmDescrSeverity.18.3 = 2
ceAlarmDescrSeverity.18.4 = 2
ceAlarmDescrSeverity.18.5 = 2
ceAlarmDescrSeverity.18.6 = 2
ceAlarmDescrSeverity.18.7 = 2
ceAlarmDescrSeverity.18.8 = 2
ceAlarmDescrSeverity.18.9 = 2
ceAlarmDescrSeverity.18.10 = 2
```

```
ceAlarmDescrSeverity.18.11 = 2
ceAlarmDescrSeverity.18.12 = 2
ceAlarmDescrSeverity.18.13 = 2
ceAlarmDescrSeverity.18.14 = 2
ceAlarmDescrText.18.0 = Fan Tray/Module Failure
ceAlarmDescrText.18.1 = All Fans Failed
ceAlarmDescrText.18.2 = Multiple Fan Failures
ceAlarmDescrText.18.3 = Fan 0 Failure
ceAlarmDescrText.18.4 = Fan 1 Failure
ceAlarmDescrText.18.5 = Fan 2 Failure
ceAlarmDescrText.18.6 = Fan 3 Failure
ceAlarmDescrText.18.7 = Fan 4 Failure
ceAlarmDescrText.18.8 = Fan 5 Failure
ceAlarmDescrText.18.9 = Fan 6 Failure
ceAlarmDescrText.18.10 = Fan 7 Failure
ceAlarmDescrText.18.11 = Fan 8 Failure
ceAlarmDescrText.18.12 = Fan 9 Failure
ceAlarmDescrText.18.13 = Fan 10 Failure
ceAlarmDescrText.18.14 = Fan 11 Failure
```

Alarm Notifications

CISCO-ENTITY-ALARM-MIB supports the alarm asserted (ceAlarmAsserted) and alarm cleared (ceAlarmCleared) notifications. The notification can be enabled by setting the ceAlarmNotifiesEnable object through the snmp SET. The ceAlarmNotifiesEnable contains the severity level of the alarms notification or the value 0:

```
severity 1: critical       Service affecting Condition
severity 2: major          Immediate action needed
severity 3: minor          Minor warning conditions
severity 4: informational  Informational messages
```

The severity 4 will enable notification for all severity level.

The severity 3 will enable notifications for severity 1, 2, and 3.

The severity 2 will enable notifications for severity 1 and 2.

The severity 1 will enable notifications for severity 1 only.

The value of 0 will disable the alarm notification.

The alarm notification can be enabled or disabled via the CLI command. Use the "NO" form to disable the alarm notification:

```
snmp-server enable traps alarm [critical, major, minor, information]
no snmp-server enable traps alarm [critical, major, minor, information]
```

The alarm notification contains exactly the same information described in alarm history entry. Refer to the Alarm History Table Section for the MIB objects and to interpret the alarm notifications received.

Example A-7 Displaying the Sample Notification Received

```
sysUpTime.0 = 161726
 snmpTrapOID.0 = ceAlarmAsserted
 ceAlarmHistEntPhysicalIndex.103 = 800
 ceAlarmHistAlarmType.103 = 0
 ceAlarmHistSeverity.103 = 2
 ceAlarmHistTimeStamp.103 = 161725
 ceAlarmDescrText.17.0 = Unknown state

sysUpTime.0 = 161728
```

```
 snmpTrapOID.0 = ceAlarmCleared
 ceAlarmHistEntPhysicalIndex.104 = 801
 ceAlarmHistAlarmType.104 = 5
 ceAlarmHistSeverity.104 = 3
 ceAlarmHistTimeStamp.104 = 161725
 ceAlarmDescrText.18.5 = Receiver has loss of signal

sysUpTime.0 = 161729
 snmpTrapOID.0 = ceAlarmCleared
 ceAlarmHistEntPhysicalIndex.105 = 801
 ceAlarmHistAlarmType.105 = 12
 ceAlarmHistSeverity.105 = 3
 ceAlarmHistTimeStamp.105 = 161725
 ceAlarmDescrText.18.12 = Ds1 Physical Port Link Down
```

Generating SNMP Notifications

This section provides information about the SNMP notifications generated in response to events and conditions on the router, and describes how to identify the hosts that are to receive notifications.

- Identifying Hosts to Receive Notifications
- Configuration Changes
- FRU Status Changes

Identifying Hosts to Receive Notifications

You can use the CLI or SNMP to identify hosts to receive SNMP notifications and to specify the types of notifications they are to receive (notifications or informs). For CLI instructions, see the "Monitoring Notifications" section on page 4-1. To use SNMP to configure this information, use the following MIB objects:

Use SNMP-NOTIFICATION-MIB objects, including the following, to select target hosts and specify the types of notifications to generate for those hosts:

- snmpNotifyTable—Contains objects to select hosts and notification types:
  - snmpNotifyTag is an arbitrary octet string (a tag value) used to identify the hosts to receive SNMP notifications. Information about target hosts is defined in the snmpTargetAddrTable (SNMP-TARGET-MIB), and each host has one or more tag values associated with it. If a host in snmpTargetAddrTable has a tag value that matches this snmpNotifyTag value, the host is selected to receive the types of notifications specified by snmpNotifyType.
  - snmpNotifyType is the type of SNMP notification to send: notification(1) or inform(2).
- snmpNotifyFilterProfileTable and snmpNotifyFilterTable—Use objects in these tables to create notification filters to limit the types of notifications sent to target hosts.

Use SNMP-TARGET-MIB objects to configure information about the hosts to receive notifications:

- snmpTargetAddrTable—Transport addresses of hosts to receive SNMP notifications. Each entry provides information about a host address, including a list of tag values:
  - snmpTargetAddrTagList—A set of tag values associated with the host address. If a host's tag value matches snmpNotifyTag, the host is selected to receive the types of notifications defined by snmpNotifyType.
- snmpTargetParamsTable—SNMP parameters to use when generating SNMP notifications.

Use the notification enable objects in appropriate MIBs to enable and disable specific SNMP notifications. For example, to generate mplsLdpSessionUp or mplsLdpSessionDown notifications, the MPLS-LDP-MIB object mplsLdpSessionUpDownTrapEnable must be set to enabled(1).

Configuration Changes

If entity notifications are enabled, the router generates an entConfigChange notification (ENTITY-MIB) when the information in any of the following tables changes (which indicates a change to the router configuration):

- entPhysicalTable
- entAliasMappingTable
- entPhysicalContainsTable

Note A management application that tracks configuration changes checks the value of the entLastChangeTime object to detect any entConfigChange notifications that were missed as a result of throttling or transmission loss.

Enabling notifications for Configuration Changes

To configure the router to generate an entConfigChange notification each time its configuration changes, enter the following command from the CLI. Use the **no** form of the command to disable the notifications.

```
Router(config)# snmp-server enable traps entity
Router(config)# no snmp-server enable traps entity
```

FRU Status Changes

If FRU notifications are enabled, the router generates the following notifications in response to changes in the status of an FRU:

- cefcModuleStatusChange—The operational status (cefcModuleOperStatus) of an FRU changes.
- cefcFRUInserted—An FRU is inserted in the chassis. The notification indicates the entPhysicalIndex of the FRU and the container it was inserted in.
- cefcFRURemoved—An FRU is removed from the chassis. The notification indicates the entPhysicalIndex of the FRU and the container it was removed from.

Note See the CISCO-ENTITY-FRU-CONTROL-MIB for more information about these notifications.

Enabling FRU Notifications

To configure the router to generate notifications for FRU events, enter the following command from the CLI. Use the **no** form of the command to disable the notifications.

```
Router(config)# snmp-server enable traps fru-ctrl
Router(config)# no snmp-server enable traps fru-ctrl
```

To enable FRU notifications through SNMP, set cefcMIBEnableStatusNotification to true(1). Disable the notifications by setting cefcMIBEnableStatusNotification to false(2).

Monitoring Router Interfaces

This section provides information about how to monitor the status of router interfaces to see if there is a problem or a condition that might affect service on the interface. To determine if an interface is Down or experiencing problems, you can:

Check the Interface's Operational and Administrative Status

To check the status of an interface, view the following IF-MIB objects for the interface:

- ifAdminStatus—The administratively configured (desired) state of an interface. Use ifAdminStatus to enable or disable the interface.
- ifOperStatus—The current operational state of an interface.

Monitor linkDown and linkUp Notifications

To determine if an interface has failed, you can monitor linkDown and linkUp notifications for the interface. See the "Enabling Interface linkUp/linkDown Notifications" section on page A-26 for instructions on how to enable these notifications.

- linkDown—Indicates that an interface failed or is about to fail.
- linkUp—Indicates that an interface is no longer in the Down state.

Enabling Interface linkUp/linkDown Notifications

To configure SNMP to send a notification when a router interface changes state to Up (ready) or Down (not ready), perform the following steps to enable linkUp and linkDown notifications:

Step 1 Issue the following CLI command to enable linkUp and linkDown notifications for most, but not necessarily all, interfaces:

```
Router(config)# snmp-server enable traps snmp linkdown linkup
```

Step 2 View the setting of the ifLinkUpDownTrapEnable object (IF-MIB ifXTable) for each interface to determine if linkUp and linkDown notifications are enabled or disabled for that interface.

Step 3 To enable linkUp and linkDown notifications on an interface, set ifLinkUpDownTrapEnable to enabled(1). To configure the router to send linkDown notifications only for the lowest layer of an interface, see the "SNMP Notification Filtering for linkDown Notifications" section on page A-27.

Step 4 To enable the Internet Engineering Task Force (IETF) standard for linkUp and linkDown notifications, issue the following command. (The IETF standard is based on RFC 2233.)

```
Router(config)# snmp-server trap link ietf
```

Step 5 To disable notifications, use the **no** form of the appropriate command.

SNMP Notification Filtering for linkDown Notifications

Use the SNMP notification filtering feature to filter linkDown notifications so that SNMP sends a linkDown notification only if the main interface goes down. If an interfaces goes down, all of its subinterfaces go down, which results in numerous linkDown notifications for each subinterface. This feature filters out those subinterface notifications.

This feature is turned off by default. To enable the SNMP notification filtering feature, issue the following CLI command. Use the **no** form of the command to disable the feature.

```
[no] snmp ifmib trap throttle
```

Billing Customers for Traffic

This section describes how to use SNMP interface counters to determine the amount to bill customers for traffic.

Input and Output Interface Counts

The router maintains information about the number of packets and bytes that are received on an input interface and transmitted on an output interface.

For detailed constraints about IF-MIB counter support, see the "IF-MIB (RFC 2863)" section on page 3-57.

Read the following important information about the IF-MIB counter support:

- Unless noted, all IF-MIB counters are supported on Cisco ASR 903 Series Router interfaces.
- For IF-MIB high capacity counter support, we conform to the RFC 2863 standard. The RFC 2863 standard states that for interfaces that operate:
  - At 20 million bits per second or less, 32-bit byte and packet counters *must* be supported.
  - Faster than 20 million bits per second and slower than 650,000,000 bits per second, 32-bit packet counters and 64-bit octet counters *must* be supported.
  - At 650,000,000 bits per second or faster, 64-bit packet counters *and* 64-bit octet counters *must* be supported.

Using IF-MIB Counters

This section describes the IF-MIB counters and how you can use them on various interfaces and subinterfaces. The subinterface counters are specific to the protocols. This section addresses the IF-MIB counters for ATM interfaces.

The IF-MIB counters are defined with respect to lower and upper layers:

- ifInDiscards—The number of inbound packets which were discarded, even though no errors were detected to prevent their being deliverable to a higher-layer protocol. One reason for discarding such a packet could be to free up buffer space.
- IfInErrors—The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol for packet-oriented interfaces.

- ifInUnknownProtos—The number of packets received through the interface which were discarded because of an unknown or unsupported protocol for packet-oriented interfaces.
- ifOutDiscards—The number of outbound packets which were discarded even though no errors were detected to prevent their being transmitted. One reason for discarding such a packet is to free up buffer space.
- ififOutErrors—The number of outbound packets that could not be transmitted because of errors for packet-oriented interfaces.

The logical flow for counters works as follows:

1. When a packet arrives on an interface, check for the following:
   a. Error in packet—If any errors are detected, increment ifInErrors and drop the packet.
   b. Protocol errors—If any errors are detected, increment ifInUnknownProtos and drop the packet.
   c. Resources (buffers)—If unable to get resources, increment ifInDiscards and drop the packet.
   d. Increment ifInUcastPkts/ ifInNUcastPkts and process the packet (At this point, increment the ifInOctets with the size of packet).
2. When a packet is to be sent out of an interface:
   a. Increment ifOutUcasePkts/ ifOutNUcastPkts (Here we also increment ifOutOctets with the size of packet).
   b. Check for error in packet and if there are any errors in packet, increment ifOutErrors and drop the packet.
   c. Check for resources (buffers) and if you cannot get resources then increment ifOutDiscards and drop packet.

This following output is an example IF-MIB entries:

IfXEntry ::=

```
SEQUENCE {
    ifName                  DisplayString,
    ifInMulticastPkts       Counter32,
    ifInBroadcastPkts       Counter32,
    ifOutMulticastPkts      Counter32,
    ifOutBroadcastPkts      Counter32,
    ifHCInOctets            Counter64,
    ifHCInUcastPkts         Counter64,
    ifHCInMulticastPkts     Counter64,
    ifHCInBroadcastPkts     Counter64,
    ifHCOutOctets           Counter64,
    ifHCOutUcastPkts        Counter64,
    ifHCOutMulticastPkts    Counter64,
    ifHCOutBroadcastPkts    Counter64,
    ifLinkUpDownTrapEnable  INTEGER,
    ifHighSpeed             Gauge32,
    ifPromiscuousMode       TruthValue,
    ifConnectorPresent      TruthValue,
    ifAlias                 DisplayString,
    ifCounterDiscontinuityTime TimeStamp
```

Sample Counters

The high capacity counters are 64-bit versions of the basic ifTable counters. They have the same basic semantics as their 32-bit counterparts; their syntax is extended to 64 bits.

Table A-1 lists capacity counter object identifiers (OIDs).

Table A-1 Capacity Counters Object Identifiers

Name	Object Identifier (OID)
ifHCInOctets	::= { ifXEntry 6 }
ifHCInUcastPkts	::= { ifXEntry 7 }
ifHCInMulticastPkts	::= { ifXEntry 8 }
ifHCInBroadcastPkts	::= { ifXEntry 9 }
ifHCOutOctets	::= { ifXEntry 10 }
ifHCOutUcastPkts	::= { ifXEntry 11 }
ifHCOutMulticastPkts	::= { ifXEntry 12 }
ifHCOutBroadcastPkts	::= { ifXEntry 13 }
ifLinkUpDownTrapEnable	::= { ifXEntry 14 }
ifHighSpeed	::= { ifXEntry 15 }
ifPromiscuousMode	::= { ifXEntry 16 }
ifConnectorPresent	::= { ifXEntry 17 }
ifAlias	::= { ifXEntry 18 }
ifCounterDiscontinuityTime	::= { ifXEntry 19 }

Related Information and Useful Links

The following URLs provide access to helpful information about Cisco IF-MIB counters:

- Frequently asked questions about SNMP counters:

  http://www.cisco.com/en/US/customer/tech/tk648/tk362/technologies_q_and_a_item09186a00800b69ac.shtml

- Access Cisco IOS XE MIB Tools from the following URL:

  http://tools.cisco.com/ITDIT/MIBS/servlet/index

Overview of Interface Module

An Interface Module (IM) is a type of port adapter that inserts into a subslot to provide network connectivity and increased interface port density. The IM helps in providing services related to VPNs, pseudowires, and so on.

The different types of IM cards supported are:

- A900-IMA8S (8-port Gigabit Ethernet Interface Module using SFPs)
- A900-IM8T (8-port Gigabit Ethernet Interface Module with RJ-45 connectors)
- A900-IMA16D (16 port T1/E1 Interface Module)
- A900-IM1X (Ten Gigabit Ethernet Interface Module)

Displaying the Hardware Type

These commands in the Cisco ASR 903 Series Router help to display the hardware details:

- **show platform**

```
Router# show platform

Chassis type: ASR-903
Slot Type State Insert time (ago)
--------- ------------------- --------------------- -----------------
0/1 A900-IM8T ok 3d06h
0/2 A900-IMA16D ok 23:21:45
0/3 A900-IM8T ok 3d06h
R0 A900-RSP1A-55 ok, active 3d07h
F0 ok, active 3d07h
P0 Unknown ps, fail never
P1 Unknown ps, fail never
P2 A903-FAN ok 3d07h

Slot CPLD Version Firmware Version
--------- ------------------- ---------------------------------------
R0 11070719 12.2(20110714:143033) [ashohegd-ROMM...
F0 11070719 12.2(20110714:143033) [ashohegd-ROMM...
```

- **show hardware module subslot**

```
Router# show hardware-module
Router# sh hw-module subslot 0/3 ?
entity entity MIB details - not intended for production use
fpd Show Field Programmable Devices (FPD) information
oir Show oir summary
sensors Environmental sensor summary
subblock subblock details - not intended for production use
tech-support Show subslot information for Tech-Support


Router# show hardware-module subslot 0/1 entity
WARNING: This command is not intended for production use and should only be used under the
supervision of Cisco Systems technical support personnel.

Entity state for SPA in subslot 0/1
SPA type:                 (0x73D) 8xGE IM
last spa type:            (0x73D) 8xGE IM
oper_status:              (1) ok
card status:              (2) full
last trap: spa type:      (0x73D) 8xGE IM
last trap: oper status:   (1) ok
last_spa_env_get_ok:      false
last_spa_env_read_time:   (0) 40455228 msecs ago
resync_reqd:              false
resync_count:             0

SPA physical index:       550
SPA container index:      5

SPA has no transceiver subblock
non-zero port indices:
port 0 has index 551
port 1 has index 552
port 2 has index 553
port 3 has index 554
port 4 has index 555
port 5 has index 556
```

```
port 6 has index 557
port 7 has index 558

non-zero SPA temp sensors:
sensor 0 has index 567
sensor 1 has index 568
sensor 2 has index 569
sensor 3 has index 570
sensor 4 has index 571

non-zero SPA volt sensors:
sensor 0 has index 579
sensor 1 has index 580
sensor 2 has index 581
sensor 3 has index 582
sensor 4 has index 583
sensor 5 has index 584
```