Using MIBs

This chapter describes how to perform tasks on the Cisco 1100 Series ISR

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://sso.cisco.com/autho/forms/MCOlogin.html

  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.

Managing Physical Entities

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

Purpose and Benefits

The physical entity management feature of the Cisco 1100 Series ISR 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.

Sample of ENTITY-MIB entPhysicalTable Entries

The samples in this section show how information is stored in the entPhysicalTable. You can perform asset inventory by examining entPhysicalTable entries.

Note The sample outputs and values that appear throughout this chapter are examples of data you can view when using MIBs.

The following output shows the ENTITY-MIB entPhysicalTable sample entries for RP card.

ENTITY-MIB entPhysicalTable Entries

```
entPhysicalDescr.7000 = Cisco ISR1100 Route Processor
entPhysicalDescr.7001 = Temp: Inlet 1
entPhysicalDescr.7002 = Temp: Inlet 2
entPhysicalDescr.7003 = Temp: Outlet 1
entPhysicalDescr.7004 = Temp: Outlet 2
entPhysicalDescr.7005 = Temp: core-A
entPhysicalDescr.7006 = Temp: core-B
entPhysicalDescr.7007 = Temp: core-C
entPhysicalDescr.7008 = V: 12v
entPhysicalDescr.7009 = V: 5v
entPhysicalDescr.7010 = V: 3.3v
entPhysicalDescr.7011 = V: 3.0v
entPhysicalDescr.7012 = V: 2.5v
entPhysicalDescr.7013 = V: 1.05v
entPhysicalDescr.7014 = V: 1.8v
entPhysicalDescr.7015 = V: 1.2v
entPhysicalDescr.7016 = V: Vcore-C
entPhysicalDescr.7017 = V: 1.1v
entPhysicalDescr.7018 = V: 1.0v
entPhysicalDescr.7019 = V: 1.8v-A
```

```
entPhysicalDescr.7020 = V: 1.5v-A
entPhysicalDescr.7021 = V: 1.5v-C1
entPhysicalDescr.7022 = V: 1.5v-B
entPhysicalDescr.7023 = V: Vcore-A
entPhysicalDescr.7024 = V: 1.5v-C2
entPhysicalDescr.7025 = V: Vcore-B1
entPhysicalDescr.7026 = V: Vcore-B2
entPhysicalDescr.7027 = V: 0.75v-B
entPhysicalDescr.7028 = V: 0.75v-C
entPhysicalDescr.7029 = I: 12v
entPhysicalDescr.7030 = P: pwr
entPhysicalDescr.7035 = CPU 0 of module R0
entPhysicalDescr.7036 = USB Port
entPhysicalDescr.7038 = USB Port
entPhysicalDescr.7040 = Network Management Ethernet
```

```
entPhysicalContainedIn.7000 = 1
entPhysicalContainedIn.7001 = 7000
entPhysicalContainedIn.7002 = 7000
entPhysicalContainedIn.7003 = 7000
entPhysicalContainedIn.7004 = 7000
entPhysicalContainedIn.7005 = 7000
entPhysicalContainedIn.7006 = 7000
entPhysicalContainedIn.7007 = 7000
entPhysicalContainedIn.7008 = 7000
entPhysicalContainedIn.7009 = 7000
entPhysicalContainedIn.7010 = 7000
entPhysicalContainedIn.7011 = 7000
entPhysicalContainedIn.7012 = 7000
entPhysicalContainedIn.7013 = 7000
entPhysicalContainedIn.7014 = 7000
entPhysicalContainedIn.7015 = 7000
entPhysicalContainedIn.7016 = 7000
entPhysicalContainedIn.7017 = 7000
entPhysicalContainedIn.7018 = 7000
entPhysicalContainedIn.7019 = 7000
entPhysicalContainedIn.7020 = 7000
entPhysicalContainedIn.7021 = 7000
entPhysicalContainedIn.7022 = 7000
entPhysicalContainedIn.7023 = 7000
entPhysicalContainedIn.7024 = 7000
entPhysicalContainedIn.7025 = 7000
entPhysicalContainedIn.7026 = 7000
entPhysicalContainedIn.7027 = 7000
entPhysicalContainedIn.7028 = 7000
entPhysicalContainedIn.7029 = 7000
entPhysicalContainedIn.7030 = 7000
entPhysicalContainedIn.7035 = 7000
entPhysicalContainedIn.7036 = 7000
entPhysicalContainedIn.7038 = 7000
entPhysicalContainedIn.7040 = 7000
```

where **entPhysicalContainedIn** indicates the entPhysicalIndex of a component's parent entity.

```
entPhysicalClass.7000 = module(9)
entPhysicalClass.7001 = sensor(8)
entPhysicalClass.7002 = sensor(8)
entPhysicalClass.7003 = sensor(8)
entPhysicalClass.7004 = sensor(8)
```

```
entPhysicalClass.7005 = sensor(8)
entPhysicalClass.7006 = sensor(8)
entPhysicalClass.7007 = sensor(8)
entPhysicalClass.7008 = sensor(8)
entPhysicalClass.7009 = sensor(8)
entPhysicalClass.7010 = sensor(8)
entPhysicalClass.7011 = sensor(8)
entPhysicalClass.7012 = sensor(8)
entPhysicalClass.7013 = sensor(8)
entPhysicalClass.7014 = sensor(8)
entPhysicalClass.7015 = sensor(8)
entPhysicalClass.7016 = sensor(8)
entPhysicalClass.7017 = sensor(8)
entPhysicalClass.7018 = sensor(8)
entPhysicalClass.7019 = sensor(8)
entPhysicalClass.7020 = sensor(8)
entPhysicalClass.7021 = sensor(8)
entPhysicalClass.7022 = sensor(8)
entPhysicalClass.7023 = sensor(8)
entPhysicalClass.7024 = sensor(8)
entPhysicalClass.7025 = sensor(8)
entPhysicalClass.7026 = sensor(8)
entPhysicalClass.7027 = sensor(8)
entPhysicalClass.7028 = sensor(8)
entPhysicalClass.7029 = sensor(8)
entPhysicalClass.7030 = sensor(8)
entPhysicalClass.7035 = cpu(12)
entPhysicalClass.7036 = port(10)
entPhysicalClass.7038 = port(10)
entPhysicalClass.7040 = port(10)
```

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

```
entPhysicalParentRelPos.7000 = 6
entPhysicalParentRelPos.7001 = 0
entPhysicalParentRelPos.7002 = 1
entPhysicalParentRelPos.7003 = 2
entPhysicalParentRelPos.7004 = 3
entPhysicalParentRelPos.7005 = 4
entPhysicalParentRelPos.7006 = 5
entPhysicalParentRelPos.7007 = 6
entPhysicalParentRelPos.7008 = 7
entPhysicalParentRelPos.7009 = 8
entPhysicalParentRelPos.7010 = 9
entPhysicalParentRelPos.7011 = 10
entPhysicalParentRelPos.7012 = 11
entPhysicalParentRelPos.7013 = 12
entPhysicalParentRelPos.7014 = 13
entPhysicalParentRelPos.7015 = 14
entPhysicalParentRelPos.7016 = 15
entPhysicalParentRelPos.7017 = 16
entPhysicalParentRelPos.7018 = 17
entPhysicalParentRelPos.7019 = 18
entPhysicalParentRelPos.7020 = 19
entPhysicalParentRelPos.7021 = 20
entPhysicalParentRelPos.7022 = 21
entPhysicalParentRelPos.7023 = 22
entPhysicalParentRelPos.7024 = 23
entPhysicalParentRelPos.7025 = 24
entPhysicalParentRelPos.7026 = 25
entPhysicalParentRelPos.7027 = 26
entPhysicalParentRelPos.7028 = 27
entPhysicalParentRelPos.7029 = 28
```

```
entPhysicalParentRelPos.7030 = 29
entPhysicalParentRelPos.7035 = 0
entPhysicalParentRelPos.7036 = 0
entPhysicalParentRelPos.7038 = 1
entPhysicalParentRelPos.7040 = 2
```

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

```
entPhysicalName.7000 = module R0
entPhysicalName.7001 = Temp: Inlet 1 R0/0
entPhysicalName.7002 = Temp: Inlet 2 R0/1
entPhysicalName.7003 = Temp: Outlet 1 R0/2
entPhysicalName.7004 = Temp: Outlet 2 R0/3
entPhysicalName.7005 = Temp: core-A R0/4
entPhysicalName.7006 = Temp: core-B R0/5
entPhysicalName.7007 = Temp: core-C R0/6
entPhysicalName.7008 = V: 12v R0/7
entPhysicalName.7009 = V: 5v R0/8
entPhysicalName.7010 = V: 3.3v R0/9
entPhysicalName.7011 = V: 3.0v R0/10
entPhysicalName.7012 = V: 2.5v R0/11
entPhysicalName.7013 = V: 1.05v R0/12
entPhysicalName.7014 = V: 1.8v R0/13
entPhysicalName.7015 = V: 1.2v R0/14
entPhysicalName.7016 = V: Vcore-C R0/15
entPhysicalName.7017 = V: 1.1v R0/16
entPhysicalName.7018 = V: 1.0v R0/17
entPhysicalName.7019 = V: 1.8v-A R0/18
entPhysicalName.7020 = V: 1.5v-A R0/19
entPhysicalName.7021 = V: 1.5v-C1 R0/20
entPhysicalName.7022 = V: 1.5v-B R0/21
entPhysicalName.7023 = V: Vcore-A R0/22
entPhysicalName.7024 = V: 1.5v-C2 R0/23
entPhysicalName.7025 = V: Vcore-B1 R0/24
entPhysicalName.7026 = V: Vcore-B2 R0/25
entPhysicalName.7027 = V: 0.75v-B R0/26
entPhysicalName.7028 = V: 0.75v-C R0/27
entPhysicalName.7029 = I: 12v R0/28
entPhysicalName.7030 = P: pwr R0/29
entPhysicalName.7035 = cpu R0/0
entPhysicalName.7036 = usb R0/0
entPhysicalName.7038 = usb R0/1
entPhysicalName.7040 = NME R0
```

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

```
entPhysicalHardwareRev.7000 = V01
entPhysicalHardwareRev.7001 =
entPhysicalHardwareRev.7002 =
entPhysicalHardwareRev.7003 =
entPhysicalHardwareRev.7004 =
entPhysicalHardwareRev.7005 =
entPhysicalHardwareRev.7006 =
entPhysicalHardwareRev.7007 =
entPhysicalHardwareRev.7008 =
entPhysicalHardwareRev.7009 =
entPhysicalHardwareRev.7010 =
entPhysicalHardwareRev.7011 =
entPhysicalHardwareRev.7012 =
```

```
entPhysicalHardwareRev.7013 =
entPhysicalHardwareRev.7014 =
entPhysicalHardwareRev.7015 =
entPhysicalHardwareRev.7016 =
entPhysicalHardwareRev.7017 =
entPhysicalHardwareRev.7018 =
entPhysicalHardwareRev.7019 =
entPhysicalHardwareRev.7020 =
entPhysicalHardwareRev.7021 =
entPhysicalHardwareRev.7022 =
entPhysicalHardwareRev.7023 =
entPhysicalHardwareRev.7024 =
entPhysicalHardwareRev.7025 =
entPhysicalHardwareRev.7026 =
entPhysicalHardwareRev.7027 =
entPhysicalHardwareRev.7028 =
entPhysicalHardwareRev.7029 =
entPhysicalHardwareRev.7030 =
entPhysicalHardwareRev.7035 =
entPhysicalHardwareRev.7036 =
entPhysicalHardwareRev.7038 =
entPhysicalHardwareRev.7040 =
```

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

```
entPhysicalSerialNum.7000 = FOC16150HB1
entPhysicalSerialNum.7001 =
entPhysicalSerialNum.7002 =
entPhysicalSerialNum.7003 =
entPhysicalSerialNum.7004 =
entPhysicalSerialNum.7005 =
entPhysicalSerialNum.7006 =
entPhysicalSerialNum.7007 =
entPhysicalSerialNum.7008 =
entPhysicalSerialNum.7009 =
entPhysicalSerialNum.7010 =
entPhysicalSerialNum.7011 =
entPhysicalSerialNum.7012 =
entPhysicalSerialNum.7013 =
entPhysicalSerialNum.7014 =
entPhysicalSerialNum.7015 =
entPhysicalSerialNum.7016 =
entPhysicalSerialNum.7017 =
entPhysicalSerialNum.7018 =
entPhysicalSerialNum.7019 =
entPhysicalSerialNum.7020 =
entPhysicalSerialNum.7021 =
entPhysicalSerialNum.7022 =
entPhysicalSerialNum.7023 =
entPhysicalSerialNum.7024 =
entPhysicalSerialNum.7025 =
entPhysicalSerialNum.7026 =
entPhysicalSerialNum.7027 =
entPhysicalSerialNum.7028 =
entPhysicalSerialNum.7029 =
entPhysicalSerialNum.7030 =
entPhysicalSerialNum.7035 =
entPhysicalSerialNum.7036 =
entPhysicalSerialNum.7038 =
entPhysicalSerialNum.7040 =
```

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

```
entPhysicalMfgName.7000 = Cisco Systems Inc
entPhysicalMfgName.7001 =
entPhysicalMfgName.7002 =
entPhysicalMfgName.7003 =
entPhysicalMfgName.7004 =
entPhysicalMfgName.7005 =
entPhysicalMfgName.7006 =
entPhysicalMfgName.7007 =
entPhysicalMfgName.7008 =
entPhysicalMfgName.7009 =
entPhysicalMfgName.7010 =
entPhysicalMfgName.7011 =
entPhysicalMfgName.7012 =
entPhysicalMfgName.7013 =
entPhysicalMfgName.7014 =
entPhysicalMfgName.7015 =
entPhysicalMfgName.7016 =
entPhysicalMfgName.7017 =
entPhysicalMfgName.7018 =
entPhysicalMfgName.7019 =
entPhysicalMfgName.7020 =
entPhysicalMfgName.7021 =
entPhysicalMfgName.7022 =
entPhysicalMfgName.7023 =
entPhysicalMfgName.7024 =
entPhysicalMfgName.7025 =
entPhysicalMfgName.7026 =
entPhysicalMfgName.7027 =
entPhysicalMfgName.7028 =
entPhysicalMfgName.7029 =
entPhysicalMfgName.7030 =
entPhysicalMfgName.7035 =
entPhysicalMfgName.7036 =
entPhysicalMfgName.7038 =
entPhysicalMfgName.7040 =
```

where **entPhysicalMfgName** provides the manufacturer's name for the physical component.

```
entPhysicalModelName.7000 = ISR4451/K9
entPhysicalModelName.7001 =
entPhysicalModelName.7002 =
entPhysicalModelName.7003 =
entPhysicalModelName.7004 =
entPhysicalModelName.7005 =
entPhysicalModelName.7006 =
entPhysicalModelName.7007 =
entPhysicalModelName.7008 =
entPhysicalModelName.7009 =
entPhysicalModelName.7010 =
entPhysicalModelName.7011 =
entPhysicalModelName.7012 =
entPhysicalModelName.7013 =
entPhysicalModelName.7014 =
entPhysicalModelName.7015 =
entPhysicalModelName.7016 =
entPhysicalModelName.7017 =
entPhysicalModelName.7018 =
entPhysicalModelName.7019 =
entPhysicalModelName.7020 =
entPhysicalModelName.7021 =
```

```
entPhysicalModelName.7022 =
entPhysicalModelName.7023 =
entPhysicalModelName.7024 =
entPhysicalModelName.7025 =
entPhysicalModelName.7026 =
entPhysicalModelName.7027 =
entPhysicalModelName.7028 =
entPhysicalModelName.7029 =
entPhysicalModelName.7030 =
entPhysicalModelName.7035 =
entPhysicalModelName.7036 =
entPhysicalModelName.7038 =
entPhysicalModelName.7040 =
```

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

```
entPhysicalIsFRU.7000 = false(2)
entPhysicalIsFRU.7001 = false(2)
entPhysicalIsFRU.7002 = false(2)
entPhysicalIsFRU.7003 = false(2)
entPhysicalIsFRU.7004 = false(2)
entPhysicalIsFRU.7005 = false(2)
entPhysicalIsFRU.7006 = false(2)
entPhysicalIsFRU.7007 = false(2)
entPhysicalIsFRU.7008 = false(2)
entPhysicalIsFRU.7009 = false(2)
entPhysicalIsFRU.7010 = false(2)
entPhysicalIsFRU.7011 = false(2)
entPhysicalIsFRU.7012 = false(2)
entPhysicalIsFRU.7013 = false(2)
entPhysicalIsFRU.7014 = false(2)
entPhysicalIsFRU.7015 = false(2)
entPhysicalIsFRU.7016 = false(2)
entPhysicalIsFRU.7017 = false(2)
entPhysicalIsFRU.7018 = false(2)
entPhysicalIsFRU.7019 = false(2)
entPhysicalIsFRU.7020 = false(2)
entPhysicalIsFRU.7021 = false(2)
entPhysicalIsFRU.7022 = false(2)
entPhysicalIsFRU.7023 = false(2)
entPhysicalIsFRU.7024 = false(2)
entPhysicalIsFRU.7025 = false(2)
entPhysicalIsFRU.7026 = false(2)
entPhysicalIsFRU.7027 = false(2)
entPhysicalIsFRU.7028 = false(2)
entPhysicalIsFRU.7029 = false(2)
entPhysicalIsFRU.7030 = false(2)
entPhysicalIsFRU.7035 = false(2)
entPhysicalIsFRU.7036 = false(2)
entPhysicalIsFRU.7038 = false(2)
entPhysicalIsFRU.7040 = false(2)
```

where **entPhysicalIsFRU** indicates whether or not this physical entity is considered a field replaceable unit (FRU).

Note the following about the sample configuration:

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

```
entPhysicalDescr.7000 = Cisco ISR1100 Route Processor
entPhysicalDescr.1 = Cisco C1117-4P Chassis
entPhysicalDescr.2 = Power Supply Bay
entPhysicalDescr.3 = External Power Supply Module
entPhysicalDescr.13 = Power Supply
entPhysicalDescr.14 = Fan
entPhysicalDescr.22 = POE Bay
entPhysicalDescr.42 = Internal POE Bay
entPhysicalDescr.1000 = Cisco C1117-4P Built-In NIM controller
entPhysicalDescr.1015 = Front Panel 1 port Gigabitethernet Module
entPhysicalDescr.1016 = C1117-1x1GE
entPhysicalDescr.1090 = subslot 0/0 transceiver container 0
entPhysicalDescr.1245 = C1117-ES-4
entPhysicalDescr.1246 = C1117-ES-4
entPhysicalDescr.1247 = C1117-ES-4
entPhysicalDescr.1248 = C1117-ES-4
entPhysicalDescr.1249 = C1117-ES-4
entPhysicalDescr.7000 = Cisco C1117-4P Route Processor
entPhysicalDescr.7001 = Temp: Int1
entPhysicalDescr.7002 = Temp: Int2
entPhysicalDescr.7003 = Temp: Int3
entPhysicalDescr.7004 = Temp: Int4
entPhysicalDescr.7005 = Temp: CPU
entPhysicalDescr.7006 = Temp: Wifi
entPhysicalDescr.7035 = CPU 0 of module R0
entPhysicalDescr.7036 = USB Port
entPhysicalDescr.7037 = USB Flash
entPhysicalDescr.7038 = Network Management Ethernet
entPhysicalDescr.9000 = Cisco C1117-4P Forwarding Processor
entPhysicalDescr.9001 = QFP 0 of module F0
entPhysicalContainedIn.1 = 0
entPhysicalContainedIn.2 = 1
entPhysicalContainedIn.3 = 2
entPhysicalContainedIn.13 = 3
entPhysicalContainedIn.14 = 3
entPhysicalContainedIn.22 = 1
entPhysicalContainedIn.42 = 1
entPhysicalContainedIn.1000 = 1
entPhysicalContainedIn.1015 = 1000
entPhysicalContainedIn.1016 = 1015
entPhysicalContainedIn.1090 = 1015
entPhysicalContainedIn.1245 = 1000
entPhysicalContainedIn.1246 = 1245
entPhysicalContainedIn.1247 = 1245
entPhysicalContainedIn.1248 = 1245
entPhysicalContainedIn.1249 = 1245
entPhysicalContainedIn.7000 = 1
entPhysicalContainedIn.7001 = 7000
entPhysicalContainedIn.7002 = 7000
entPhysicalContainedIn.7003 = 7000
entPhysicalContainedIn.7004 = 7000
entPhysicalContainedIn.7005 = 7000
entPhysicalContainedIn.7006 = 7000
entPhysicalContainedIn.7035 = 7000
entPhysicalContainedIn.7036 = 7000
entPhysicalContainedIn.7037 = 7036
entPhysicalContainedIn.7038 = 7000
entPhysicalContainedIn.9000 = 1
entPhysicalContainedIn.9001 = 9000
After the line: where entPhysicalContainedIn indicates the entPhysicalIndex of a component's parent entity.
entPhysicalClass.1 = chassis(3)
entPhysicalClass.2 = container(5)
entPhysicalClass.3 = powerSupply(6)
entPhysicalClass.13 = powerSupply(6)
```

```
entPhysicalClass.14 = fan(7)
entPhysicalClass.22 = container(5)
entPhysicalClass.42 = container(5)
entPhysicalClass.1000 = module(9)
entPhysicalClass.1015 = module(9)
entPhysicalClass.1016 = port(10)
entPhysicalClass.1090 = container(5)
entPhysicalClass.1245 = module(9)
entPhysicalClass.1246 = port(10)
entPhysicalClass.1247 = port(10)
entPhysicalClass.1248 = port(10)
entPhysicalClass.1249 = port(10)
entPhysicalClass.7000 = module(9)
entPhysicalClass.7001 = sensor(8)
entPhysicalClass.7002 = sensor(8)
entPhysicalClass.7003 = sensor(8)
entPhysicalClass.7004 = sensor(8)
entPhysicalClass.7005 = sensor(8)
entPhysicalClass.7006 = sensor(8)
entPhysicalClass.7035 = 12
entPhysicalClass.7036 = container(5)
entPhysicalClass.7037 = module(9)
entPhysicalClass.7038 = port(10)
entPhysicalClass.9000 = module(9)
entPhysicalClass.9001 = 12
After the line: where entPhysicalClass indicates.....
entPhysicalParentRelPos.1 = -1
entPhysicalParentRelPos.2 = 9
entPhysicalParentRelPos.3 = 0
entPhysicalParentRelPos.13 = 0
entPhysicalParentRelPos.14 = 0
entPhysicalParentRelPos.22 = 10
entPhysicalParentRelPos.42 = 11
entPhysicalParentRelPos.1000 = 0
entPhysicalParentRelPos.1015 = 0
entPhysicalParentRelPos.1016 = 0
entPhysicalParentRelPos.1090 = 0
entPhysicalParentRelPos.1245 = 1
entPhysicalParentRelPos.1246 = 0
entPhysicalParentRelPos.1247 = 1
entPhysicalParentRelPos.1248 = 2
entPhysicalParentRelPos.1249 = 3
entPhysicalParentRelPos.7000 = 6
entPhysicalParentRelPos.7001 = 0
entPhysicalParentRelPos.7002 = 1
entPhysicalParentRelPos.7003 = 2
entPhysicalParentRelPos.7004 = 3
entPhysicalParentRelPos.7005 = 4
entPhysicalParentRelPos.7006 = 5
entPhysicalParentRelPos.7035 = 0
entPhysicalParentRelPos.7036 = 0
entPhysicalParentRelPos.7037 = 0
entPhysicalParentRelPos.7038 = 1
entPhysicalParentRelPos.9000 = 8
entPhysicalParentRelPos.9001 = 0
After this line: where entPhysicalParentRelPos indicates the....
entPhysicalName.1 = Chassis
entPhysicalName.2 = Power Supply Bay 0
entPhysicalName.3 = Power Supply Module 0
entPhysicalName.13 = Power Supply 0
entPhysicalName.14 = Fan 0/0
entPhysicalName.22 = POE Bay 0
entPhysicalName.42 = Internal POE Bay 0
entPhysicalName.1000 = module 0
```

```
entPhysicalName.1015 = NIM subslot 0/0
entPhysicalName.1016 = GigabitEthernet0/0/0
entPhysicalName.1090 = subslot 0/0 transceiver container 0
entPhysicalName.1245 = NIM subslot 0/1
entPhysicalName.1246 = GigabitEthernet0/1/0
entPhysicalName.1247 = GigabitEthernet0/1/1
entPhysicalName.1248 = GigabitEthernet0/1/2
entPhysicalName.1249 = GigabitEthernet0/1/3
entPhysicalName.7000 = module R0
entPhysicalName.7001 = Temp: Int1 R0/0
entPhysicalName.7002 = Temp: Int2 R0/1
entPhysicalName.7003 = Temp: Int3 R0/2
entPhysicalName.7004 = Temp: Int4 R0/3
entPhysicalName.7005 = Temp: CPU R0/4
entPhysicalName.7006 = Temp: Wifi R0/5
entPhysicalName.7035 = cpu R0/0
entPhysicalName.7036 = usb R0/0
entPhysicalName.7037 = usb0
entPhysicalName.7038 = NME R0
entPhysicalName.9000 = module F0
entPhysicalName.9001 = qfp F0/0
After this line : where entPhysicalName provides the ....
entPhysicalHardwareRev.1 = V01
entPhysicalHardwareRev.2 =
entPhysicalHardwareRev.3 =
entPhysicalHardwareRev.13 =
entPhysicalHardwareRev.14 =
entPhysicalHardwareRev.22 =
entPhysicalHardwareRev.42 =
entPhysicalHardwareRev.1000 =
entPhysicalHardwareRev.1015 = V01
entPhysicalHardwareRev.1016 =
entPhysicalHardwareRev.1090 =
entPhysicalHardwareRev.1245 = V01
entPhysicalHardwareRev.1246 =
entPhysicalHardwareRev.1247 =
entPhysicalHardwareRev.1248 =
entPhysicalHardwareRev.1249 =
entPhysicalHardwareRev.7000 = V01
entPhysicalHardwareRev.7001 =
entPhysicalHardwareRev.7002 =
entPhysicalHardwareRev.7003 =
entPhysicalHardwareRev.7004 =
entPhysicalHardwareRev.7005 =
entPhysicalHardwareRev.7006 =
entPhysicalHardwareRev.7035 =
entPhysicalHardwareRev.7036 =
entPhysicalHardwareRev.7037 =
entPhysicalHardwareRev.7038 =
entPhysicalHardwareRev.9000 =
entPhysicalHardwareRev.9001 =
After this line : where entPhysicalHardwareRev provides the...
entPhysicalSerialNum.1 = FGL203820ED
entPhysicalSerialNum.2 =
entPhysicalSerialNum.3 =
entPhysicalSerialNum.13 =
entPhysicalSerialNum.14 =
entPhysicalSerialNum.22 =
entPhysicalSerialNum.42 =
entPhysicalSerialNum.1000 =
entPhysicalSerialNum.1015 =
entPhysicalSerialNum.1016 =
entPhysicalSerialNum.1090 =
entPhysicalSerialNum.1245 =
```

```
entPhysicalSerialNum.1246 =
entPhysicalSerialNum.1247 =
entPhysicalSerialNum.1248 =
entPhysicalSerialNum.1249 =
entPhysicalSerialNum.7000 = FOC20341XGM
entPhysicalSerialNum.7001 =
entPhysicalSerialNum.7002 =
entPhysicalSerialNum.7003 =
entPhysicalSerialNum.7004 =
entPhysicalSerialNum.7005 =
entPhysicalSerialNum.7006 =
entPhysicalSerialNum.7035 =
entPhysicalSerialNum.7036 =
entPhysicalSerialNum.7037 =
entPhysicalSerialNum.7038 =
entPhysicalSerialNum.9000 =
entPhysicalSerialNum.9001 =
After this line : where entPhysicalSerialNum provides the ....
entPhysicalMfgName.1 = Cisco Systems Inc
entPhysicalMfgName.2 =
entPhysicalMfgName.3 = Cisco Systems Inc
entPhysicalMfgName.13 =
entPhysicalMfgName.14 =
entPhysicalMfgName.22 =
entPhysicalMfgName.42 =
entPhysicalMfgName.1000 = Cisco Systems Inc
entPhysicalMfgName.1015 = Cisco Systems Inc
entPhysicalMfgName.1016 =
entPhysicalMfgName.1090 =
entPhysicalMfgName.1245 = Cisco Systems Inc
entPhysicalMfgName.1246 =
entPhysicalMfgName.1247 =
entPhysicalMfgName.1248 =
entPhysicalMfgName.1249 =
entPhysicalMfgName.7000 = Cisco Systems Inc
entPhysicalMfgName.7001 =
entPhysicalMfgName.7002 =
entPhysicalMfgName.7003 =
entPhysicalMfgName.7004 =
entPhysicalMfgName.7005 =
entPhysicalMfgName.7006 =
entPhysicalMfgName.7035 =
entPhysicalMfgName.7036 =
entPhysicalMfgName.7037 =
entPhysicalMfgName.7038 =
entPhysicalMfgName.9000 = Cisco Systems Inc
entPhysicalMfgName.9001 = Cisco Systems Inc
After this line: where entPhysicalMfgName provides the....
entPhysicalModelName.1 = C1117-4P
entPhysicalModelName.2 =
entPhysicalModelName.3 = PWR-12V
entPhysicalModelName.13 =
entPhysicalModelName.14 =
entPhysicalModelName.22 =
entPhysicalModelName.42 =
entPhysicalModelName.1000 = C1117-4P
entPhysicalModelName.1015 = C1117-1x1GE
entPhysicalModelName.1016 =
entPhysicalModelName.1090 =
entPhysicalModelName.1245 = C1117-ES-4
entPhysicalModelName.1246 =
entPhysicalModelName.1247 =
entPhysicalModelName.1248 =
entPhysicalModelName.1249 =
```

```
entPhysicalModelName.7000 = C1117-4P
entPhysicalModelName.7001 =
entPhysicalModelName.7002 =
entPhysicalModelName.7003 =
entPhysicalModelName.7004 =
entPhysicalModelName.7005 =
entPhysicalModelName.7006 =
entPhysicalModelName.7035 =
entPhysicalModelName.7036 =
entPhysicalModelName.7037 =
entPhysicalModelName.7038 =
entPhysicalModelName.9000 = C1117-4P
entPhysicalModelName.9001 =
After this line : where entPhysicalModelName provides the .....
entPhysicalIsFRU.1 = true(1)
entPhysicalIsFRU.2 = false(2)
entPhysicalIsFRU.3 = true(1)
entPhysicalIsFRU.13 = false(2)
entPhysicalIsFRU.14 = false(2)
entPhysicalIsFRU.22 = false(2)
entPhysicalIsFRU.42 = false(2)
entPhysicalIsFRU.1000 = false(2)
entPhysicalIsFRU.1015 = false(2)
entPhysicalIsFRU.1016 = false(2)
entPhysicalIsFRU.1090 = false(2)
entPhysicalIsFRU.1245 = false(2)
entPhysicalIsFRU.1246 = false(2)
entPhysicalIsFRU.1247 = false(2)
entPhysicalIsFRU.1248 = false(2)
entPhysicalIsFRU.1249 = false(2)
entPhysicalIsFRU.7000 = false(2)
entPhysicalIsFRU.7001 = false(2)
entPhysicalIsFRU.7002 = false(2)
entPhysicalIsFRU.7003 = false(2)
entPhysicalIsFRU.7004 = false(2)
entPhysicalIsFRU.7005 = false(2)
entPhysicalIsFRU.7006 = false(2)
entPhysicalIsFRU.7035 = false(2)
entPhysicalIsFRU.7036 = false(2)
entPhysicalIsFRU.7037 = true(1)
entPhysicalIsFRU.7038 = false(2)
entPhysicalIsFRU.9000 = false(2)
entPhysicalIsFRU.9001 = false(2)
```

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

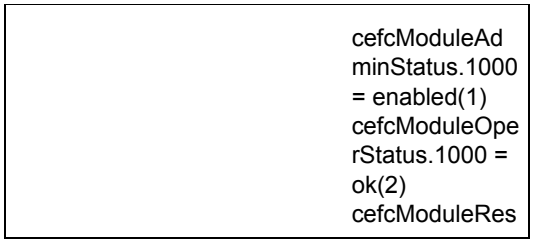
View objects in the CISCO-ENTITY-FRU-CONTROL-MIB cefcModuleTable to determine the administrative and operational status of FRUs, such as power supplies and line cards:

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

- cefcModuleOperStatus—The current operational state of the FRU.

Figure 5-1 shows a cefcModuleTable entry for a SIP card whose entPhysicalIndex is 1000.

Figure 5-1 Sample cefcModuleTable Entry



See the "FRU Status Changes" section on page 5-29 for information about how the router generates notifications to indicate changes in FRU status.

Using ENTITY-ALARM-MIB to Monitor Entity Alarms

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 "Entity Containment Tree" section for the entity hierarchy. The following sample output contains the information for the ISR 4451-X power supply in power supply bay 0:

```
blr-srtg-tftp:95> getmany -v2c 10.104.45.236 public entityMIB | grep "\.3 "
entPhysicalDescr.3 = External Power Supply Module
entPhysicalVendorType.3 = cevPowerSupply.583
entPhysicalContainedIn.3 = 2
entPhysicalClass.3 = powerSupply(6
entPhysicalParentRelPos.3 = 0
entPhysicalName.3 = Power Supply Module 0
entPhysicalHardwareRev.3 = V01
entPhysicalFirmwareRev.3 =
entPhysicalSoftwareRev.3 =
entPhysicalSerialNum.3 = JAB0929092D
entPhysicalMfgName.3 = Cisco Systems Inc
entPhysicalModelName.3 = PWR-12V
entPhysicalAlias.3 =
entPhysicalAssetID.3 =
entPhysicalIsFRU.3 = true(1)
entPhysicalEntry.17.3 = 00 00  00 00   00 00  00 00
entPhysicalEntry.18.3 = URN:CLEI:IPUIAFMRAA
entPhysicalChildIndex.2.3 = 3
```

For more information on this MIB, refer to ENTITY-MIB (RFC 4133)

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.

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 are the sample output:

```
blr-srtg-tftp:96> getmany -v2c 10.104.45.236 public ceAlarmDescrMapTable
ceAlarmDescrVendorType.1 = cevContainerSFP
ceAlarmDescrVendorType.2 = cevContainer.395
ceAlarmDescrVendorType.3 = cevContainer.396
ceAlarmDescrVendorType.4 = cevSensorModuleDeviceTemp
ceAlarmDescrVendorType.5 = cevSensorModuleDeviceVoltage
ceAlarmDescrVendorType.6 = cevSensorModuleDeviceCurrent
ceAlarmDescrVendorType.7 = cevSensor.133
ceAlarmDescrVendorType.8 = cevSensor.132
ceAlarmDescrVendorType.9 = cevSensor.134
ceAlarmDescrVendorType.10 = cevSensor
ceAlarmDescrVendorType.11 = cevModule.96.63
ceAlarmDescrVendorType.12 = cevContainer.333
ceAlarmDescrVendorType.13 = cevPortGe
ceAlarmDescrVendorType.14 = cevModule.96.64
ceAlarmDescrVendorType.15 = cevModule.96.65
ceAlarmDescrVendorType.16 = cevPowerSupply.583
ceAlarmDescrVendorType.17 = cevModule.96.37
ceAlarmDescrVendorType.18 = cevModule.96.56
ceAlarmDescrVendorType.19 = cevModule.96.38
ceAlarmDescrVendorType.20 = cevPortAdslAnnexA
```

The temperature sensor in ISR 1100 modules (RP) contains cevSensorModuleDeviceTemp as entPhysicalvendorType OID. From the above sample output, the index (ceAlarmDescrIndex) 5 is mapped to the RP sensor which has the cevSensorModuleDeviceTemp as the entPhysicalVendorType.

Note The generic vendor OID, cevSenor, is used in case the ISR 4451-X 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 type of entity in the Cisco 1100 Series ISR modules. The index 5 is obtained from the ceAlarmDescrMapTable in the previous section:

```
blr-srtg-tftp:97> getmany -v2c 10.104.45.236 public ceAlarmDescrTable | grep "\.4\."
ceAlarmDescrSeverity.4.0 = 1
ceAlarmDescrSeverity.4.1 = 1
ceAlarmDescrSeverity.4.2 = 1
ceAlarmDescrSeverity.4.3 = 2
ceAlarmDescrSeverity.4.4 = 3
ceAlarmDescrSeverity.4.5 = 1
ceAlarmDescrSeverity.4.6 = 1
ceAlarmDescrSeverity.4.7 = 2
```

```
ceAlarmDescrSeverity.4.8 = 3
ceAlarmDescrSeverity.4.9 = 1
ceAlarmDescrText.4.0 = Faulty Temperature Sensor
ceAlarmDescrText.4.1 = Temp Above Normal (Shutdown)
ceAlarmDescrText.4.2 = Temp Above Normal
ceAlarmDescrText.4.3 = Temp Above Normal
ceAlarmDescrText.4.4 = Temp Above Normal
ceAlarmDescrText.4.5 = Temp Below Normal (Shutdown)
ceAlarmDescrText.4.6 = Temp Below Normal
ceAlarmDescrText.4.7 = Temp Below Normal
ceAlarmDescrText.4.8 = Temp Below Normal
ceAlarmDescrText.4.9 = CHECK FOR OPEN SLOTS & BLOCKED AIR INTAKE
```

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 0 is for faulty 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.

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

octet 32:

```
    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
```

From the entity physical table (entPhysicalTable in ENTITY-MIB), we understnd that the Cisco 1100 Series ISR AC power supply in power supply bay 0 has 4 as entPhysicalIndex .

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

```
ciscouser-248->getone -v2c 9.0.0.56 public ceAlarmList.4
ceAlarmList.4 =
```

```
09 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
```

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 5, “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 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 5-1 Displaying Sample Output for the Alarm History

```
ciscouser-257->getnext -v2c 9.0.0.56 public ceAlarmHistory
ceAlarmHistTableSize.0 = 200 → the size of alarm history table
ptolemy-258->getnext -v2c 9.0.0.56 public ceAlarmHistTableSize.0
ceAlarmHistLastIndex.0 = 21 → the index for the last alarm added
```

Example 5-2 Displaying the Last Alarm Action (asserted or cleared) Added to the Alarm History Table

```
ptolemy-259->getmany -v2c 9.0.0.56 public ceAlarmHistTable | grep "\.21 "
ceAlarmHistType.21 = cleared(2)  →  alarm cleared
ceAlarmHistEntPhysicalIndex.21=4  →  it is for physical entity indexed by 4
ceAlarmHistAlarmType.21 = 3  →  alarm type is 3
ceAlarmHistSeverity.21 = major(2)  →  the alarm severity is major(2)
ceAlarmHistTimeStamp.21 = 7506193
```

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 5-3 Displaying the Physical Entity That has Value 13 as entPhysicalIndex

```
entPhysicalDescr.13 = Power Supply
entPhysicalVendorType.13 = cevPowerSupply.364
entPhysicalContainedIn.13 = 3
entPhysicalClass.13 = powerSupply(6)
entPhysicalParentRelPos.13 = 0
entPhysicalName.13 = Power Supply 0
entPhysicalHardwareRev.13 =
entPhysicalFirmwareRev.13 =
entPhysicalSoftwareRev.13 =
entPhysicalSerialNum.13 =
entPhysicalMfgName.13 =
entPhysicalModelName.13 =
entPhysicalAlias.13 = abcd
entPhysicalIsFRU.13 = false(2)
entPhysicalMfgDate.13 = 00 00 00 00 00 00 00 00
entPhysicalUris.13 =
```

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 5-4 Displaying the Sample Notification Received

```
Received SNMPv2c Trap:
Community: public
From: 9.0.0.56
sysUpTimeInstance = 7500792
snmpTrapOID.0 = ceAlarmCleared
ceAlarmHistEntPhysicalIndex.19 = 4
ceAlarmHistAlarmType.19 = 0
ceAlarmHistSeverity.19 = critical(1)
ceAlarmHistTimeStamp.19 = 7500792


Received SNMPv2c Trap:
Community: public
From: 9.0.0.56
sysUpTimeInstance = 7504592
snmpTrapOID.0 = ceAlarmAsserted
ceAlarmHistEntPhysicalIndex.20 = 4
ceAlarmHistAlarmType.20 = 3
ceAlarmHistSeverity.20 = major(2)
ceAlarmHistTimeStamp.20 = 7504592


Received SNMPv2c Trap:
Community: public
From: 9.0.0.56
sysUpTimeInstance = 7506193
snmpTrapOID.0 = ceAlarmCleared
ceAlarmHistEntPhysicalIndex.21 = 4
ceAlarmHistAlarmType.21 = 3
ceAlarmHistSeverity.21 = major(2)
ceAlarmHistTimeStamp.21 = 7506193
```

Entity Containment Tree

The following is sample entity hierarchy for a Cisco 1100 Series ISR, MIB Variables printed : <entPhysicalName entPhysicalClass>

ENTITY-MIB containment tree:

```
starwin:35> /users/tiswanso/bin/entity_hier.pl -h 10.104.45.235 -c public
/users/tiswanso/bin/entity_hier.pl -h 10.104.45.235 -c public

Storing Parent to child relationships
  {parentIdx}{class}{relPos} = childIdx
-----------------------------------------------
{1245}{port}{2} = 1248
{1}{container}{9} = 2
{0}{chassis}{-1} = 1
{7000}{sensor}{5} = 7006
{1245}{port}{0} = 1246
{1000}{module}{2} = 1475
{7000}{sensor}{4} = 7005
{1475}{module}{0} = 7007
{1245}{port}{3} = 1249
{7000}{container}{0} = 7036
{1245}{port}{7} = 1253
{1245}{port}{5} = 1251
{1}{module}{8} = 9000
{7000}{sensor}{3} = 7004
{1245}{port}{4} = 1250
{1475}{port}{1} = 1477
```

```
{1015}{container}{0} = 1090
{1}{module}{6} = 7000
{7000}{sensor}{2} = 7003
{7000}{12}{0} = 7035
{1245}{port}{1} = 1247
{1}{container}{11} = 42
{1}{container}{10} = 22
{9000}{12}{0} = 9001
{1475}{port}{0} = 1476
{3}{powerSupply}{0} = 13
{7000}{sensor}{0} = 7001
{1015}{container}{1} = 1100
{2}{powerSupply}{0} = 3
{1245}{port}{6} = 1252
{1000}{module}{0} = 1015
{7000}{port}{1} = 7038
{7000}{sensor}{1} = 7002
{1000}{module}{1} = 1245
{1}{module}{0} = 1000
{1015}{port}{1} = 1017
Entity Hierarchy Output Format:

   |

   <<entPhysicalClass>>

   |

   +-->[entPhysicalParentRelPos]   entPhysicalIndex : "entPhysicalName"

       |                                              "entPhysicalDescr"

       |                                              "entPhysicalVendorType"

       |

      ... hierarchy of children (entPhysicalContainedIn == entPhysicalIndex)




|

<<chassis>>

|

+-->[ -1]    1 : "Chassis"

    |            "Cisco C1111-8PLTEEA Chassis"

    |            "cevChassis.1859"

    |

    <<container>>

    |

    +-->[  9]    2 : "Power Supply Bay 0"

    |   |            "Power Supply Bay"
```

```
|   |            "cevContainer.395"
|   |
|   <<powerSupply>>
|   |
|   +-->[  0]    3 : "Power Supply Module 0"
|       |            "External Power Supply Module"
|       |            "cevPowerSupply.583"
|       |
|       <<powerSupply>>
|       |
|       +-->[  0]   13 : "Power Supply 0"
|                        "Power Supply"
|                        "cevPowerSupply.364"
|
+-->[ 10]   22 : "POE Bay 0"
|                "POE Bay"
|                "cevContainer.396"
|
+-->[ 11]   42 : "Internal POE Bay 0"
|                "Internal POE Bay"
|                "cevContainer.397"
|
<<module>>
|
+-->[  0]  1000 : "module 0"
|   |            "Cisco C1111-8PLTEEA Built-In NIM controller"
|   |            "cevModule.96.36"
|   |
|   <<module>>
|   |
|   +-->[  0]  1015 : "NIM subslot 0/0"
|   |   |            "Front Panel 2 port Gigabitethernet Module"
```

```
|   |   |            "cevModule.96.20"
|   |   |
|   |   <<container>>
|   |   |
|   |   +-->[  0]  1090 : "subslot 0/0 transceiver container 0"
|   |   |                 "subslot 0/0 transceiver container 0"
|   |   |                 "cevContainerSFP"
|   |   |
|   |   +-->[  1]  1100 : "subslot 0/0 transceiver container 1"
|   |   |                 "subslot 0/0 transceiver container 1"
|   |   |                 "cevContainerSFP"
|   |   |
|   |   <<port>>
|   |   |
|   |   +-->[  1]  1017 : "GigabitEthernet0/0/1"
|   |                     "C1111-2x1GE"
|   |                     "cevPortGe"
|   |
|   +-->[  1]  1245 : "NIM subslot 0/1"
|   |   |            "C1111-ES-8"
|   |   |            "cevModule.96.21"
|   |   |
|   |   <<port>>
|   |   |
|   |   +-->[  0]  1246 : "GigabitEthernet0/1/0"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  1]  1247 : "GigabitEthernet0/1/1"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
```

```
|   |   +-->[  2]  1248 : "GigabitEthernet0/1/2"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  3]  1249 : "GigabitEthernet0/1/3"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  4]  1250 : "GigabitEthernet0/1/4"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  5]  1251 : "GigabitEthernet0/1/5"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  6]  1252 : "GigabitEthernet0/1/6"
|   |   |                 "C1111-ES-8"
|   |   |                 "cevPortGe"
|   |   |
|   |   +-->[  7]  1253 : "GigabitEthernet0/1/7"
|   |                     "C1111-ES-8"
|   |                     "cevPortGe"
|   |
|   +-->[  2]  1475 : "NIM subslot 0/2"
|       |             "C1111-LTE Module"
|       |             "cevModule.96.22"
|       |
|       <<module>>
|       |
|       +-->[  0]  7007 : "Modem 0 on Cellular0/2/0"
|       |                 "Sierra Wireless EM7455/EM7430"
```

```
|       |                "cevModuleDaughterCard.88"
|       |
|       <<port>>
|       |
|       +-->[  0]  1476 : "Cellular0/2/0"
|       |                "LTE Adv CAT6 - Multimode LTE/DC-HSPA+/HSPA+/HSPA/UMTS/EDGE/GPRS"
|       |                "cevPortCBusSerial"
|       |
|       +-->[  1]  1477 : "Cellular0/2/1"
|                        "LTE Adv CAT6 - Multimode LTE/DC-HSPA+/HSPA+/HSPA/UMTS/EDGE/GPRS"
|                        "cevPortCBusSerial"
|
+-->[  6]  7000 : "module R0"
|   |            "Cisco C1111-8PLTEEA Route Processor"
|   |            "cevModule.96.34"
|   |
|   <<12>>
|   |
|   +-->[  0]  7035 : "cpu R0/0"
|   |                "CPU 0 of module R0"
|   |                "cevModuleCpuType"
|   |
|   <<container>>
|   |
|   +-->[  0]  7036 : "usb R0/0"
|   |                "USB Port"
|   |                "cevContainer.333"
|   |
|   <<port>>
|   |
|   +-->[  1]  7038 : "NME R0"
|   |                "Network Management Ethernet"
```

```
|   |                "cevPortGe"
|   |
|   <<sensor>>
|   |
|   +-->[  0]  7001 : "Temp: Int1 R0/0"
|   |                "Temp: Int1"
|   |                "cevSensorModuleDeviceTemp"
|   |
|   +-->[  1]  7002 : "Temp: Int2 R0/1"
|   |                "Temp: Int2"
|   |                "cevSensorModuleDeviceTemp"
|   |
|   +-->[  2]  7003 : "Temp: Int3 R0/2"
|   |                "Temp: Int3"
|   |                "cevSensorModuleDeviceTemp"
|   |
|   +-->[  3]  7004 : "Temp: Int4 R0/3"
|   |                "Temp: Int4"
|   |                "cevSensorModuleDeviceTemp"
|   |
|   +-->[  4]  7005 : "Temp: CPU R0/4"
|   |                "Temp: CPU"
|   |                "cevSensorModuleDeviceTemp"
|   |
|   +-->[  5]  7006 : "Temp: Wifi R0/5"
|                    "Temp: Wifi"
|                    "cevSensorModuleDeviceTemp"
|
+-->[  8]  9000 : "module F0"
    |             "Cisco C1111-8PLTEEA Forwarding Processor"
    |             "cevModule.96.35"
    |
    <<12>>
    |
    +-->[  0]  9001 : "qfp F0/0"
                      "QFP 0 of module F0"
                      "cevModuleCpuType"
```

```
-------------------------------------------
  Printing leftover entity relationships:
-------------------------------------------
```

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

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 "Enabling Notifications" section on page 4-2. 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 5-30 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 5-30.

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 and QoS data information to determine the amount to bill customers for traffic. It also includes a scenario for demonstrating that a QoS service policy attached to an interface is policing traffic on that interface.

This section contains the following topics:

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-MIBB (RFC 2863) section.

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

- Unless noted, all IF-MIB counters are supported on Cisco 1100 Series ISR interfaces.
- For IF-MIB high capacity counter support, Cisco conforms 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.
- When a QoS service policy is attached to an interface, the router applies the rules of the policy to traffic on the interface and increments the packet and bytes counts on the interface.

The following CISCO-CLASS-BASED-QOS-MIB objects provide interface counts:

- cbQosCMDropPkt and cbQosCMDropByte (cbQosCMStatsTable)—Total number of packets and bytes that were dropped because they exceeded the limits set by the service policy. These counts include only those packets and bytes that were dropped because they exceeded service policy limits. The counts do not include packets and bytes dropped for other reasons.
- cbQosPoliceConformedPkt and cbQosPoliceConformedByte (cbQosPoliceStatsTable)—Total number of packets and bytes that conformed to the limits of the service policy and were transmitted.

Determining the Amount of Traffic to Bill to a Customer

Perform these steps to determine how much traffic on an interface is billable to a particular customer:

Step 1 Determine which service policy on the interface applies to the customer.

Step 2 Determine the index values of the service policy and class map used to define the customer's traffic. You need this information in the following steps.

Step 3 Generate traffic with the traffic generator. The data rate should be more than that is configured for Conform burst(bc)/Exceed burst(be) for the policy.

Step 4 (Optional) Access the cbQosCMDropPkt object (cbQosCMStatsTable) for the customer to determine how much of the customer's traffic was dropped because it exceeded service policy limits.

Scenario for Demonstrating QoS Traffic Policing

This section describes a scenario that demonstrates the use of SNMP QoS statistics to determine how much traffic on an interface is billable to a particular customer. It also shows how packet counts are affected when a service policy is applied to traffic on the interface.

To create the scenario, follow these steps, each of which is described in the sections that follow:

1. Create and attach a service policy to an interface.
2. View packet counts before the service policy is applied to traffic on the interface.
3. Issue a **ping** command to generate traffic on the interface. Note that the service policy is applied to the traffic.
4. View packet counts after the service policy is applied to determine how much traffic to bill the customer for:
   - Conformed packets—The number of packets within the range set by the service policy and for which you can charge the customer.
   - Exceeded or dropped packets—The number of packets that were not transmitted because they were outside the range of the service policy. These packets are not billable to the customer.

Note In the above scenario, the Cisco 1100 Series ISR is used as an interim device (that is, traffic originates elsewhere and is destined for another device).

Service Policy Configuration

This scenario uses the following policy-map configuration. For information on how to create a policy map, see "Configuring Quality of Service" in the *QoS: Classification Configuration Guide, Cisco IOS XE Release 3.9S.*

```
Policy Map test-police
    Class class-default
     police cir 1000000 bc 10000 be 20000
       conform-action transmit
       exceed-action drop
       violate-action drop

interface GigabitEthernet1/1/5
 ip address 15.1.0.52 255.0.0.0
 no negotiation auto
 service-policy output test-police
end
```

Packet Counts Before the Service Policy Is Applied

The following CLI and SNMP output shows the interface's output traffic before the service policy is applied:

CLI Command Output

```
Router# sh policy-map interface gi 1/1/5

 GigabitEthernet1/1/5

  Service-policy output: test-police

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: any
      police:
          cir 1000000 bps, bc 10000 bytes, be 20000 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        violated 0 packets, 0 bytes; actions:
```

```
          drop
        conformed 0 bps, exceed 0 bps, violate 0 bps
```

SNMP Output

```
ciscouser:4> getmany 9.0.0.52 cbQosIfIndex
cbQosIfIndex.290 = 18
ciscouser:5> getone 9.0.0.52 ifDescr.18
ifDescr.18 = GigabitEthernet1/1/5
ciscouser:6>

getmany 9.0.0.52 cbQosCMDropPkt cbQosCMDropByte
cbQosCMDropPkt.290.9756705 = 0
cbQosCMDropByte.290.9756705 = 0
ciscouser:77>
```

Packet Counts After the Service Policy Is Applied

After you generate traffic using the traffic generator, look at the number of packets that exceeded and conformed to the committed information rate (CIR) set by the **police** command:

- 19351 packets conformed to the police rate and were transmitted
- 80 packets exceeded the police rate and were dropped
- 16066130 packets violated the police rate and were dropped

The following CLI and SNMP output show the counts on the interface after the service policy is applied. The object cbQosCMDropPkt refers to sum of exceeded and violated packets and cbQosCMDropByte refers to the sum of exceeded and violated bytes. (In the output, exceeded andviolated packet counts are shown in boldface.)

CLI Command Output

```
Router#sh show policy-map int gi 1/1/5

 GigabitEthernet1/1/5

  Service-policy output: test-police

    Class-map: class-default (match-any)
      16085561 packets, 1994609369 bytes

      5 minute offered rate 16051000 bps, drop rate 16032000 bps
      Match: any
      police:
          cir 1000000 bps, bc 10000 bytes, be 10000 bytes
        conformed 19351 packets, 2399329 bytes; actions:
          transmit
        exceeded 80 packets, 9920 bytes; actions:
          drop
        violated 16066130 packets, 1992200120 bytes; actions:
          drop
        conformed 0 bps, exceed 0 bps, violate 16032000 bps
Router#
```

SNMP Output

```
getmany 9.0.0.52 cbQosCMDropPkt cbQosCMDropByte
cbQosCMDropPkt.290.9756705 = 16066210
cbQosCMDropByte.290.9756705 = 1992210040
ptolemy:77>
```

. . .

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 5-1 lists capacity counter object identifiers (OIDs).

Table 5-1 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 MIB Tools from the following URL:
  http://tools.cisco.com/ITDIT/MIBS/servlet/index

Displaying the Module Hardware Type

To verify the SIP hardware type that is installed in your Cisco 1100 Series ISR, you can use the show platform command. The example below shows some list of such commands.

Example 5-5 Example of the show platform command

The following example shows the output of the **show platform** command on the Cisco 1100 Series ISR[1]:

```
Router#sh platform


Router#sh platform ?
  hardware  Show platform hardware information
  software  Show platform software information
  |         Output modifiers
  <cr>


Router#sh platform har
Router#sh platform hardware ?
backplaneswitch-manager  Backplane Switch Manager hardware
  crypto-device            crypto device information
  interface                Interface information
  network-clocks           Show network clock device
  port                     port information
  qfp                      Quantum Flow Processor
  raid                     raid information
  slot                     Slot information
  subslot                  Subslot information
  throughput               Show throughput commands


Router#sh platform hardware slot 0 ?
  dram            CPU DRAM commands
  eobc            Show EOBC
  fan             Fan commands
  i95             i95 driver statistics
  io-port         IO Port information
  led             LED-related commands
  mcu             MCU related commands
  network-clocks  Show network clock device
  pcie            PCIE-related commands
  plim            PLIM information
  rommon          Rommon commands
  sensor          Sensor information
  serdes          Serdes information
  spa             Module related information
```

1.