Sun™ Netra™ SNMP Management Agent 1.0 User's Reference Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 U.S.A.
650-960-1300

Part No. 806-5817-10
August 2000, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Contents

Figures

Tables

Preface

This manual is intended for engineers who are implementing SNMP management systems for Netra servers.

How This Book Is Organized

Chapter 1 provides a summary of the Sun Netra SNMP Management Agent 1.0 SNMP agent.

Chapter 2 provides an overview of how a Sun Netra product is modeled and structured using managed objects.

Chapter 3 describes the MIB interfaces used in the Sun Netra SNMP Management Agent 1.0 SNMP agent.

Chapter 4 describes the managed object class inheritance hierarchy and attributes.

Chapter 5 describes the SNMP traps that can be generated by the system.

Appendix A explains the SNMP features exploited by Sun Netra SNMP Management Agent 1.0.

Appendix B lists reference documents and standards.

Glossary is a list of words, acronyms, and phrases and their definitions.

Related Documentation

TABLE P-1

Application	Title	Part Number
Installation and configuration	*Sun Netra SNMP Management Agent 1.0 Installation and Configuration Guide*	806-5111-10
Platform-specific information	*Sun Netra SNMP Management Agent 1.0 Supplement for Netra t Servers*	806-5818-10
Netra t 1120/1125 servers	*Netra t 1120/1125 Installation and Basic Maintenance Guide*	805-6803-10
Netra t1 Model 100/105 servers	*LOMlite User's Guide*	806-2038-10
Netra t 1400/1405 servers	*Netra t 1400/1405 Installation and User's Guide* *LOMlite User's Guide*	806-0575-10 806-2038-10

Accessing Sun Documentation Online

The `docs.sun.com`[sm] web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at:

`http://www1.fatbrain.com/documentation/sun`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can email your comments to Sun at:

docfeedback@sun.com

Please include the part number (806-5817-10) of your document in the subject line of your email.

CHAPTER 1

Overview

Sun Netra SNMP Management Agent 1.0 is an SNMP agent for remotely managing the hardware of Sun Netra Products. Sun Netra SNMP Management Agent 1.0 combines the simplicity and performance of SNMP with the flexibility and scalability of the ITU-T TMN network management standards. Its powerful management interface is a key differentiator for supporting the rapid development of re-usable, automated, fault and policy management solutions. Lightweight, fast and highly-scalable, Sun Netra SNMP Management Agent 1.0 is an invaluable component for integrating Sun Netra Products into new or existing managed networks.

Key Features

- End user benefits:
  - Gives a responsive, highly-scalable remote monitoring capability for Sun Netra Products.
  - Improves availability through proactive problem detection and continuous system monitoring.
  - Allows online access to inventory data of hardware for part replacement.
- Developer benefits:
  - Uses a simple, generic set of building blocks to interface to the hardware infrastructure.
  - Allows developers to easily and quickly create high-value automated management services.
  - Provides highly-configurable event and alarm management facilities for distributed policy management.
- Ease of integration:
  - Allows Sun Netra Products to be easily managed by leading SNMP-compliant enterprise management frameworks (HP, CA, Tivoli, etc.).
  - Complements existing MIB II agents.

How It Works

Sun Netra SNMP Management Agent 1.0 is an intelligent SNMPv2 agent for continuous monitoring of key hardware variables. Proactive problem detection and automated alarming means that problems can be identified sooner and resolved faster, reducing potential system downtime. In addition, Sun Netra SNMP Management Agent 1.0 provides comprehensive access to hardware health, configuration and inventory data. These allow developers to generate and collect value-add reports which can be collected via remote monitoring.

Using Sun Netra SNMP Management Agent 1.0's generic management interface and its comprehensive event mechanisms, developers can dynamically build up configuration and health status data for any Sun Netra Product, hence reducing development cost and enabling rapid deployment of solutions and upgrades. Sun Netra SNMP Management Agent 1.0's unique generic interface and highly-accessible, standard internet protocols makes Sun Netra SNMP Management Agent 1.0 an essential component of a homogeneous, yet holistic network management solution.

CHAPTER 2

Management Interface

This chapter provides an overview of how the Sun Netra SNMP Management Agent 1.0 models a Sun Netra product, and how this model is strcutured. This is called the Sun Network Information Model (SunNIM).

Modeling a Sun Netra Product

SunNIM presents a Sun Netra Product as a collection of nested *hardware resources* within a chassis. Some resources may be nested directly within the chassis, such as a motherboard, or a network connector. Others may be nested within other resources, for example a motherboard may include a processor or a drive may hold a disk. These relationships extending from the chassis form a *hierarchy* of hardware resources. This hierarchy is modeled using *relationships* between *managed objects* representing the hardware resources.

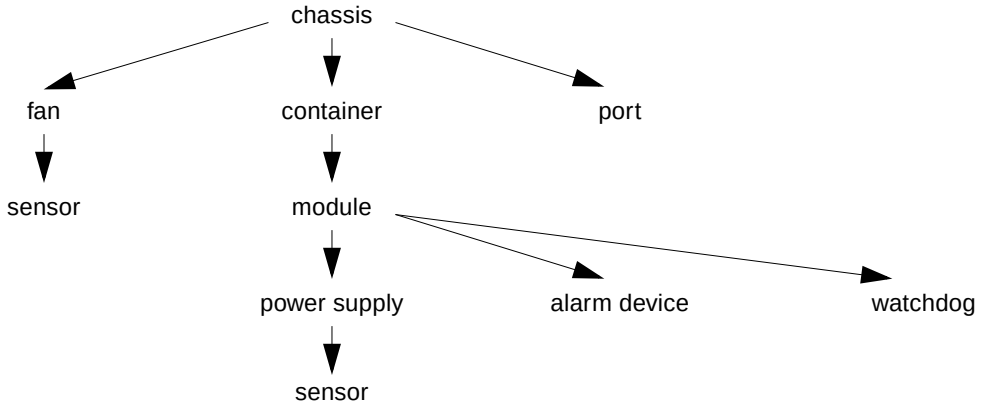

FIGURE 2-1 Hardware Resource Hierarchy

Managed Objects

The SunNIM provides a useful set of common platform building blocks representing fundamental hardware resources. These platform building blocks are called *managed objects*. A hardware resource will be represented by a managed object if it can be monitored or provides useful configuration information.

Additional managed objects are used to represent other features of the management interface; for example, hardware resources can issue asynchronous status reports, called *notifications*, in response to problems or changes in configuration.

Managed objects are defined in terms of managed object *classes*. Characteristics of the hardware resource are represented by *attributes* of the managed object. New classes, called *subclasses*, are defined in terms of existing classes. A subclass *inherits* all the characteristics of its *superclass*, but represents its own characteristics by adding new attributes and notifications.

FIGURE 2-2 shows the class names of the hardware building blocks defined by SunNIM.

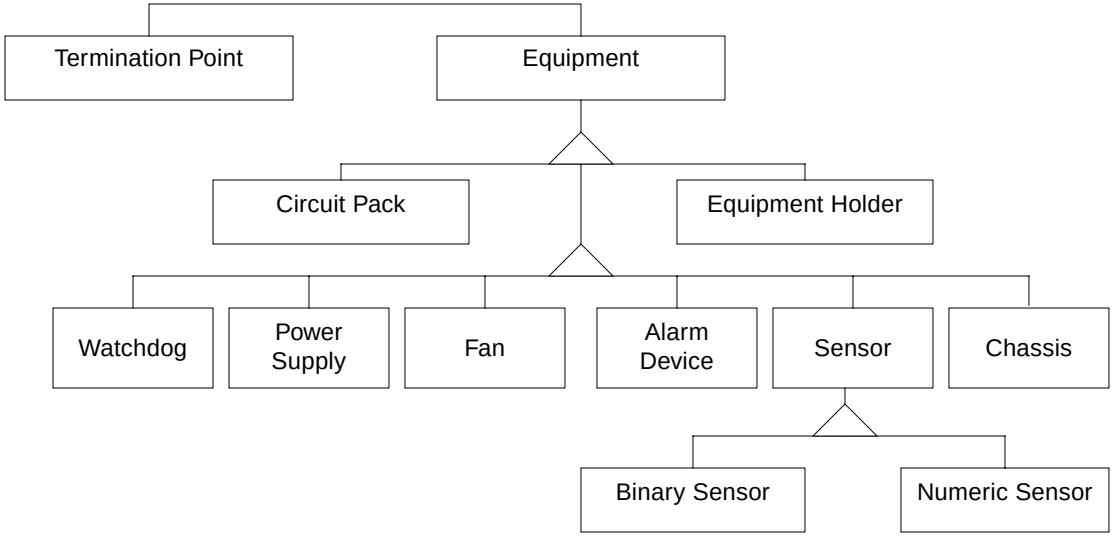

FIGURE 2-2 SunNIM Hardware Resource Class Inheritance Diagram

Derivation of SunNIM Classes

The SunNIM classes are based on industry-standard management concepts. The Sun Netra SNMP Management Agent 1.0 uses a subset of the ITU-T Generic Network Information Model (GNIM) chosen for its representation of hardware infrastructure. The GNIM provides a powerful and extendible framework which supports uniform fault and configuration management in a Telecommunications Management Network (TMN).

Sun Netra SNMP Management Agent 1.0 uses a subset of ITU-T GNIM chosen its representation of hardware infrastructure. FIGURE 2-3 shows the subset of the ITU-T GNIM classes presented by SunNIM.

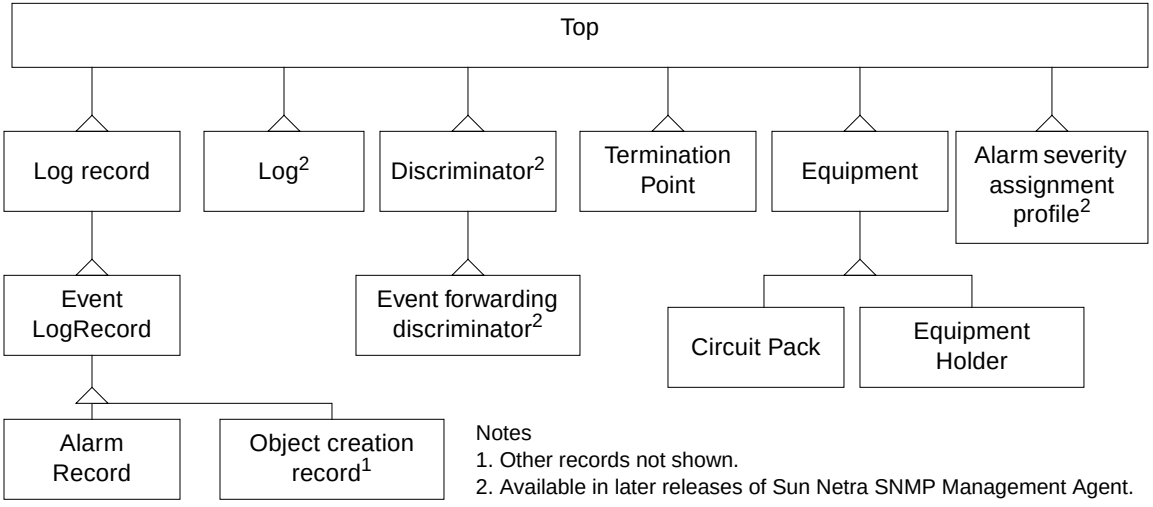

FIGURE 2-3 The Supported Subset of the ITU-T GNIM Classes

The ITU-T GNIM has been specialized through inheritance to provide managed object classes which represent the characteristics of different types of hardware resource. SunNIM extends the GNIM Equipment class using the Distributed Management Task Force Common Interface Model (DMTF CIMv2) concepts chosen for their system and fault monitoring capabilities. The DMTF CIMv2 provides generic attributes representing the different characteristics of hardware resources.

CHAPTER 3

The SunNIM MIBs

This chapter describes how the SunNIM managed objects and their relationships are presented by the SNMP interface

SNMP Representation of SunNIM

Sun Netra SNMP Management Agent 1.0 uses three SNMP MIBs to present the SunNIM:

- ENTITY-MIB (RFC 2737)
- SUN-NET-INFO-MIB (SunNim)
- SUN-NIM-EXT-MIB (SunNem)

Appendix A provides an overview of the generic SNMP capabilities used to represent the SunNIM.

The ENTITY-MIB

The ENTITY-MIB is defined by the IETF standard RFC2737. the ENTITY-MIB provides a mechanism for presenting hierarchies of physical entities using SNMP tables.

SunNIM uses the ENTITY-MIB to provide:

- A hierarchy of hardware resources—relationships between managed objects
- Common hardware resource characteristics—a mapping of common attributes from the GNIM Top, Equipment, and Termination Point classes.

This information is presented using three SNMP tables:

- Physical Entity Table (*entPhysicalTable*)

  This table provides a row per hardware resource. These rows are called *Entries* and a particular row is referred to as an *instance*. Each entry contains the physical class (*entPhysicalClass*) and common characteristics of the hardware resource. Each entry has a unique index (*entPhysicalIndex*) and contains a reference (*entPhysicalContainedIn*) which points to the row of the hardware resource which acts as the *container* for this resource.

- Physical Mapping Table (*entPhysicalContainsTable*)

  This table provides a virtual copy of the hierarchy of the hardware resources represented in the Physical Entity Table. This table is two-dimensional, indexed firstly by the entPhysicalIndex of the containing entry, and secondly by the entPhysicalIndexes of the contained entries.

- General Table (*entGeneralTable*)

  This table provides the time interval between the agent starting and the last change to the Physical Entity Table or Physical Mapping Table.

FIGURE 3-1, FIGURE 3-2 and FIGURE 3-3 show how an example hierarchy of hardware resources are presented using the ENTITY-MIB.

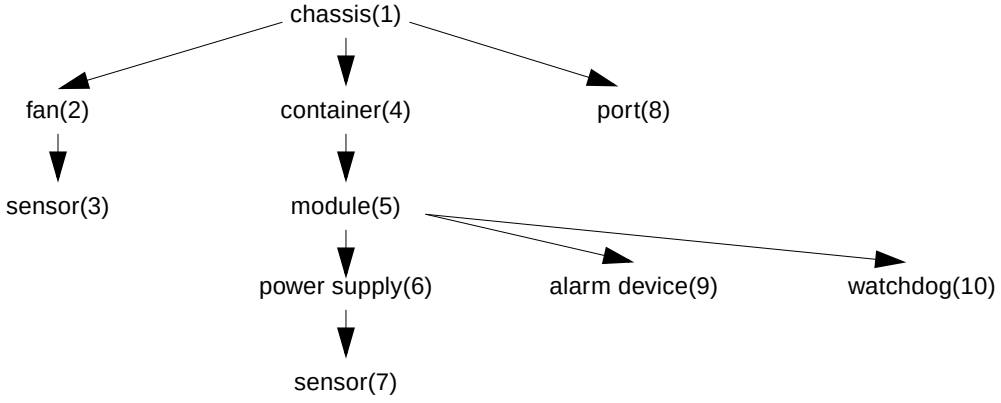

FIGURE 3-1 Hardware Resource Hierarchy

entPhysicalIndex	entPhysicalClass	entPhysicalContainedIn	. . .
1	chassis	0	. . .
2	fan	1	. . .
3	sensor	2	. . .
4	container	1	. . .
5	module	4	. . .
6	power supply	5	. . .
7	sensor	6	. . .
8	port	1	. . .
9	other	5	. . .
10	other	5	. . .

FIGURE 3-2 Physical Entity Table

entPhysicalIndex	entPhysicalChildIndex
1	2
1	4
1	8
2	3
4	5
5	6
5	9
5	10
6	7

FIGURE 3-3 Physical Mapping Table

The ENTITY-MIB also provides a trap containing the last change time stamp for the time that any Entity in the Physical Entity Table or Physical Mapping Table is changed.

The SUN-NET-INFO-MIB

The SUN-NET-INFO-MIB (SunNim) provides the additional attributes from the GNIM classes that are not represented in the Physical Entity Table. The SunNim MIB extends the Physical Entity Table by adding four sparsely populated Table Extensions:

- Equipment Table Extension (*sunNimEquipmentTable*)

  This extends the Physical Entity Table to provide further information for managed objects of the GNIM Equipment class. This class is applicable for all hardware resources with the exception of communication ports. Subclasses of the GNIM Equipment class are represented by further Table Extensions.

  - Equipment Holder Table Extension (*sunNimEquipmentHolderTable*)

    This augments the GNIM Equipment Table Extension. It provides the additional information relevant for managed objects of the GNIM Equipment Holder class.

  - Circuit Pack Table Extension (*sunNimCircuitPackTable*)

    This augments the GNIM Equipment Table Extension. It provides the additional information relevant for managed objects of the GNIM Circuit Pack class.

- Termination Point Table Extension (*sunNimTerminationPointTable*)

  This extends the Physical Entity Table to provide further information for managed objects of the GNIM Termination Point class.

FIGURE 3-4 shows example Table Extensions to the Physcial Entity Table.

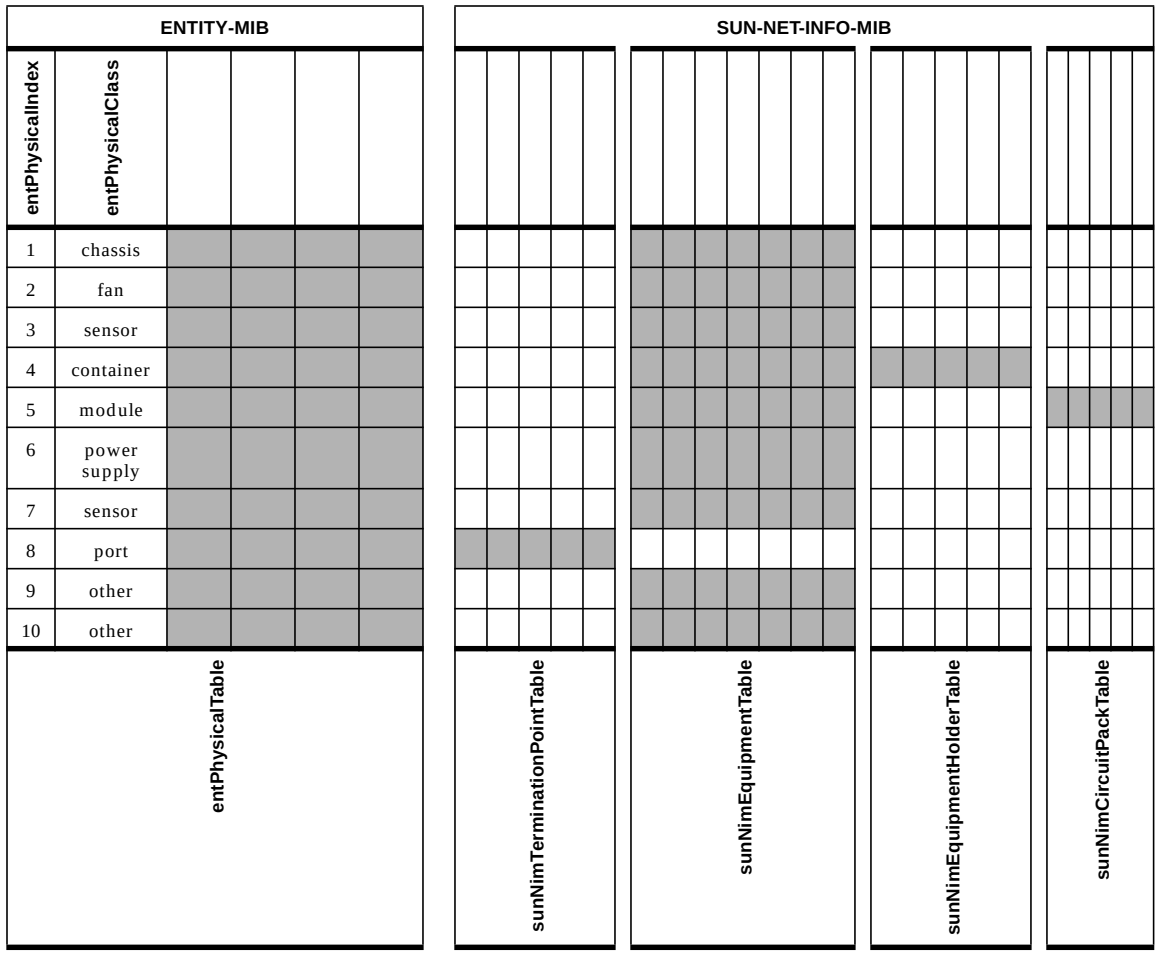

FIGURE 3-4 SunNim Table Extensions

The SunNim MIB also defines the event records for the GNIM notifications. These records form part of the payload of the SunNim SNMP trap notifications. The record data that accompanies an SNMP trap is as follows:

- Object Creation Record (*sunNimObjectCreation*)

  This indicates that a hardware resource has been added to the hierarchy.

- Object Deletion Record (*sunNimObjectDeletion*)

  This indicates that a hardware resource has been removed from the hierarchy.

- Communications Alarm Record (*sunNimCommunicationsAlarm*)

  This indicates that a failure has occurred in the communication services that the hardware resource supports.

- Environmental Alarm Record (*sunNimEnvironmentalAlarm*)

  This indicates an environmental condition relating to the hardware resource.

- Equipment Alarm Record (*sunNimEquipmentAlarm*)

  This indicates that a hardware resource has become faulty.

- Processing Error Alarm Record (*sunNimProcessingErrorAlarm*)

  This indicates that a hardware resource has an associated software or processing fault.

- State Change Record (*sunNimStateChange*)

  This indicates that the state of the hardware resource has changed.

- Integer Attribute Value Change Record (*sunNimAttributeChangeInteger*)

  This indicates a change in a characteristic of a hardware resource modeled by an attribute of type Integer.

- String Attribute Value Change Record (*sunNimAttributeChangeString*)

  This indicates a change in a characteristic of a hardware resource modeled by an attribute of type String.

The SUN-NIM-EXT-MIB

The SUN-NIM-EXT-MIB (SunNem) provides the attributes of the DMTF-derived subclasses of the GNIM Equipment class.

The SunNem further extends the Physical Entity Table by the addition of seven sparsely populated Table Extensions. Note that the Power Supply and Chassis subclasses do not extend the characteristics of the GNIM Equipment class, and as such do not have their own Table Extensions.

- Physical Table Extension (*sunNemPhysicalTable*)

  This extends the Physical Entity Table. It is used to supplement the *entPhysicalClass* column in the Physical Entity Table. If a hardware resource has an *entPhysicalClass* of 'other', but is of a class modeled by the SunNem, that is, the Watchdog or AlarmDevice class, this table will identify its SunNem class.

- Sensor Table Extension (*sunNemSensorTable*)

  This augments the Equipment Table Extension. It provides the additional information relevant for managed objects of the Sensor class. Subclasses of Sensor class are represented by further Table Extensions.

  - Binary Sensor Table Extension (*SunNemBinarySensorTable*)

    This augments the Sensor Table Extension. It provides additional information relevant for managed objects of the Binary Sensor class.

  - Numeric Sensor Table Extension (*sunNemNumericSensorTable*)

    This augments the Sensor Table Extension. It provides additional information relevant for managed objects of the Numeric Sensor class.

- Fan Table Extension (*sunNemFanTable*)

  This augments the Equipment Table Extension. It provides the additional information relevant for managed objects of the Fan class.

- Alarm Table Extension (*sunNemAlarmTable*)

  This augments the Equipment Table Extension. It provides the additional information relevant for managed objects of the Sensor class.

- Watchdog Table Extension (*sunNemWatchdogTable*)

  This augments the Equipment Table Extension. It provides the additional information relevant for managed objects of the Watchdog class.

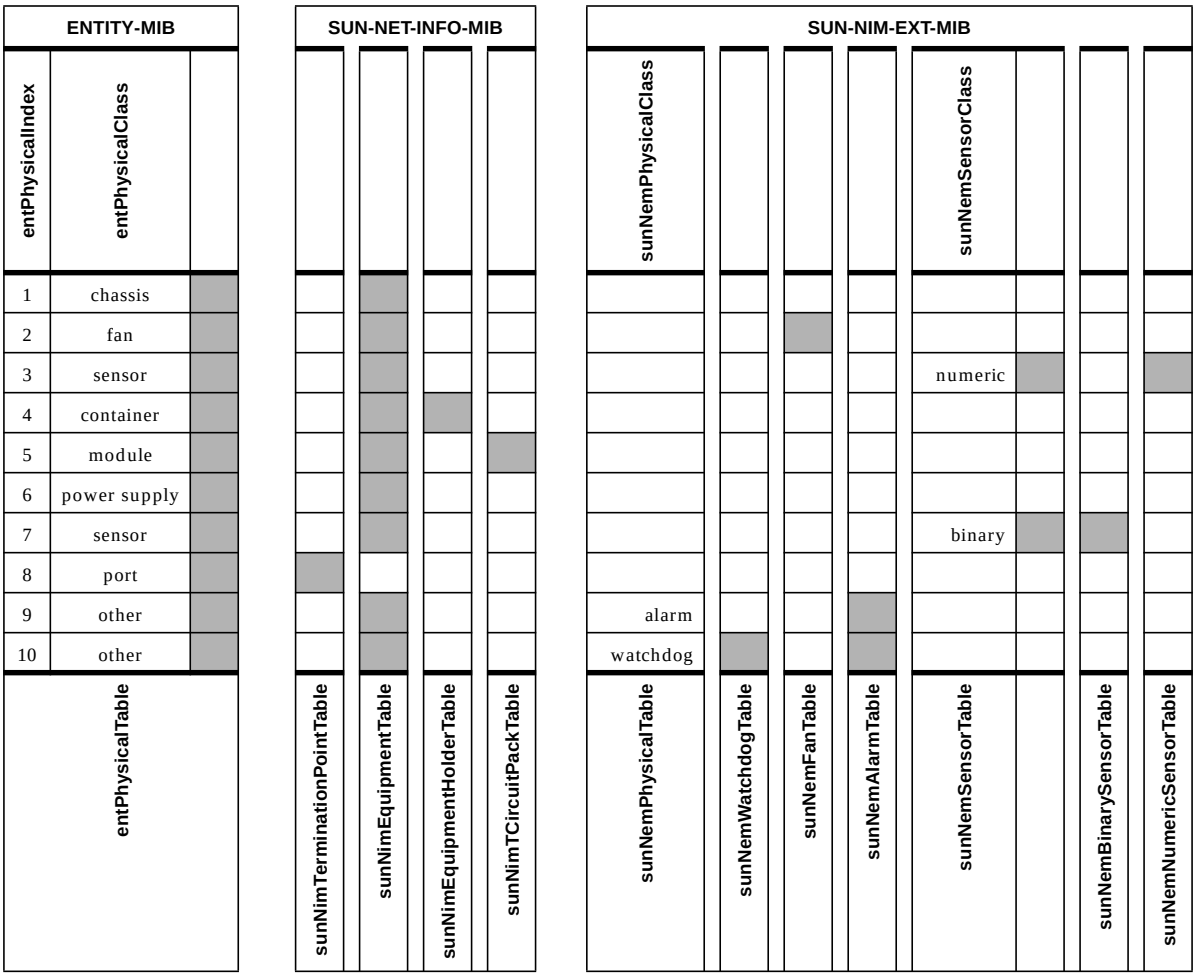

FIGURE 3-5 SunNem Table Extensions

CHAPTER 4

The SunNIM Hardware Resources

This chapter provides an overview of the SunNIM hardware resources and attributes, as defined in the SUN-NET-INFO-MIB and SUN-NIM-EXT-MIB. Only those attributes implemented in this release of Sun Netra SNMP Management Agent 1.0 are described here. The remaining attributes will be implemented in future releases.

SunNIM Class Hierarchy

FIGURE 4-1 shows the inheritance hierarchy of the SunNIM classes used to model hardware resources within a Sun Netra product.

The *Physical Entity* superclass provides an attribute for defining the relationship between managed objects. It also provides standard SNMP attributes which correspond to GNIM attributes in the Top, Termination Point and Equipment classes.

The classes *NIM Termination Point* and *NIM Equipment* are subclassed from *Physical Entity* to provide the additional attributes defined in the corresponding GNIM classes.

The *NIM Equipment* is subclassed to provide the *NIM Equipment Holder* and *NIM Circuit Pack*.

The *NIM Equipment* class is then further specialized to provide the DMTF-derived classes. The classes *NEM Power Supply* and *NEM Chassis* are defined to complete the alignment of SunNIM to standard SNMP classes of hardware resources.

The NEM subclass of *NIM Equipment* provides attributes specific to the hardware resource that is applicable for fault monitoring.

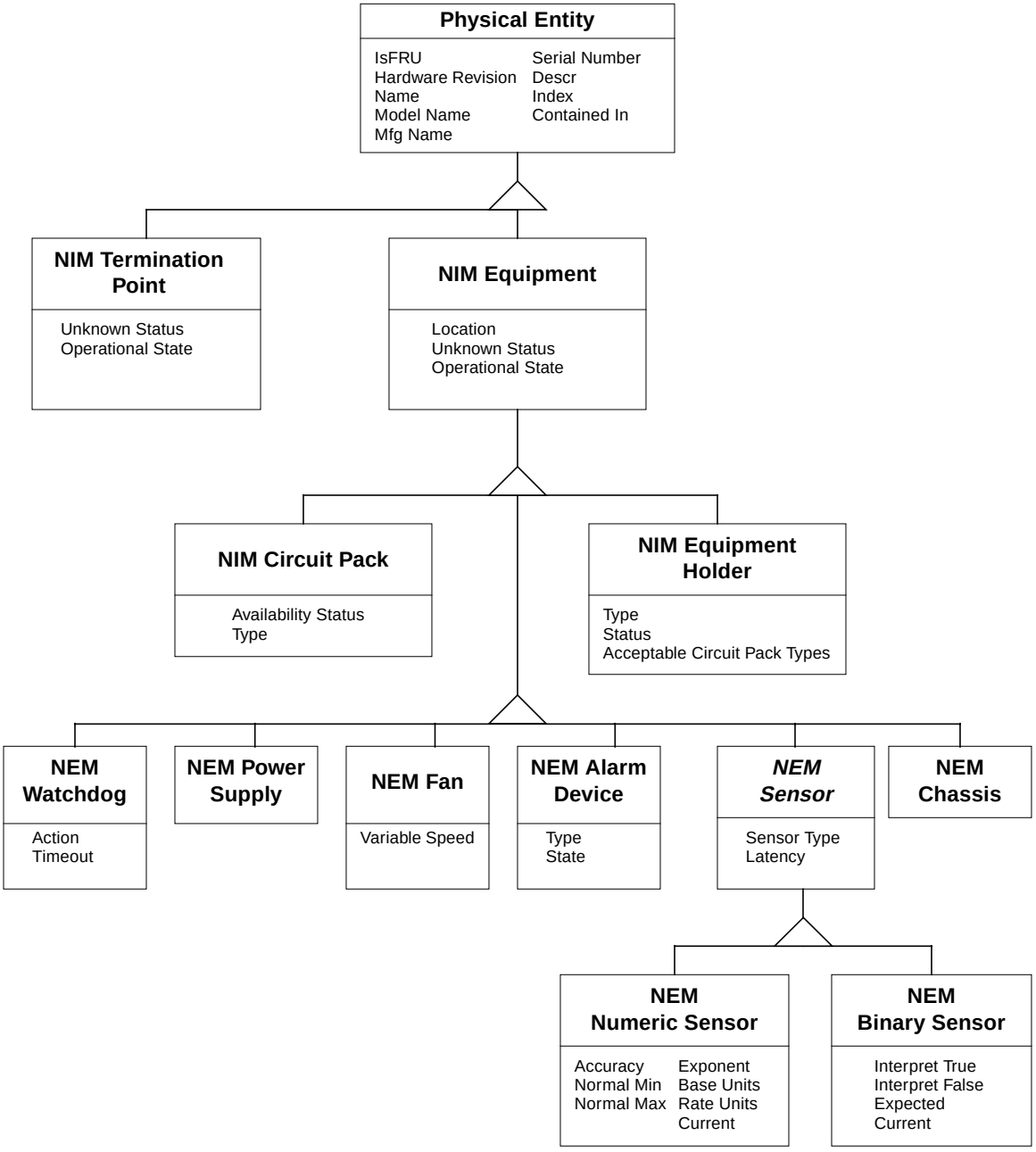

FIGURE 4-1 The SunNIM Hardware Resource Inheritance Class Diagram

SunNIM Class Definitions

Physical Entity

The *Physical Entity* superclass is used to represent the characteristics which are generic to all hardware resource:

- *Descr*
  This is a text string containing the known name for the hardware resource. This name is typically the name used to describe the resource in product documentation, on product legends or, possibly, the name stored in firmware.
- *Is FRU*
  This is a boolean representing whether the hardware resource is a field replaceable unit. Only hardware resources of the class *NIM Circuit Pack* are considered to be FRUs.
- *Hardware Rev*
  This is a text string containing the manufacturer's hardware revision information for the hardware resource. Not all hardware resources within Sun Netra Products have associated hardware revision information.
- *Name*
  This is a text string containing the logical name by which the hardware resource is known to the operating system and associated utilities. This name can be a device node or a defined name used by system utilities, where applicable. Not all resources have a device name.
- *Model Name*
  This is a text string containing the Manufacturer's customer visible part number or part definition. Not all hardware resources within Sun Netra Products have associated part numbers or definitions.
- *Serial Num*
  This is a text string containing the Manufacturer's serial number for the hardware resource. Not all hardware resources within Sun Netra Products have associated serial numbers.
- *Mfg Name*
  This is a text string containing the Manufacturer's name for the hardware resource. Not all hardware resources within tthe Sun Netra products have an associated manufacturer’s name.

The Physical Entity superclass also contains attributes which are used for describing the hierarchy of hardware resources :

- *Class*
  This enumerated type contains an indication of the general hardware type of a particular physical resource. The supported values of this class are defined by the ENTITY-MIB. This attribute can be used as an indication of the relevant Table Extensions for the managed object. The mapping between the Entity MIB classes and the SunNIM classes are as shown in TABLE 4-1:

TABLE 4-1 Physical Entity Superclass 'Class' Attribute Mapping

Entity Class	SunNIM Class
Chassis	NEM Chassis
Backplane	NIM Equipment
Container	NIM Equipment Holder
Power Supply	NEM Power Supply
Fan	NEM Fan
Sensor	NEM Sensor, *plus subclasses*
Module	NIM Circuit Pack
Port	NIM Termination Point
Stack	*Not Implemented*
Other	NIM Equipment, *plus subclasses*
Unknown	*Not Implemented*

- *Index*
  This is an integer which uniquely identifies the entry in the Physical Entity Table that identifies the managed object. the *NEM Chassis* managed object is always assigned an *Index* of '1'. Values for other hardware resources are not pre-allocated and will vary on each invocation of the agent.
- *Contained In*
  This is an integer representing the *Index* attribute of the managed object containing this managed object. This attribute models the relationship between the managed objects.

The following attributes of the ENTITY-MIB Physical Entity are not implemented as they do not map to the GNIM or DMFT standards, or they require functionality supported in future releases:

- Vendor Type
- Parent Rel Pos
- Firmware Rev
- Software Rev
- Alias
- Asset ID

The NIM Classes

The attributes of the NIM classes are used to represent the characteristics of hardware resources. The availability and operability of the resource to the manager are represented by the *state* of the managed object. Different NIM classes have a variety of attributes that express aspects of the managed object's *state*. These state attributes are shown in ***bold italics***.

NIM Equipment

The *NIM Equipment* class is used to represent the characteristics that are generic to all hardware resources, with the exception of ports. This class contains attributes representing configuration and generic health status information. This class is further subclassed to provide more detail configuration information and monitoring data for particular types of hardware resource. The NIM Equipment class has the following attributes:

- *Location Name*
  This is a text string containing a locator for the hardware resource. For resources contained directly within the Chassis, this attribute will correlate with legends on slots and product documentation, or will provide a geographical indication of the position of the hardware resource within the Chassis. Other hardware resources will typically have a *location* corresponding to the *Descr* of the managed object for the hardware resource in which it is contained.
- *Unknown Status*
  This attribute indicates whether the other state attributes may possibly not reflect the true state of the hardware resource. This is a boolean representing whether the managed object is able to accurately report faults against the hardware resource. If the hardware resource is unable truthfully to reflect its *state*, this attribute will be set to true.
- *Operational State*
  This is an enumerated type representing whether the hardware resource is physically installed and capable of providing service. This attribute contributes to the *state* of the managed object.

TABLE 4-2 NIM Equipment Operational State Values

Operational State	Description
Disabled	The resource is totally inoperable and unable to provide service to the user.
Enabled	The resource is partially or fully operable and available for use.

NIM Circuit Pack

The *NIM Circuit Pack* class is used to represent the characteristics that are generic to a replaceable hardware resource or FRU. A replaceable hardware resource is defined as a hardware module whose purpose is to package internal hardware components into a recognized form-factor. Typically, a FRU will have a defined form-factor and physical appearance. It can be a pluggable removable unit, which is plugged into a connector, it can be more permanently sited within a bay, or it can fit into a drawer, rack or shelf. The *NIM Circuit Pack* class has the following attributes:

- *Type*
  This is a text string used for assessing the hardware resource's compatibility with its container. This attribute can identify functionality and form-factor characteristics of the hardware resource.
- *Availability Status*
  This is an set of enumerated types further qualifying the *Operational State* of the managed object. This attribute contributes to the *state* of the managed object.

TABLE 4-3 NIM Circuit Pack Availability Status Values

Availability Status	Description
In Test	The hardware resource is undergoing a test procedure.
Failed	The hardware resource has an internal fault that prevents it from operating. *Operational State* is `disabled`.
Power Off	The hardware resource requires power to be applied and is not powered on.
Off Line	The hardware resource requires a routine operation to be performed to place it online and make it available for use. *Operational State* is `disabled`.
Off Duty	The hardware resource has been made inactive by an internal control process.
Dependency	The hardware resource cannot operate because some other resource on which it depends is unavailable. *Operational State* is `disabled`.
Degraded	The service available from the hardware resource is degraded in some respect, such as in speed or operating capability. However, the hardware resource remains available for service. *Operational State* is `enabled`.
Not Installed	The hardware resource represented by the managed object is not present, or is incomplete. *Operational State* is `disabled`.

NIM Equipment Holder

The *NIM Equipment Holder* class is used to represent the characteristics of hardware resources that are capable of holding removable hardware resources. The *NIM Equipment Holder* class has the following attributes:

- *Type*
  This is an enumerated type representing the holder type of the hardware resource:

TABLE 4-4 NIM Equipment Holder Type Values

Type	Description
Bay	A bay is typically a unit of vertical space within a rack that contains shelves or drawers for holding telecommunications equipment. SunNIM interprets its use within a chassis as a physical receptacle requiring cables for signal connections
Shelf	A horizontal support or subrack for holding telecommunications equipment within a rack.
Slot	A physical receptacle with an integral connector for signal connections for removable equipment.
Drawer	A horizontal enclosure for holding telecommunications equipment within a rack.
Rack	A rack is the support infrastructure for holding telecommunications equipment, holders, and cable management systems within a self-contained enclosure.

- *Status*
  This is an enumerated type indicating the status of the holder with regards to any replaceable hardware resources that it may contain.

TABLE 4-5 NIM Equipment Holder Status Values

Status	Description
Holder Empty	There is no removable hardware resource in the holder
In Acceptable List	The holder contains a removable hardware resource that is one of the types in the *Acceptable Circuit Pack Types* list
Not In Acceptable List	The holder contains a removable hardware resource recognizable by the network element; but not one of the types in the *Acceptable Circuit Pack Types* list
Unknown Type	The holder contains an unrecognizable removable hardware resource

- *Acceptable Circuit Pack Types*
  This is a list of text strings representing the types of removable hardware resource that are acceptable by the holder. These types are tested for compatibility with the removable hardware resource's *Type* attribute.

NIM Termination Point

The *NIM Termination Point* class is used to represent the characteristics of communication ports. The *NIM Termination Point* class has the following attributes:

- *Operational State*
  This is an enumerated type representing whether the port is providing the required physical connectivity. This attribute contributes to the *state* of the managed object.

TABLE 4-6 NIM Termination Point Operational State Values

Operational State	Description
Disabled	The port is unable to provide physical connection.
Enabled	The port is providing partial or full physical connectivity.

- *Unknown Status*
  This is a boolean representing whether the managed object is able to accurately report faults against the port. If the port is unable to reflect its true operational *state*, this attribute will be set to true. This attribute indicates that the other state attributes may not represent the true state of the communication port.

The NEM Classes

NEM Power Supply

The *NEM Power Supply* class is used to represent a power supply. It does not extend the characteristics of the *NIM Equipment* class. A power supply typically contains sensors representing monitored properties, for example voltages, current, and temperature. It can also contain other hardware resources such as fans. This is modeled using relationships between the managed objects.

If a power supply is a removable hardware resource, it will be modeled within a managed object of *NIM Circuit Pack* class.

NEM Watchdog

The *NEM Watchdog* class is used to represent the characteristics of timer hardware resources that allow the hardware to monitor the state of the operating system or applications. The *NEM Watchdog* class has the following attributes:

- *Action*
  This is an enumerated type representing the action taken by the watchdog if it is not reset within the period specified by the *Timeout*.

TABLE 4-7 NEM Watchdog Action Values

Action	Description
Status Only	The watchdog is readable by software, but performs no action
System Interrupt	The watchdog generates a hardware interrupt to the system being monitored
Reset	The watchdog reset the system being monitored
Power Off	The watchdog powers off the system being monitored
Power Cycle	The watchdog powers off and then on, the system being monitored

- *Timeout*
  This is an integer indicating the interval in microseconds after which the watchdog will timeout if not reset.

NEM Alarm Device

The *NEM Alarm Device* class is used to represent the characteristics of hardware resources which emit indications relating to problem situations, for instance buzzers, LEDs, relays,, vibrators, and software alarms. The *NEM Alarm Device* class has the following attributes:

- *Type*
  This is an enumerated type representing the means by which the alarm is communicated:

TABLE 4-8 NEM Alarm Device Type Values

Type	Description
audible	The alarm device is audible change on the device
visible	The alarm causes a visible change on the device
motion	The alarm causes motion of the device
switch	The alarm causes an electrical signal change
other	The alarm device type is not one of the above

- *State*
  This is an enumerated type representing the state of the alarm:

TABLE 4-9 NEM Alarm Device State Values

State	Description
unknown	The state of the alarm is undefined or unobservable
off	The alarm is inactive
steady	The alarm is active
alternating	The alarm is cycling between its inactive and active states

NEM Fan

The *NEM Fan* class is used to represent the characteristics of active cooling fans. A fan typically contains a sensor representing the speed of rotation. This is modeled using a relationship between the *NEM Fan* managed object and a managed object of class *NEM Binary Sensor* or *NEM Numeric Sensor.* The *NEM Fan* class has the following attribute:

- *Variable Speed*
  This is a boolean representing whether the fan supports variable speeds.

NEM Sensor

The *NEM Sensor* superclass is used to represent the generic characteristics of hardware resources that measure properties of other hardware resources. The *NEM Sensor* class has the following attributes:

- *Type*
  This is an enumerated type identifying the property that the sensor measures. Some of the possible values of *Type* are shown below:

TABLE 4-10 NEM Sensor Type Values

Type	Description
Temperature	Sensor for measuring the environmental temperature
Voltage	Sensor for measuring the electrical voltage
Current	Sensor for measuring the electrical current
Tachometer	Sensor for measuring the speed / revolutions of a device
Counter	A general purpose sensor which counts defined events

- *Latency*
  Where the sensor is polled, this integer represents the update interval measured in microseconds. Where the sensor is event-driven, this value represents the maximum expected latency in processing that event.

NEM Binary Sensor

A *NEM Binary Sensor* class is used to represent the characteristics of sensors which return binary output. The *NEM Binary Sensor* class has the following attributes:

- *Interpret True*
  This is a text string indicating the interpretation of a ‘true’ value from the sensor.
- *Interpret False*
  This is a text string indicating the interpretation of a ‘false’ value from the sensor.
- *Expected*
  This is a boolean indicating the anticipated value of the sensor.
- *Current*
  This is a boolean indicating the most recent value of the sensor.

NEM Numeric Sensor

A *NEM Numeric Sensor* class is used to represent the characteristics of sensors which can return numeric readings. The numeric sensor values are qualified by a Unit of Measurement as defined below:

Unit of Measurement = *Base Unit* * $10^{Exponent}$

This qualification allows for units of measurement such as milliamps and microvolts. If a *Rate Unit* is defined, the Unit of Measurement is further refined as below:

Unit of Measurement = *Base Unit* * $10^{Exponent}$ per *Rate Unit*

This qualification allows for units of measurement such as rpm and km/hr. The *NEM Numeric Sensor* class has the following attributes:

- *Accuracy*
  This is an integer indicating the degree of error of the sensor for the measured property. *Accuracy* can vary depending on whether the sensor reading is linear over its dynamic range.
- *Normal Min*
  This integer indicates the defined threshold below which the sensor reading is not expected to fall. This value is expressed in terms of the Unit of Measurement as defined above. This attribute may not be applicable to some sensors.
- *Normal Max*
  This integer indicates the defined threshold above which the sensor reading is not expected to rise. This value is expressed in terms of the Unit of Measurement as defined above. This attribute may not be applicable to some sensors.
- *Exponent*
  This integer is used to scale the *Base Unit* by some power of 10.
- *Base Units*
  This enumerated type indicates the unit of measurement, prior to qualification as defined above. Examples of values of this type are:
  - deg C
  - volts
  - amps
  - revolutions

- *Rate Units*
  This enumerated type indicates whether the sensor is measuring as absolute value or a rate. In the latter case the rate is expressed as ‘per unit of time’. Examples of values of this type are:
  - per microsecond
  - per millisecond
  - per second
  - per minute
  - per hour
  - none
- *Current*
  This is an integer indicating the most recent value of the sensor.

NEM Chassis

The *NEM Chassis* class is used to represent the primary enclosure for the modeled Sun Netra product. It does not extend the characteristics of the *NIM Equipment* class. The chassis contains all the modeled hardware resources, and is not contained within any other hardware resource.

CHAPTER 5

The SunNIM Notifications

This chapter describes the SunNIM notifications classes and attributes, as defined in the SUN-NET-INFO-MIB. Only attributes implemented in this release are described here. The remaining attributes will be implemented in future releases.

SunNIM notification classes are asynchronous messages sent by the agent to registered network managers. They are used to convey event information faster than can be achieved through polling the SunNIM managed objects.

SunNIM Notifications Class Hierarchy

FIGURE 5-1 shows the inheritance hierarchy of the SunNIM Notifications classes.

The GNIM set of Notification classes are represented using an hierarchy of both abstract and concrete classes exploiting the common attributes across these classes.

SunNIM Event Record Classes Overview

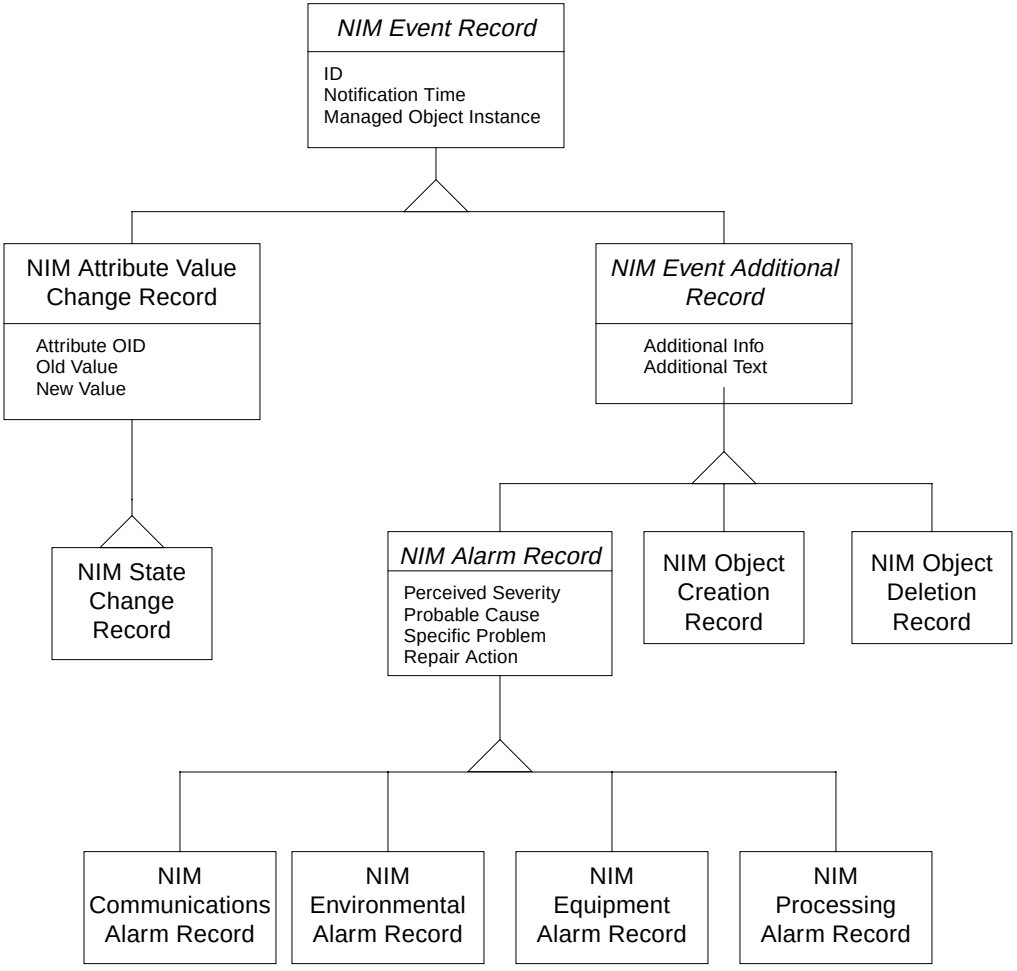

FIGURE 5-1 Event Records Inheritance Class Diagram

SunNIM Class Definitions

NIM Event Record

The NIM Event Record superclass represents the attributes common to all notifications. This class is further subclassed to provide additional information pertinent to the particular event that it records. The NIM Event Record has the following attributes:

- *ID*
  This is an integer uniquely identifying the notification and an indication of the order in which the notifications were generated by the agent. Note that the agent does not guarantee that its sequencing reflects the order of the underlying events from which the notifications were generated.
- *Notification Time*
  This is a timestamp that currently identifies the time at which the notification was generated by the agent. In future releases it is intended that this reflect the time at which the underlying event occurred.
- *Managed Object Instance*
  This is an OID that provides a direct reference to an entry in the Physical Entity Table representing the hardware resource with which the event is associated.

NIM Event Additional Record

The NIM Event Additional Record superclass represents the additional attributes common to notifications that are generated when the following events occur:

- Object Creation
- Object Deletion
- Alarms

This class is further subclassed to provide additional information pertinent to the particular event that it records. The Nim Event Additional Record has the following attributes:

- *Additional Info*
  This is an optional OID of an object which can provide additional information relevant to this notification.
- *Additonal Text*
  An optional text string that can provide additional information relevant to this notification.

NIM Alarm Record

The NIM Alarm Record superclass represents the additional attributes common to all notifications representing alarms.

This class is further subclassed for identification of the class of alarm that occurred. The NIM Alarm Record has the following attributes:

- *Perceived Severity*
  This enumerated type defines six severity levels, which provide an indication of how the service of the hardware resource has been affected by the problem. These values are shown in TABLE 5-1.

TABLE 5-1 NIM Alarm Record Perceived Severity Values

Perceived Severity	Description
`Cleared`	This clears all alarms for this hardware resource of the same alarm class that have the same *Probable Cause* and *Specific Problem* (if given).
`Indeterminate`	The severity level for the alarm can not be determined
`Critical`	A service affecting condition has occurred and an immediate corrective action is required.
`Major`	A service affecting condition has occured and an urgent corrective action is required.
`Minor`	A non-service affecting condition has occurred and corrective action should be taken in order to prevent a more serious condition arising.
`Warning`	A potential or impending service affecting fault condition has been detected and action should be taken to prevent a more serious condition arising.

The agent currently reports all notifications as being of an `Indeterminate` *Perceived Severity.*

- *Probable Cause*
  An optional enumerated type that provides further qualification as to the type of condition that caused an alarm to be generated. Examples of values of this type are:
  - Cooling system problem
  - I/O device error
  - Power problem
  - Software error
- *Specific Problem*
  An optional text string that identifies further refinements to the *Probable Cause* of the alarm.

- *Repair Action*
  A boolean that indicates whether the condition that has occurred requires that the manager intiate a corrective action.

NIM Communications Alarm Record

The NIM Communications Alarm Record class does not extend the information provided by the NIM Alarm Record class. This alarm record is used to record that the associated hardware resource has detected a communications error.

NIM Environmental Alarm Record

The NIM Environmental Alarm Record class does not extend the information provided by the NIM Alarm Record class. This alarm record is used to record that the associated hardware resource has detected a problem in the environment.

NIM Equipment Alarm Record

The NIM Equipment Alarm Record class does not extend the information provided by the NIM Alarm Record class. This alarm record is used to record that the associated hardware resource has detected a fault.

NIM Processing Alarm Record

The NIM Processing Alarm Record class does not extend the information provided by the NIM Alarm Record class. This alarm record is used to record that the associated hardware resource has detected a software or processing failure.

NIM Object Creation Record

The NIM Object Creation Record class does not extend the information provided by the NIM Event Additional Record class. This record is used to indicate that a hardware resource has been added to the hierarchy below the associated hardware resource; this may be due to a hot-plugging event. The *Additional Info* attributes contains the OID of the *Physical Entity Table* entry representing the added resource.

NIM Object Deletion Record

The NIM Object Deletion Record class does not extend the information provided by the NIM Event Additional Record class. This record is used to indicate that a hardware resource has been removed from the hierarchy below the associated hardware resource. The *Additional Info* attributes contains the OID of the *Physical Entity Table* entry representing the removed resource.

NIM Attribute Value Change Record

The NIM Attribute Value Change Record superclass represents the additional attributes common to notifications representing attribute changes of the associated hardware resource.

This class is further subclassed for attribute changes which define the *state* of the associated hardware resource. The NIM Attribute Value Change Record has the following attributes:

- *Attribute OID*
  This is an OID that provides a direct reference to an object in the *Physical Entity Table* that represents the managed object's attribute whose value has changed.
- *Old Value*
  This identifies the old value of the changed attribute of the managed object. The type of *Old Value* corresponds to that of the changed attribute.
- *New Value*
  This identifies the new value of the changed attribute of the managed object. The type of *New Value* corresponds to that of the changed attribute.

NIM State Change Record

The NIM State Change Record class does not extend the information provided by the NIM Attribute Value Change Record class. This record is used to indicate an change in the managed object's attribute that reflects an aspect of the *state* of the hardware resource..

APPENDIX **A**

SNMP Basics

This Appendix provides a brief introduction to the essential features of SNMP. It is by no means exhaustive and addresses the issues which are of particular relevance to Sun Netra SNMP Management Agent 1.0. If you are experienced with SNMP you may not need to read this Appendix.

The Simple Network Management Protocol (SNMP) is an open internet standard for management of networked devices (systems). It is defined, in line with other internet standards by a number of RFCs. There are two versions of SNMP of interest here, SNMPv1, RFC1157/STD 15 (see `http://www.ietf.org/rfc/rfc1157.txt`) being the root document from which the other related RFCs are referenced, and SNMPv2 (a superset of SNMPv1) which is defined by a supplementary set of RFCs, the most significant being RFC2578 (see `http://www.ietf.org/rfc/rfc2578.txt`). For a more complete understanding of SNMP than is conveyed by the following, you should refer to these standards.

SNMP is a network protocol which allows devices to be managed remotely by a Network Management Station (NMS). In order to be managed, a device must have an SNMP Agent associated with it. The Agent is responsible for receiving requests for data representing the state of the device from the NMS and providing an appropriate response. The Agent can also accept data from the NMS in order to allow control of the state of the device. Additionally, the Agent can generate SNMP Traps, which are unsolicited messages sent to selected NMS(s) to signal significant events relating to the device.

In order to manage/monitor devices their characteristics must be represented using some format known to both the Agent and the NMS. These characteristics can represent physical properties such as fan speeds, or services such as routing tables. The data structure defining these characteristics is known as a Management Information Base (MIB). This data model is typically organized into tables, but can also include simple values. An example of the former is routing tables, and an example of the latter is a timestamp indicating the time at which the agent was started.

The MIB is defined using a language called ASN.1, the discussion of which is beyond the scope of this document. For reference, the structure of the MIBs for SNMPv2 is defined by its Structure of Management Information (SMI) defined in RFC2578 (see `http://www.ietf.org/rfc/rfc2578.txt`). This defines the syntax and basic data types available to MIBs. The Textual Conventions (type definitions) defined in RFC2579 (see `http://www.ietf.org/rfc/rfc2579.txt`) define additional data types and enumerations.

In order for an NMS to manage a device via its Agent, the MIB corresponding to the data presented by the Agent must be loaded into the NMS. The mechanism for doing this varies depending on the implementation of the network management software. This gives the NMS the information required to address and correctly interpret the data model presented by the Agent. Note that MIBs can reference definitions in other MIBs, so in order to use a given MIB, it may be necessary to load others.

The MIB is a definition for a virtual datastore accessible via SNMP, the content being provided either by corresponding data maintained by the Agent, or by the Agent obtaining the required data on demand from the managed device. For writes of data by the NMS to this virtual data, the Agent will typically perform some action affecting the state either of itself or the managed device. In order to address the content of this virtual datastore the MIB is defined in terms of Object Identifiers (OIDs) which uniquely identify each data entry. An OID consists of an hierarchically arranged sequence of integers providing a unique name space. Each assigned integer has a associated text name. For example, the OID `1.3.6.1` corresponds to the OID name `iso.org.dod.internet` and `1.3.6.1.4` corresponds to the OID name `iso.org.dod.internet.private`. The numeric form is used within SNMP protocol transactions, whereas the text form is used in user interfaces to aid readability. Objects represented by such OIDs are commonly referred to by the last component of their name as a shorthand form. In order to avoid confusion arising from this convention, it is normal to apply a MIB-specific prefix, such as sunNim, to all object names defined therein.

All addressable objects defined in the MIB have associated maximum access rights, for instance. read-only or read-write, which determine what operations the NMS will permit the operator to attempt. The Agent is able to apply lower access rights as required, that is, it is able to refuse writes to objects which are considered read-write. This refusal may be done on the grounds of applicability of the operation to the object being addressed, or on the basis of security restrictions which can limit certain operations to restricted sets of NMS. The mechanism used to communicate security access rights is that of community strings. These are simply text strings such as ‘private’ and ‘public’ which are passed with each SNMP data request.

Much of the data content defined by MIBs is of a tabular form, organized as entries consisting of a sequence of objects (each with their own OIDs). For example, a table of fan characteristics could consist of a number of rows, one per fan, with each row containing columns corresponding to the current speed, the expected speed, and the minimum acceptable speed. The addressing of the rows within the table can be a

simple single dimensional index (a row number within the table, for instance ‘6’), or a more complex, multi-dimensional, instance specifier such as an IP address and port number (for instance `127.0.0.1, 1234`). In either case a specific data item within a table is addressed by specifying the OID giving its prefix (for instance `myFanTable.myFanEntry.myCurrentFanSpeed`) with a suffix instance specifier (for instance `127.0.0.1.1234` from the previous example) to give `myFanTable.myFanEntry.myCurrentFanSpeed.127.0.0.1.1234`.

Each table definition within the MIB has an INDEX clause which defines which instance specifier(s) to use to select a given entry. The SMI defining the MIB syntax provides an important capability whereby tables can be extended to add additional entries, effectively adding extra columns to the table. This is achieved by defining a table with an INDEX clause which is a duplicate of that of the table being extended.

An NMS communicates with the Agent by sending (UDP) packets to a well known port (161) on the system on which the agent is running. Should there be many Agents running on a given system, each managing different devices, there is a potential conflict for the use of the port resource. One possible solution is to use different, non-standard, ports numbers for each Agent. An alternative is to introduce the concept of a Master Agent which accepts SNMP requests on behalf of all the Agents running on a given system and forwards these requests as appropriate. This has the benefit of allowing a NMS to access all SNMP Agents in a consistent manner. Sun Netra SNMP Management Agent 1.0 supports either approach.

APPENDIX **B**

Bibliography

IETF SNMPv2 (RFC 2578)

IETF ENTITY-MIB (RFC 2737)

ITU-T M.3020 TMN Methodology

ITU-T M.3100 Generic Network Information Model

DMTF CIM v2 Physical and Device Schemas

ITU-T X.730's Series: System Management Functions

ITU-T X.720's Series: Structure of Management Information

ITU-T M-3010 Maintenance: Telecommunications Management Network

Glossary

agent An application which is capable of performing management operations on *Managed Objects* and emitting notifications on behalf of *Managed Objects*.

CIM see *Common Information Model.*

Class see *Managed Object Class*.

Common Information Model A common data model of an implementation-neutral schema for describing overall management information in a network enterprise environment.

container A *hardware resource* that physically contains another (containee) hardware resource.

DMTF see *Distributed Management Task Force*.

Distributed Management Task Force An industry organization developing management standards and initiatives for desktop, enterprise and internet environments.

ENTITY-MIB The specification of information in a manner that allows standard access through a network management protocol. This *MIB* describes *managed objects* used for managing multiple logical and physical entities managed by a single SNMP *agent*.

Field-Replaceable Unit An assembly that a manufacturer replaces on failure of an assembly component.

FRU see *Field-Replaceable Unit*.

Generic Network Information Model This is an ITU-T recommendation. The model describes *Managed Object classes* and their properties that are generic and useful, to describe information exchanged across interfaces defined in M.3010 architectures and services.

GNIM see *Generic Network Information Model*.

Hardware Resource A physical component or subsystem providing functionality within a *Network Element*.

Hierarchy of Hardware Resources A collection of *Hardware Resources* organised in a tree structure representing the physical architecture of the resources within the Sun Netra Product.

IETF see *Internet Engineering Task Force.*

Internet Engineering Task Force A large open international community of network designers, operators, vendors, and researchers concerned with the evolution of internet architecture and the smooth operation of the internet.

ITU-T A permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for issuing Recommendations with a view to standardizing telecommunications on a worldwide basis.

Managed Object An abstraction of a resource, for the purposes of control, coordination, and monitoring, that can be managed using network management protocols.

Managed Object Class A named set of *Managed Objects* sharing the same (named) set of attributes, notifications and operational behaviour.

Management Information Base Management information is viewed as a collection of *managed objects*, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules.

Manager An application which is capable of issuing management operations and receiving notifications.

MIB see *Management Information Base.*

NEM *NIM* Extension *MIB.*

Network Element Telecommunication equipment (or groups / parts of telecommunication equipment) and support equipment or any item or groups of items.

Network Information Model Set of *managed objects*.

Network Management Station A hardware / software resource giving access to *hardware resource* information.

NIM Classes The set of *Managed Objects* that are used to provide the *GNIM* management view of a Sun Netra Product.

NEM Classes The set of *Management Objects* that are used to provide the *DMTF*-derived extensions to the *NIM Classes* for the purposes of supporting platform-specific management of a Sun Netra Product.

NMS see *Network Management Station*.

Object Identifier An hierarchically-arranged sequence of integers providing a unique name space. This is used within a logical, integrated naming and addressing scheme for identifying and locating *Managed Objects.*

OID see *Object Identifier.*

Platform Building Block see *Managed Object Class.*

Requests for Comments A series of notes discussing and defining aspects of computer communication, focusing on network protocols, procedures, programs, and concepts.

RFC see *Requests for Comments.*

Simple Network Management Protocol A network protocol which allows devices to be managed remotely by a *Network Management Station*.

SNMP see *Simple Network Management Protocol*.

SunNem MIB see *SUN-NIM-EXT-MIB.*

SUN-NET-INFO-MIB The *MIB* which defines the *NIM Classes*.

SunNIM The set of *managed objects* that are used to provide the management view of a Sun Netra Product.

SUN-NIM-EXT-MIB The *MIB* which defines the *NEM Classes*.

SunNim MIB see *SUN-NET-INFO-MIB.*

Telecommunications Management Network (M.3010) This is an *ITU-T* Recommendation. The Telecommunications Management Network supports management activities associated with telecommunication networks.

TMN see *Telecommunications Management Network.*

Index

O

P

R

S

T

U

V

W