Sun™ Netra™ SNMP Management Agent 1.0 Supplement for Netra t Servers

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

Part No. 806-5818-10
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 solutions for Netra t servers.

Supported systems covered by this manual are:

- Netra t1 Model 100
- Netra t1 Model 105
- Netra t 1120
- Netra t 1125
- Netra t 1400
- Netra t 1405

Refer to the *Sun Netra SNMP Management Agent 1.0 User's Reference Guide* for an overview of the generic capabilities of the Sun Netra SNMP Management Agent 1.0, the *Sun Netra SNMP Management Agent 1.0 Installation and Configuration Guide*, and the documentation supplied with your system.

How This Book Is Organized

Chapter 1 describes the Sun Netra SNMP Management Agent 1.0 SNMP agent implementation on the Netra t1 Model 100 and Model 105.

Chapter 2 describes the Sun Netra SNMP Management Agent 1.0 SNMP agent implementation on the Netra t 1120 and 1125.

Chapter 3 describes the Sun Netra SNMP Management Agent 1.0 SNMP agent implementation on the Netra t 1400 and 1405.

Related Documentation

Application	Title	Part Number
Installation and configuration	*Sun Netra SNMP Management Agent 1.0 Installation and Configuration Guide*	806-5111-10
Generic user and reference information	*Sun Netra SNMP Management Agent 1.0 User's Reference Guide*	806-5817-10
Further information about Netra t 1120/1125 servers	*Netra t 1120/1125 Installation and Basic Maintenance Guide*	805-6803-10
Further information about Netra t1 Model 100/105 servers	*LOMlite User's Guide*	806-2038-10
Further information about 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 (806-5818-10) of your document in the subject line of your email.

CHAPTER 1

Netra t1 Model 100/105

This chapter provides the information available from the Sun Netra SNMP Management Agent 1.0 Management Interface on a Netra t1 Model 100 and Model 105.

The hardware resources presented by the SNMP agent for the Netra t1 Model 100/105 platform are shown in TABLE 1-16. This also illustrates the physical hierarchy of these resources. The table should be used in conjunction with FIGURE 1-1, FIGURE 1-2 and FIGURE 1-3 which identify the location of the hardware resources in the platform.

Each hardware resource in the table is shown together with the SunNIM class used to represent it.

In addition to static hardware resource characteristics, some hardware resources in the physical hierarchy have the capability to provide dynamic status information for the managed object in question. Hardware resources which can provide dynamic status information are shown in bold text in the table.

Note that the SCSI devices in the system are detected when the agent is started, but that subsequent insertion/removal of drives is not detected.

PCI cards are also detected when the agent is started, but they have no dynamic state associated with them.

The following sections describe the representation and monitoring of the hardware resources with dynamic behavior.

Fans

A fan on the Netra t1 Model 100/105 is modeled using the NemFan class. The fans are identified in the *entityMIB.entityPhysical.entPhysicalTable* by their *entPhysicalDescr* of 'Fan <*n*>', or by the *entPhysicalName* of 'fan<*n*>', where <*n*> is a number in the range 1 through 3.

Each fan contains a tachometer represented using the NemNumericSensor class. This is used to indicate the current speed of the fan expressed in revolutions per minute (RPM). The tachometers are identified by their *entPhysicalDescr* of 'Fan <*n*> Tachometer' where <*n*> is a number in the range 1 through 3 corresponding to the fan being monitored. The search can be accelerated by reference to the *entityMIB.entityMapping.entPhysicalContainsTable* to establish what is contained in a previously-located fan.

The NemFan class used to represent the fans uses the
- *sunNimEquipmentTable* and
- *sunNemMIB.sunNemMIBObjects.sunNemFanTable*

extensions to the *entPhysicalTable*. The *sunNemFanVariableSpeed* attribute in the *sunNemFanTable* is set to `true(1)`. NemFan entries in the *entPhysicalTable* have an *entPhysicalClass* of `fan(7)`.

TABLE 1-1 Netra t1 Model 100/105 sunNemFanTable – Fans

sunNemFanEntry	
Variable	**Value**
sunNemFanVariableSpeed	`true(1)`

The NemNumericSensor class used to represent the fan tachometers uses the following extensions to the *entPhysicalTable*:
- *sunNimEquipmentTable,*
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable,* and
- *sunNemMIB.sunNemMIBObjects.sunNemNumericSensorTable*

NemNumericSensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* is configured for the fan tachometers as shown in TABLE 1-2

TABLE 1-2 Netra t1 Model 100/105 sunNemSensorTable – Fan Tachometers

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`numeric(2)`
sunNemSensorType	`tachometer(6)`
sunNemSensorLatency	<*undefined*>

The *sunNemNumericSensorTable* is configured as shown in TABLE 1-3.

TABLE 1-3 Netra t1 Model 100/105 sunNemNumericSensorTable – Fan Tachometers

sunNemNumericSensorEntry	
Variable	**Value**
sunNemNumericSensorBaseUnits	`revolutions(39)`
sunNemNumericSensorExponent	`0`
sunNemNumericSensorRateUnits	`perMinute(5)`
sunNemNumericSensorCurrent	<*variable*>
sunNemNumericSensorNormalMin	`1992`
sunNemNumericSensorNormalMax	<*undefined*>
sunNemNumericSensorAccuracy	<*undefined*>

Note that the value for *sunNemNumericSensorNormalMin* given above is typical for a Netra t1 Model 100/105, but may vary according to fan specification.

Fan Failures

The fan is considered to no longer be providing service if its reported speed falls below an expected normal minimum as indicated by its tachometer's *sunNemNumericSensorNormalMin* value.

The operability of the fan is represented using the *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable,* an extension to the *entPhysicalTable,* which contains the *sunNimEquipmentOperationalState*.

The fan will have its *sunNimEquipmentOperationalState* set to `disabled(1)` should its contained tachometer indicate a failure condition. This state will be restored to `enabled(2)` should the fan recover to an operational state. An SNMP *sunNimStateChange* trap is generated in response to an operational state change. When a fan first fails the Fault LEDs are set to flash. Consequently there will be additional SNMP *sunNimAttributeChange* traps generated as a side effect of the fan failure.

The *sunNimStateChange* trap takes the form shown in TABLE 1-4:

TABLE 1-4 Netra t1 Model 100/105 sunNimStateChange – Fan Tachometers

sunNimStateChange	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNimEquipmentTable.sunNimEquipmentEntry.sunNimEquipmentOperationalState.<*instance*>[1]
sunNimNotificationOldInteger	`disabled(1)` or `enabled(2)`
sunNimNotificationNewInteger	`disabled(1)` or `enabled(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed fan.

sunNimNotificationObject is a reference to the first column of the appropriate row of the *entPhysicalTable* in the ENTITY-MIB, and the *sunNimNotificationChangedOID* is a reference to the state attribute which has changed (in this case the *sunNimEquipmentOperationalState*). The *sunNimNotificationOldInteger* and *sunNimNotificationNewInteger* values represent the old and new integer enumerations of the changed state.

An alarm notification is also generated to indicate a fan failure, as shown in TABLE 1-5:

TABLE 1-5 Netra t1 Model 100/105 sunNimEnvironmentalAlarm – Fan Failure

sunNimEnvironmentalAlarm	
Variable	**Value**
sunNimNotificationEventId	*<unique numeric identifier>*
sunNimNotificationTime	*<date> <time>*
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.*<instance>*[1]
sunNimNotificationAdditionalInfo	`0`
sunNimNotificationAdditionalText	
sunNimNotificationPerceivedSeverity	`indeterminate(1)`
sunNimNotificationProbableCause	`coolingFanFailure(107)`
sunNimNotificationSpecificProblem	Fan *<n>* speed below normal minimum threshold
sunNimNotificationRepairAction	`false(2)`

1. *<instance>* indicates the row of the *entPhysicalTable* associated with the failed fan.

Note – Attribute Value Change events (traps) are not generated for changes to the current reported tachometer reading, *sunNemNumericSensorCurrent*, as it is not, in itself, of much significance, and potentially changes frequently.

Hence there are four mechanisms available to detect fan status:

1. Polling *sunNemNumericSensorCurrent* and *sunNemNumericSensorNormalMin* of the fan's tachometer, comparing their values.
2. Polling *sunNimEquipmentOperationalState* of the fan.
3. Receiving a *sunNimStateChange* trap corresponding to the change of *sunNimEquipmentOperationalState* of the fan.
4. Receiving a *sunNimEnvironmentalAlarm* trap with a *sunNimNotificationProbableCause* of `coolingFanFailure(107)`. Note that this will only indicate failure and not any subsequent recovery.

Power Supplies

The power supply unit (PSU) is modeled using the NemPowerSupply class. The PSU is identified by its *entPhysicalDescr* of 'PSU'.

The PSU contains voltage sensors modeled using the NemBinarySensor class. There are two sensors for monitoring the PSU inputs A/B. These are only of relevance in a DC system (Netra t 1 Model 100), as an AC system (Netra t 1 Model 105) will cease to operate should the input power be removed. The input voltage sensors are identified by their *entPhysicalDescr* of 'PSU Input A' or 'PSU Input B'. The search can be accelerated by reference to the *entPhysicalContainsTable* to establish what is contained in the previously-located PSU.

The PSU is always reported as being operational as this is implicit in the agent being able to respond. It is thus not possible to report the failure of a PSU.

The NemPowerSupply class used to represent the PSU uses the *sunNimEquipmentTable* extension to the *entPhysicalTable.* NemPowerSupply entries in the *entPhysicalTable* have an *entPhysicalClass* of `powerSupply(6)`.

The NemBinarySensor class used to represent the voltage sensors uses the following extensions to the *entPhysicalTable*:

- *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemBinarySensorTable*

NemBinarySensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* configuration for the Netra t1 Model 100/105 PSU voltage sensors is shown in TABLE 1-6:

TABLE 1-6 Netra t1 Model 100/105 sunNemSensorTable – PSU Voltage Sensors

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`binary(1)`
sunNemSensorType	`voltage(4)`
sunNemSensorLatency	*<undefined>*

The *sunNemBinarySensorTable* is configured as shown in TABLE 1-7:

TABLE 1-7 Netra t1 Model 100 / 105 sunNemBinarySensorTable – PSU Voltage Sensors

sunNemBinarySensorEntry	
Variable	**Value**
sunNemBinarySensorCurrent	`true(1)` or `false(2)`
sunNemBinarySensorExpected	`true(1)`
sunNemBinarySensorInterpretTrue	Voltage within expected range
sunNemBinarySensorInterpretFalse	Voltage outside expected range

PSU Input Failures

The detection of PSU input failures is only applicable to systems with DC power supplies.

Should either of the A or B DC inputs to the PSU fall below the minimum acceptable level, the corresponding voltage sensor's state, reported as *sunNemBinarySensorCurrent* in the corresponding *sunNemBinarySensorTable* entry will change from `true(1)` to `false(2)`. Recovery of the input will result in the value returning to `true(1)`. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 1-8:

TABLE 1-8 Netra t1 Model 100 / 105 sunNimAttributeChangeInteger Trap for PSU Input Failures

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemBinarySensorTable.sunNemBinarySensorEntry.sunNemBinarySensorCurrent.<*instance*>[1]
sunNimNotificationOldInteger	`true(1)` or `false(2)`
sunNimNotificationNewInteger	`true(1)` or `false(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed PSU input.

Hence there are two mechanisms available to detect PSU input status:

1. Poll *sunNemBinarySensorCurrent* of the PSU's inputs.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemBinarySensorCurrent*.

LOMlite Alarms / Fault LEDs

The LOMlite alarms and the fault LEDs are modeled using the NemAlarmDevice class. The alarms provide a software status indication only and have no physical alarm device associated with them, whereas the LOMlite's fault indicator is associated with two LEDs, one on the front panel, and one on the rear of the chassis. The alarms / fault LEDs can be identified by their *entPhysicalDescr* of 'Alarm 1', 'Alarm 2', 'System Alarm', or 'Fault LED'.

The NemAlarmDevice class used to represent the alarms / fault LEDs uses the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemAlarmTable*

NemAlarmDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the LOMlite alarms / fault LEDs as shown in TABLE 1-9:

TABLE 1-9 Netra t1 Model 100 / 105 sunNemPhysicalTable – LOMlite Alarms / Fault LEDs

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`alarm(2)`

The *sunNemAlarmTable* is configured as shown in TABLE 1-10:

TABLE 1-10 Netra t1 Model 100/105 sunNemAlarmTable – LOMlite Alarms/Fault LEDs

sunNemAlarmTable	
Variable	**Value**
sunNemAlarmType	`other(1)` for the alarms, `visible(3)` for the fault LEDs
sunNemAlarmState	`off(2)`, `steady(3)` or `alternating(4)`

Note that the system alarm (Alarm 3) is modeled according to the convention that any associated alarm indicator is 'on' when the system is active with the ASR watchdog running correctly, and 'off' otherwise. Its state can also be changed manually. This is the reverse of the state reported by the alarms utilities, which report the status of Alarm 3 directly rather than that of the derived system indicator.

LOMlite Alarms/Fault LEDs State Changes

Should the state of any of the alarms or the fault LEDs change, the *sunNemAlarmState* in the corresponding *sunNemAlarmTable* entry will change. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 1-11:

TABLE 1-11 Netra t1 Model 100/105 sunNimAttributeChangeInteger – LOMlite Alarms/Fault LED Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemAlarmTable.sunNemAlarmEntry.sunNemAlarmState.<*instance*>[1]
sunNimNotificationOldInteger	`off(2)`, `steady(3)` or `alternating(4)`
sunNimNotificationNewInteger	`off(2)`, `steady(3)` or `alternating(4)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the alarm device.

Hence there are two mechanisms available for detecting alarm / fault LED state changes:

1. Poll the alarm / fault LED's *sunNemAlarmState*.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemAlarmState*.

ASR Watchdog

The ASR watchdog is modeled using the NemWatchdogDevice class. It can be identified by the *entPhysicalDescr* of 'Watchdog'. It is only modeled if it is enabled. If it is disabled, once the agent is running it will be removed from the model. Similarly, it will be added to the model if it is enabled.

The NemWatchdogDevice is represented using the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemWatchdogTable*

NemAlarmDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the ASR watchdog as shown in TABLE 1-12:

TABLE 1-12 Netra t1 Model 100 / 105 sunNemPhysicalTable – ASR Watchdog

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`watchdog(3)`

The *sunNemWatchdogTable* is configured as shown in TABLE 1-13:

TABLE 1-13 Netra t1 Model 100 / 105 sunNemWatchdogTable – ASR Watchdog

sunNemWatchdogTable	
Variable	**Value**
sunNemWatchdogAction	`statusOnly(1)` or `systemReset(3)`
sunNemWatchdogTimeout	<*variable*> milliseconds

ASR Watchdog Changes

Should either of the attributes in the *sunNemWatchdogTable* change, an SNMP *sunNimAttributeChange* trap is generated.

The *sunNimAttributeChangeInteger* trap will take one of the forms shown in TABLE 1-14 and TABLE 1-15:

TABLE 1-14 Netra t1 Model 100/105 sunNimAttributeChangeInteger – ASR Watchdog Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogAction.<*instance*>[1]
sunNimNotificationOldInteger	statusOnly(1) or systemReset(3)
sunNimNotificationNewInteger	statusOnly(1) or systemReset(3)

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog

TABLE 1-15 Netra t1 Model 100/105 sunNimAttributeChangeInteger – ASR Watchdog Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogTimeout.<*instance*>[1]
sunNimNotificationOldInteger	<*variable*>
sunNimNotificationNewInteger	<*variable*>

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog

Hence there are two mechanisms available to detect such attribute changes:

1. Poll the appropriate ASR watchdog attribute.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of the appropriate ASR watchdog attribute.

The intended mode of operation is that a watchdog timer expiry is configured to reset the system. Should the *sunNemWatchdogAction* attribute be set to `statusOnly(1)`, the state of the watchdog can be monitored by observing the state of the system alarm (Alarm 3). This will have its *sunNemAlarmState* set to `steady(3)` when the watchdog is active and has not timed out.

If you change the state of Alarm 3 directly, it will not be possible to detect watchdog timeouts using this mechanism.

TABLE 1-16 Netra t1 Model 100/105 Containment Model

Physical Hierarchy/Model Description	SunNIM Class	Refer to
Netra t1 (UltraSPARC-IIi <*speed*> MHz) Chassis	NemChassis	
➥**Fan 1**	NemFan	FIGURE 1-1 Item 1
➥**Fan 1 Tachometer**	NemNumericSensor	
➥**Fan 2**	NemFan	FIGURE 1-1 Item 2
➥**Fan 2 Tachometer**	NemNumericSensor	
➥**Fan 3**	NemFan	FIGURE 1-1 Item 3
➥**Fan 3 Tachometer**	NemNumericSensor	
➥SCSI Adapter Card	NimEquipment	FIGURE 1-1 Item 15
➥Serial A/LOM	NimTerminationPoint	FIGURE 1-1 Item 4
➥Serial B	NimTerminationPoint	FIGURE 1-1 Item 5
➥SCSI	NimTerminationPoint	FIGURE 1-2 Item 6
➥PSU	NemPowerSupply	FIGURE 1-1 Item 7
➥**PSU Input A**	NemBinarySensor	
➥**PSU Input B**	NemBinarySensor	
➥Drive 0	NimEquipmentHolder	
➥<*disk0*>	NimCircuitPack	FIGURE 1-1 Item 8
➥Drive 1	NimEquipmentHolder	
➥<*disk1*>	NimCircuitPack	FIGURE 1-1 Item 9
➥CD-ROM Drive	NimEquipmentHolder	
➥<*cdrom*>	NimCircuitPack	FIGURE 1-1 Item 10
➥Motherboard	NimEquipment	FIGURE 1-1 Item 16
➥<*cpu*>	NimEquipment	
➥PCI Slot	NimEquipmentHolder	
➥<*PCIcard*>	NimCircuitPack	FIGURE 1-1 Item 11
➥LOMlite	NimEquipment	
➥**Alarm 1**	NemAlarmDevice	
➥**Alarm 2**	NemAlarmDevice	
➥**System Alarm**	NemAlarmDevice	
➥Watchdog	NemWatchdog	
➥Net 0	NimTerminationPoint	FIGURE 1-2 Item 12
➥Net 1	NimTerminationPoint	FIGURE 1-2 Item 13
➥**Fault LED**	NemAlarmDevice	FIGURE 1-2 Item 14
➥**Fault LED**	NemAlarmDevice	FIGURE 1-2 Item 14 FIGURE 1-3 Item 14

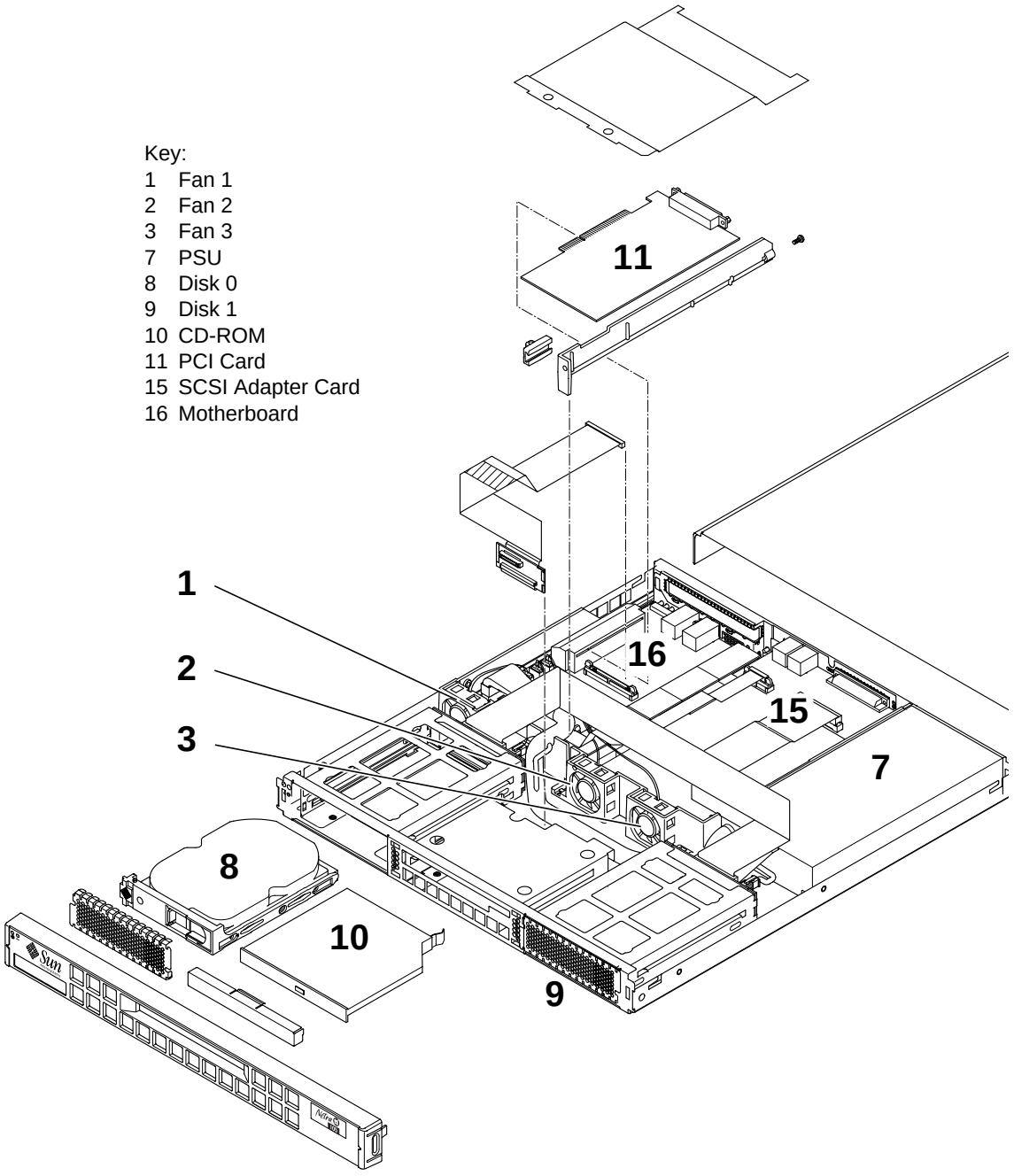


FIGURE 1-1 Netra t1 Model 100/105 Component Identification – Internal

Key:
4 Serial A/LOM
5 Serial B
6 SCSI
12 Net 0
13 Net 1
14 Fault LED

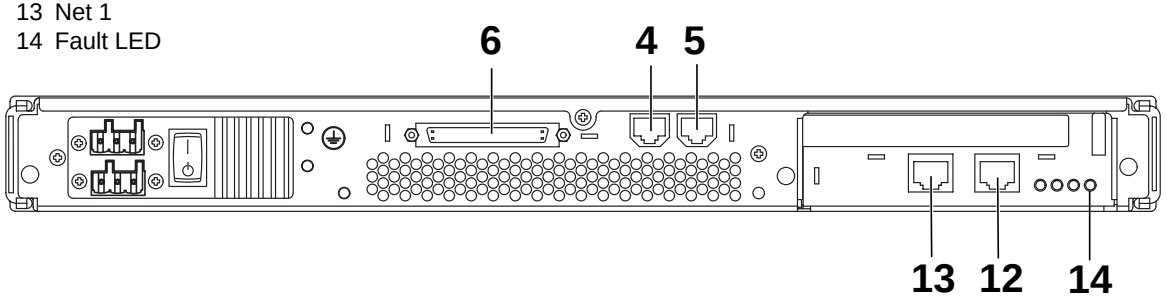


FIGURE 1-2 Netra t1 Model 100/105 Component Identification – Rear Panel

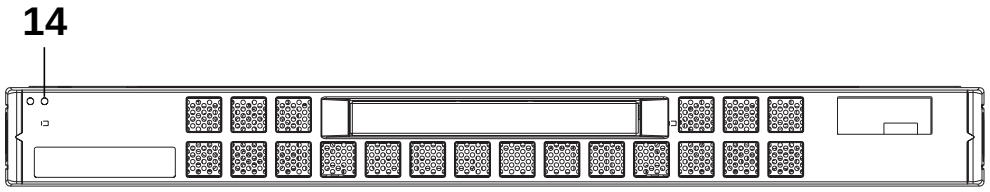


FIGURE 1-3 Netra t1 Model 100/105 Component Identification – Front Panel

CHAPTER 2

Netra t 1120 / 1125

This chapter provides the information available from the Sun Netra SNMP Management Agent 1.0 Management Interface on a Netra t 1120 and 1125.

The hardware resources presented by the SNMP agent for the Netra t 1120 / 1125 platform are shown in TABLE 2-17. This also illustrates the physical hierarchy of these resources. The table should be used in conjunction with FIGURE 2-1, FIGURE 2-2 and FIGURE 2-3 which identify the location of the hardware resources in the platform.

Each hardware resource in the table is shown together with the SunNIM class used to represent it.

In addition to static hardware resource characteristics, some hardware resources in the physical hierarchy have the capability to provide dynamic status information for the managed object in question. Hardware resources which can provide dynamic status information are shown in bold text in the table.

Note that the SCSI devices in the system are detected when the agent is started, but that subsequent insertion / removal of drives is not detected.

PCI cards are also detected when the agent is started, but they have no dynamic state associated with them.

The following describes the representation and monitoring of the hardware resources with dynamic behavior.

Fans

A fan is modeled using the NemFan class. The fans can be identified in the *entityMIB.entityPhysical.entPhysicalTable* by their *entPhysicalDescr* of 'Main Fan' and 'CPU Fan', or by the *entPhysicalName* of 'fan<*n*>' where <*n*> is 1 or 2 respectively.

Each fan contains a tachometer represented using the NemBinarySensor class. This is used to indicate whether the fan's speed is within the expected range. The tachometers are identified by their *entPhysicalDescr* of 'Main Fan Tachometer' and 'CPU Fan Tachometer', the search possibly accelerated by reference to the *entityMIB.entityPhysical.entPhysicalTable* to establish what is contained in a previously located fan.

The NemFan class used to represent the fans uses the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable* and
- *sunNemMIB.sunNemMIBObjects.sunNemFanTable*

The *sunNemFanVariableSpeed* attribute in the *sunNemFanTable* is set to `true(1)`. NemFan entries in the *entPhysicalTable* have an *entPhysicalClass* of `fan(7)`.

TABLE 2-1 Netra t 1120/1125 sunNemFanTable – Fans

sunNemFanEntry	
Variable	**Value**
sunNemFanVariableSpeed	`true(1)`

The NemBinarySensor class used to represent the fan tachometers is represented using the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemBinarySensorTable*

NemBinarySensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* is configured for the Netra t 1120/1125 fan tachometers as shown in TABLE 2-2:

TABLE 2-2 Netra t 1120/1125 sunNemSensorTable – Fan Tachometers

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`binary(1)`
sunNemSensorType	`tachometer(6)`
sunNemSensorLatency	<*undefined*>

The *sunNemBinarySensorTable* is configured as shown in TABLE 2-3:

TABLE 2-3 Netra t 1120/1125 sunNemBinarySensorTable – Fan Tachometers

sunNemNumericSensorEntry	
Variable	**Value**
sunNemBinarySensorCurrent	`true(1)` or `false(2)`
sunNemBinarySensorExpected	`true(1)`
sunNemBinarySensorInterpretTrue	Speed within expected range
sunNemBinarySensorInterpretFalse	Speed outside expected range

Fan Failures

The fan is considered to no longer be providing service if its reported speed falls outside the expected range. This is indicated by its tachometer's *sunNemBinarySensorCurrent* value.

The operability of the fan is represented using the *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable*, an extension to the *entPhysicalTable*, which contains the*sunNimEquipmentOperationalState*.

The fan will have its *sunNimEquipmentOperationalState* set to `disabled(1)` should its contained tachometer indicate a failure condition. This state will be restored to `enabled(2)` should the fan recover to an operational state. An SNMP *sunNimStateChange* trap is generated in response to an operational state change. A *sunNimAttributeChange* trap is also generated corresponding to the change in the *sunNemBinarySensorCurrent* value.

The *sunNimStateChange* trap will take the form shown in TABLE 2-4:

TABLE 2-4 Netra t 1120/1125 sunNimStateChange – Fan Failures

sunNimStateChange	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNimEquipmentTable.sunNimEquipmentEntry.sunNimEquipmentOperationalState.<*instance*>[1]
sunNimNotificationOldInteger	`disabled(1)` or `enabled(2)`
sunNimNotificationNewInteger	`disabled(1)` or `enabled(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed fan.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 2-5:

TABLE 2-5 Netra t 1120/1125 sunNimAttributeChangeInteger – Fan Failures

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemBinarySensorTable.sunNemBinarySensorEntry.sunNemBinarySensorCurrent.<*instance*>[1]
sunNimNotificationOldInteger	`true(1)` or `false(2)`
sunNimNotificationNewInteger	`true(1)` or `false(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed fan.

sunNimNotificationObject is a reference to the first column of the appropriate row of the *entPhysicalTable* in the ENTITY-MIB, and the *sunNimNotificationChangedOID* is a reference to the state attribute which has changed (in this case the *sunNimEquipmentOperationalState*). The *sunNimNotificationOldInteger* and *sunNimNotificationNewInteger* values represent the old and new integer enumerations of the changed state.

An alarm notification is also generated to indicate a fan failure, as shown in TABLE 2-6:

TABLE 2-6 Netra t 1120/1125 sunNimEnvironmentalAlarm – Fan Failures

sunNimEnvironmentalAlarm	
Variable	**Value**
sunNimNotificationEventId	*<unique numeric identifier>*
sunNimNotificationTime	*<date>* *<time>*
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.*<instance>*[1]
sunNimNotificationAdditionalInfo	`0`
sunNimNotificationAdditionalText	
sunNimNotificationPerceivedSeverity	`indeterminate(1)`
sunNimNotificationProbableCause	`coolingFanFailure(107)`
sunNimNotificationSpecificProblem	[Main|CPU] Fan speed below normal minimum threshold
sunNimNotificationRepairAction	`false(2)`

1. *<instance>* indicates the row of the *entPhysicalTable* associated with the failed fan.

Hence there are five mechanisms to detect fan status:

1. Poll *sunNemBinarySensorCurrent* of the fan's tachometer.
2. Poll *sunNimEquipmentOperationalState* of the fan.
3. Receiving a *sunNimStateChange* trap corresponding to the change of *sunNimEquipmentOperationalState* of the fan.
4. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemBinarySensorCurrent* of the fan's tachometer.
5. Receiving a *sunNimEnvironmentalAlarm* trap with a *sunNimNotificationProbableCause* of `coolingFanFailure(107)`. Note that this will only indicate failure and not any subsequent recovery.

Power Supplies

The power supply unit (PSU) is modeled using the NemPowerSupply class. The PSU can be identified by its *entPhysicalDescr* of 'PSU'.

The PSU contains voltage sensors modeled using the NemBinarySensor class. There are two sensors for monitoring the PSU inputs A/B. These are only of relevance in a DC system (Netra t 1120), as an AC system (Netra t 1125) will cease to operate should the input power be removed. The input voltage sensors can be identified by their *entPhysicalDescr* of 'PSU Input A' or 'PSU Input B'. The search can be accelerated by reference to the to *entPhysicalContainsTable* to establish what is contained in the previously located PSU.

The PSU is always reported as being operational as this is implicit in the agent being able to respond. It is thus not possible to report the failure of a PSU.

The NemPowerSupply class used to represent the PSU uses the *sunNimEquipmentTable* extension to the *entPhysicalTable.* NemPowerSupply entries in the *entPhysicalTable* have an *entPhysicalClass* of `powerSupply(6)`.

The NemBinarySensor class used to represent the voltage sensors is represented using the following extensions to the *entPhysicalTable*:

- *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemBinarySensorTable*

NemBinarySensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* is configured for the PSU voltage sensors as shown in TABLE 2-7:

TABLE 2-7 Netra t 1120/1125 sunNemSensorTable – PSU Voltage Sensors

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`binary(1)`
sunNemSensorType	`voltage(4)`
sunNemSensorLatency	*<undefined>*

The *sunNemBinarySensorTable* is configured as shown in TABLE 2-8:

TABLE 2-8 Netra t 1120/1125 sunNemBinarySensorTable – PSU Voltage Sensors

sunNemBinarySensorEntry	
Variable	**Value**
sunNemBinarySensorCurrent	`true(1) or false(2)`
sunNemBinarySensorExpected	`true(1)`
sunNemBinarySensorInterpretTrue	Voltage within expected range
sunNemBinarySensorInterpretFalse	Voltage outside expected range

PSU Input Failures

The detection of PSU input failures is only applicable to systems with DC power supplies.

Should either of the A or B DC inputs to the PSU fall below the minimum acceptable level, the corresponding voltage sensor's state, reported as *sunNemBinarySensorCurrent* in the corresponding *sunNemBinarySensorTable* entry will change from `true(1)` to `false(2)`. Recovery of the input will result in the value returning to `true(1)`. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The A/B DC supply input statuses are indicated by LEDs on the from panel, and a change in state of one of these LEDs will also result in a *sunNimAttributeChangeInteger* trap – see “Alarms/Supply A/B Monitor State Changes” on page 25.

The *sunNimAttributeChangeInteger* trap will take the form as shown in TABLE 2-9:

TABLE 2-9 Netra t 1120/1125 sunNimAttributeChangeInteger – PSU Failures

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	`<unique numeric identifier>`
sunNimNotificationTime	`<date> <time>`
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemBinarySensorTable.sunNemBinarySensorEntry.sunNemBinarySensorCurrent.<*instance*>[1]
sunNimNotificationOldInteger	`true(1)` or `false(2)`
sunNimNotificationNewInteger	`true(1)` or `false(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed PSU input.

Hence there are three mechanisms to detect PSU input status:

1. Poll *sunNemBinarySensorCurrent* of the PSU's inputs.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemBinarySensorCurrent*.
3. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemAlarmState* (see below).

Alarms/Supply A/B Monitors

The alarms/Supply A/B monitors are modeled using the NemAlarmDevice class. Alarms 1 through 3 are associated with front panel LEDs, Alarm 3 being labeled 'System'. On a DC system (Netra t 1120) there are also two LEDs on the front panel indicating the state of the A/B DC inputs to the PSU. The LEDs are identified by their *entPhysicalDescr* of 'Alarm 1 LED', 'Alarm 2 LED', 'System LED', 'Supply A LED' or 'Supply B LED'.

The NemAlarmDevice class used to represent the LEDs uses the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemAlarmTable*

NemAlarmDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the LEDs as shown in TABLE 2-10:

TABLE 2-10 Netra t 1120/1125 sunNemPhysicalTable – LEDs

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`alarm(2)`

The *sunNemAlarmTable* is configured as shown in TABLE 2-11:

TABLE 2-11 Netra t 1120/1125 sunNemAlarmTable – LEDs

sunNemAlarmTable	
Variable	**Value**
sunNemAlarmType	`visible(3)`
sunNemAlarmState	`off(2)` or `steady(3)`

Note that the system alarm (Alarm 3) is modeled according to the convention that any associated alarm indicator is ‘on’ when the system is active with the ASR watchdog running correctly, and ‘off’ otherwise. Its state can also be changed manually. This is the reverse of the state reported by the alarms utilities, which report the status of Alarm 3 directly rather than that of the derived system indicator.

Alarms/Supply A/B Monitor State Changes

Should the state of any of the LEDs change, the *sunNemAlarmState* in the corresponding *sunNemAlarmTable* entry will change. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 2-12:

TABLE 2-12 Netra t 1120/1125 sunNimAttributeChangeInteger – LED State Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	*<unique numeric identifier>*
sunNimNotificationTime	*<date> <time>*
sunNimNotificationObject	*entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<instance>*[1]
sunNimNotificationChangedOID	sunNemAlarmTable.sunNemAlarmEntry.sunNemAlarmState.*<instance>*[1]
sunNimNotificationOldInteger	`off(2)` or `steady(3)`
sunNimNotificationNewInteger	`off(2)` or `steady(3)`

1. *<instance>* indicates the row of the *entPhysicalTable* associated with the alarm device.

Hence there are two mechanisms to detect LED state changes:

1. Poll the LED's *sunNemAlarmState*.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemAlarmState*.

ASR Watchdog

The ASR watchdog is modeled using the NemWatchdogDevice class. It can be identified by the *entPhysicalDescr* of 'Watchdog'. It is only modeled if it is enabled. If it is disabled, once the agent is running it will be removed from the model. Similarly, it will be added to the model if it is enabled.

The NemWatchdogDevice is represented using the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemWatchdogTable*

NemAlarmDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the ASR watchdog as shown in TABLE 2-13:

TABLE 2-13 Netra t 1120/1125 sunNemPhysicalTable – ASR Watchdog

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`watchdog(3)`

The *sunNemWatchdogTable* is configured as shown in TABLE 2-14:

TABLE 2-14 Netra t 1120/1125 sunNemWatchdogTable – ASR Watchdog

sunNemWatchdogTable	
Variable	**Value**
sunNemWatchdogAction	`statusOnly(1)` or `systemReset(3)`
sunNemWatchdogTimeout	<*variable*> milliseconds

ASR Watchdog Changes

Should either of the attributes in the *sunNemWatchdogTable* change, an SNMP *sunNimAttributeChange* trap is generated.

The *sunNimAttributeChangeInteger* trap will take one of the forms shown in TABLE 2-15 and TABLE 2-16:

TABLE 2-15 Netra t 1120/1125 sunNimAttributeChangeInteger – ASR Watchdog Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogAction.<*instance*>[1]
sunNimNotificationOldInteger	`statusOnly(1)` or `systemReset(3)`
sunNimNotificationNewInteger	`statusOnly(1)` or `systemReset(3)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog.

TABLE 2-16 Netra t 1120/1125 sunNimAttributeChangeInteger – ASR Watchdog Changes

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	*entPhysicalTable.entPhysicalEntry.entPhysicalIndex.*<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogTimeout.<*instance*>[1]
sunNimNotificationOldInteger	<*variable*>
sunNimNotificationNewInteger	<*variable*>

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog.

where <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog.

Hence there are two mechanisms to detect such attribute changes:

1. Poll the appropriate ASR watchdog attribute.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of the appropriate ASR watchdog attribute.

The intended mode of operation is that a watchdog timer expiry is configured to reset the system. Should the *sunNemWatchdogAction* attribute be set to `statusOnly(1)`, the state of the watchdog can be monitored by observing the state of the system alarm (Alarm 3). This will have its *sunNemAlarmState* set to `steady(3)` when the watchdog is active and has not timed out.

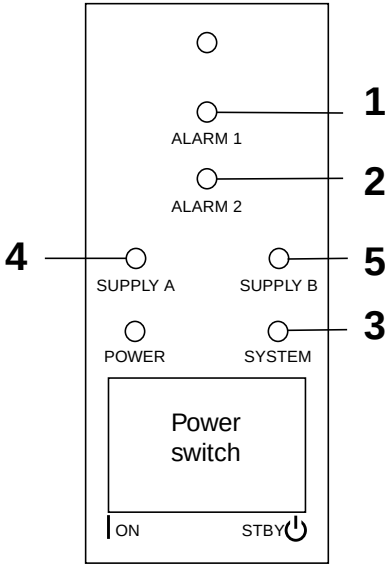
If you change the state of Alarm 3 directly, it will not be possible to detect watchdog timeouts using this mechanism.

TABLE 2-17 Netra t 1120/1125 Containment Model

Physical Hierarchy/Model Description	NIM/NEM Class	Refer to
Netra t 1120/1125 (2 x UltraSPARC-II <*speed*> MHz) Chassis	NimChassis	
➥**Alarm 1 LED**	NemAlarmDevice	FIGURE 2-1 Item 1
➥**Alarm 2 LED**	NemAlarmDevice	FIGURE 2-1 Item 2
➥**System LED**	NemAlarmDevice	FIGURE 2-1 Item 3
➥**Supply A LED**	NemAlarmDevice	FIGURE 2-1 Item 4
➥**Supply B LED**	NemAlarmDevice	FIGURE 2-1 Item 5
➥**Main Fan**	NemFan	FIGURE 2-3 Item 6
➥**Main Fan Tachometer**	NemBinarySensor	
➥SCSI Subassembly	NimEquipment	
➥Disk 0	NimEquipmentHolder	
➥<*disk0*>	NimCircuitPack	FIGURE 2-3 Item 7
➥Disk 1	NimEquipmentHolder	
➥<*disk1*>	NimCircuitPack	FIGURE 2-3 Item 8
➥Drive 5	NimEquipmentHolder	
➥<*cdrom*>	NimCircuitPack	FIGURE 2-3 Item 9
➥Drive 6	NimEquipmentHolder	
➥<*tape*>	NimCircuitPack	FIGURE 2-3 Item 10
➥PSU	NemPower Supply	
➥**PSU Input A**	NemBinary Sensor	
➥**PSU Input B**	NemBinary Sensor	
➥Motherboard	NimEquipment	FIGURE 2-3 Item 26
➥CPU Slot 0	NimEquipmentHolder	
➥CPU Module 0	NimCircuitPack	
➥<*cpu0*>	NimEquipment	FIGURE 2-3 Item 11
➥CPU Slot 1	NimEquipmentHolder	
➥CPU Module 1	NimCircuitPack	
➥<*cpu1*>	NimEquipment	FIGURE 2-3 Item 12
➥Processor Mounting Bracket	NimEquipment	
➥**CPU Fan**	NemFan	FIGURE 2-3 Item 13
➥**CPU Fan Tachometer**	NemBinarySensor	
➥EBus Slot	NimEquipmentHolder	
➥Alarms Card	NimCircuitPack	FIGURE 2-3 Item 14
➥Alarms Service Port Connector	NimTerminationPoint	FIGURE 2-2 Item 15
➥Alarm 1	NemAlarmDevice	
➥Alarm 2	NemAlarmDevice	
➥System (Alarm 3)	NemAlarmDevice	
➥Watchdog	NemWatchdog	
➥PCI 66 Slot 1	NimEquipmentHolder	
➥<*PCIcard1*>	NimCircuitPack	FIGURE 2-2 Item 19

TABLE 2-17 Netra t 1120/1125 Containment Model *(Continued)*

Physical Hierarchy/Model Description	NIM/NEM Class	Refer to
➥PCI Slot 2	NimEquipmentHolder	
➥*<PCIcard2>*		FIGURE 2-2 Item 18
➥PCI Slot 3	NimEquipmentHolder	
➥*<PCIcard3>*	NimCircuitPack	FIGURE 2-2 Item 17
➥PCI Slot 4	NimEquipmentHolder	
➥*<PCIcard4>*	NimCircuitPack	FIGURE 2-2 Item 16
➥TPE	NimTerminationPoint	FIGURE 2-2 Item 20
➥MII	NimTerminationPoint	FIGURE 2-2 Item 21
➥Serial A	NimTerminationPoint	FIGURE 2-2 Item 22
➥Serial B	NimTerminationPoint	FIGURE 2-2 Item 23
➥Parallel	NimTerminationPoint	FIGURE 2-2 Item 24
➥SCSI	NimTerminationPoint	FIGURE 2-2 Item 25



Key:
1 Alarm 1 LED
2 Alarm 2 LED
3 System LED
4 Supply A LED (not Netra t 1125)
5 Supply B LED (not Netra t 1125)

FIGURE 2-1 Netra t 1120/1125 Component Identification – Front Panel

Key:
15 Alarms Port
16 PCI 1
17 PCI 2
18 PCI 3
19 PCI 4
20 TPE
21 MII
22 Serial A
23 Serial B
24 Parallel
25 SCSI

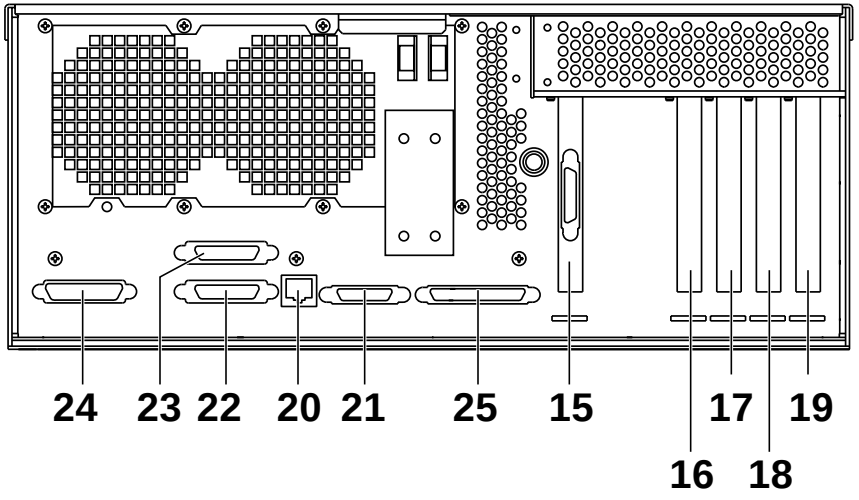


FIGURE 2-2 Netra t 1120/1125 Component Identification – Rear Panel

Key:
6 Main Fan
7 Disk 0
8 Disk 1 (if fitted—not shown)
9 CD-ROM
10 Tape
11 CPU 0
12 CPU 1
13 CPU Fan
14 Alarms Card
26 Motherboard

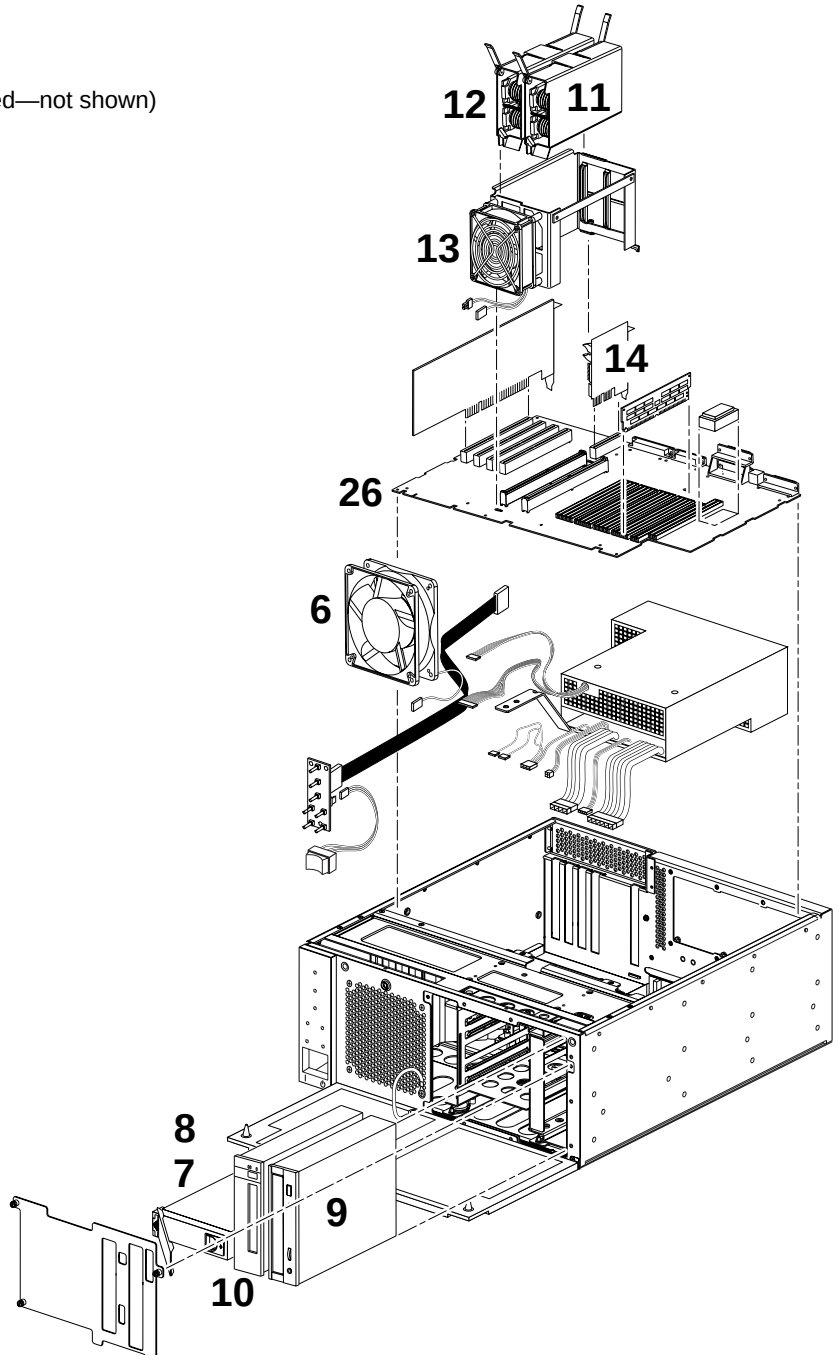


FIGURE 2-3 Netra t 1120/1125 Component Identification – Internal

CHAPTER 3

Netra t 1400 / 1405

This chapter provides the information available from the Sun Netra SNMP Management Agent 1.0 Management Interface on a Netra t 1400 and 1405.

The hardware resources presented by the SNMP agent for the Netra t 1400 / 1405 platform are shown in TABLE 3-19. It also illustrates the physical hierarchy of these resources. The table should be used in conjunction with FIGURE 3-1, FIGURE 3-2 and FIGURE 3-3 which identify the location of the hardware resources in the platform.

Each hardware resource in the table is shown together with the SunNIM class used to represent it.

In addition to static hardware resource characteristics, some hardware resources in the physical hierarchy have the capability to provide dynamic status information for the managed object in question. Hardware resources which can provide dynamic status information are shown in bold text in the table.

Note that the SCSI devices in the system are detected when the agent is started, but that subsequent insertion / removal of drives is not detected.

PCI cards are also detected when the agent is started, but they have no dynamic state associated with them.

The following describes the representation and monitoring of the hardware resources with dynamic behavior.

Fans

A fan is modeled using the NemFan class. The fans are identified in the *entityMIB.entityPhysical.entPhysicalTable* by their *entPhysicalDescr* of 'System Fan <n>', or by the *entPhysicalName* of 'fan<n>', where <n> is a number in the range 1 through 4.

Each fan contains a tachometer represented using the NemNumericSensor class. This is used to indicate the current speed of the fan expressed in revolutions per minute (RPM). The tachometers are identified by their *entPhysicalDescr* of 'Fan <n> Tachometer' where <n> is a number in the range 1 through 4 corresponding to the fan being monitored, the search possibly accelerated by reference to the in the *entityMIB.entityPhysical.entPhysicalTable* to establish what is contained in a previously located fan.

The NemFan class used to represent the fans uses the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable* and
- *sunNemMIB.sunNemMIBObjects.sunNemFanTable*

The *sunNemFanVariableSpeed* attribute in the *sunNemFanTable* is set to `true(1)`. NemFan entries in the *entPhysicalTable* have an *entPhysicalClass* of `fan(7)`.

TABLE 3-1 Netra t 1400/1405 sunNemFanTable – Fans

sunNemFanEntry	
Variable	**Value**
sunNemFanVariableSpeed	`true(1)`

The NemNumericSensor class used to represent the fan tachometers is represented using the followingextensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemNumericSensorTable*

NemNumericSensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* is configured for the fan tachometers as shown in TABLE 3-2:

TABLE 3-2 Netra t 1400/1405 sunNemSensorTable – Fan Tachometers

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`numeric(2)`
sunNemSensorType	`tachometer(6)`
sunNemSensorLatency	*<undefined>*

The *sunNemNumericSensorTable* is configured as shown in TABLE 3-3:

TABLE 3-3 Netra t 1400/1405 sunNemNumericSensorTable – Fan Tachometers

sunNemNumericSensorEntry	
Variable	**Value**
sunNemNumericSensorBaseUnits	`revolutions(39)`
sunNemNumericSensorExponent	`0`
sunNemNumericSensorRateUnits	`perMinute(5)`
sunNemNumericSensorCurrent	*<variable>*
sunNemNumericSensorNormalMin	*<fan dependent>*
sunNemNumericSensorNormalMax	*<undefined>*
sunNemNumericSensorAccuracy	*<undefined>*

Note that the value for *sunNemNumericSensorNormalMin* varies according to fan specification.

Fan Failures

The fan is considered to no longer be providing service if its reported speed falls below an expected normal minimum as indicated by its tachometer's *sunNemNumericSensorNormalMin* value.

The operability of the fan is represented using the *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable*, an extension to the *entPhysicalTable*, which contains the *sunNimEquipmentOperationalState*.

The fan will have its *sunNimEquipmentOperationalState* set to `disabled(1)` should its contained tachometer indicate a failure condition. This state will be restored to `enabled(2)` should the fan recover to an operational state. An SNMP *sunNimStateChange* trap is generated in response to an operational state change. When a fan first fails the fault LEDs are set to flash. Consequently there will be additional SNMP *sunNimAttributeChange* traps generated as a side effect of the fan failure.

The *sunNimStateChange* trap will take the form shown in TABLE 3-4:

TABLE 3-4 Netra t 1400/1405 sunNimStateChange – Fan Failure

sunNimStateChange	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNimEquipmentTable.sunNimEquipmentEntry.sunNimEquipmentOperationalState.<*instance*>[1]
sunNimNotificationOldInteger	`disabled(1)` or `enabled(2)`
sunNimNotificationNewInteger	`disabled(1)` or `enabled(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed fan.

sunNimNotificationObject is a reference to the first column of the appropriate row of the *entPhysicalTable* in the ENTITY-MIB, and the *sunNimNotificationChangedOID* is a reference to the state attribute which has changed (in this case the *sunNimEquipmentOperationalState*). The *sunNimNotificationOldInteger* and *sunNimNotificationNewInteger* values represent the old and new integer enumerations of the changed state.

An alarm notification is also generated to indicate a fan failure, as shown in TABLE 3-5:

TABLE 3-5 Netra t 1400/1405 sunNimEnvironmentalAlarm – Fan Failure

sunNimEnvironmentalAlarm	
Variable	**Value**
sunNimNotificationEventId	*<unique numeric identifier>*
sunNimNotificationTime	*<date> <time>*
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.*<instance>*[1]
sunNimNotificationAdditionalInfo	`0`
sunNimNotificationAdditionalText	
sunNimNotificationPerceivedSeverity	`indeterminate(1)`
sunNimNotificationProbableCause	`coolingFanFailure(107)`
sunNimNotificationSpecificProblem	Fan *<n>* speed below normal minimum threshold
sunNimNotificationRepairAction	`false(2)`

1. *<instance>* indicates the row of the *entPhysicalTable* associated with the failed fan.

Note – Attribute Value Change events (traps) are not generated for changes to the current reported tachometer reading, s*unNemNumericSensorCurrent*, as it is not, in itself, of much significance, and potentially changes frequently.

Hence there are four mechanisms to detect fan status:

1. Polling *sunNemNumericSensorCurrent* and *sunNemNumericSensorNormalMin* of the fan's tachometer, comparing their values.
2. Polling *sunNimEquipmentOperationalState* of the fan.
3. Receiving a *sunNimStateChange* trap corresponding to the change of *sunNimEquipmentOperationalState* of the fan.
4. Receiving a *sunNimEnvironmentalAlarm* trap with a *sunNimNotificationProbableCause* of `coolingFanFailure(107)`. Note that this will only indicate failure and not any subsequent recovery.

PSUs

The power supply units (PSUs) are modeled using the NemPowerSupply class. The PSUs are identified by its *entPhysicalDescr* of 'PSU <*n*>' where <*n*> is a number in the range 1 through 3.

Each PSU contains voltage sensors modeled using the NemBinarySensor class. There are two sensors for monitoring the PSU inputs A/B. In a DC system (Netra t 1400); these sensors independently monitor the A and B DC supplies. In an AC system (Netra t 1405) both sensors reflect the state of the AC input. The input voltage sensors is identified by their *entPhysicalDescr* of 'PSU <*n*> Input A' or 'PSU <*n*> Input B'. The search can be accelerated by reference to the *entPhysicalContainsTable* to establish what is contained in the previously-located PSU.

Each PSU also contains an output voltage sensor which indicates if the supply is producing output. This is identified by its *entPhysicalDescr* of 'PSU <*n*> Output', the search possibly accelerated by reference to the *entPhysicalContainsTable* to establish what is contained in the previously located PSU.

The NemPowerSupply class used to represent the PSU uses the *sunNimEquipmentTable* extension to the *entPhysicalTable*. NemPowerSupply entries in the *entPhysicalTable* have an *entPhysicalClass* of `powerSupply(6)`.

The NemBinarySensor class used to represent the voltage sensors uses the following extensions to the *entPhysicalTable*:

- *sunNimMIB.sunNimMIBObjects.sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemSensorTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemBinarySensorTable*

NemBinarySensor entries in the *entPhysicalTable* have an *entPhysicalClass* of `sensor(8)`. The *sunNemSensorTable* is configured for the PSU voltage sensors as shown in TABLE 3-6:

TABLE 3-6 Netra t 1400/1405 sunNemSensorTable – PSU Voltage Sensors

sunNemSensorEntry	
Variable	**Value**
sunNemSensorClass	`binary(1)`
sunNemSensorType	`voltage(4)`
sunNemSensorLatency	<*undefined*>

The *sunNemBinarySensorTable* is configured as shown in TABLE 3-7:

TABLE 3-7 Netra t 1400/1405 sunNemBinarySensorTable – PSU Voltage Sensors

sunNemBinarySensorEntry	
Variable	**Value**
sunNemBinarySensorCurrent	`true(1)` or `false(2)`
sunNemBinarySensorExpected	`true(1)`
sunNemBinarySensorInterpretTrue	Voltage within expected range
sunNemBinarySensorInterpretFalse	Voltage outside expected range

PSU Input Failures

Should either of the A or B DC inputs to the PSU fall below the minimum acceptable level, the corresponding voltage sensor's state, reported as *sunNemBinarySensorCurrent* in the corresponding *sunNemBinarySensorTable* entry will change from `true(1)` to `false(2)`. Recovery of the input will result in the value returning to `true(1)`. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 3-8:

TABLE 3-8 Netra t 1400/1405 sunNimAttributeChangeInteger – PSU Input Failure

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemBinarySensorTable.sunNemBinarySensorEntry.sunNemBinarySensorCurrent.<*instance*>[1]
sunNimNotificationOldInteger	`true(1)` or `false(2)`
sunNimNotificationNewInteger	`true(1)` or `false(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed PSU input.

Hence there are two mechanisms to detect PSU input status:

1. Poll *sunNemBinarySensorCurrent* of the PSU's inputs.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemBinarySensorCurrent*.

PSU Output Failures

Should the PSU output not fall within the acceptable range, the output voltage sensor's state, reported as *sunNemBinarySensorCurrent* in the corresponding *sunNemBinarySensorTable* entry, will change from `true(1)` to `false(2)`. Recovery of the PSU's output will result in the value returning to `true(1)`. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 3-9:

TABLE 3-9 Netra t 1400/1405 sunNimAttributeChangeInteger – PSU Output Failure

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemBinarySensorTable.sunNemBinarySensorEntry.sunNemBinarySensorCurrent.<*instance*>[1]
sunNimNotificationOldInteger	`true(1)` or `false(2)`
sunNimNotificationNewInteger	`true(1)` or `false(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed PSU output.

Should the PSU output voltage not be within the acceptable range, it is indicative of the PSU failing to provide service. The PSU will have its *sunNimEquipmentOperationalState* set to `disabled(1)` should this condition arise. This state will be restored to `enabled(2)` should it recover to an operational state. An SNMP *sunNimStateChange* trap is generated in response to an operational state change. A PSU can be considered to have failed if its output is not within the acceptable range while wither input A or B is present. When a PSU first fails the fault LED is set to flash. Consequently there will be additional SNMP *sunNimAttributeChange* traps generated as a side effect of the PSU failure.

The *sunNimStateChange* trap will take the form shown in TABLE 3-10:

TABLE 3-10 Netra t 1400/1405 sunNimStateChange – PSU Output Failure

sunNimStateChange	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNimEquipmentTable.sunNimEquipmentEntry.sunNimEquipmentOperationalState.<*instance*>[1]
sunNimNotificationOldInteger	`disabled(1)` or `enabled(2)`
sunNimNotificationNewInteger	`disabled(1)` or `enabled(2)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the failed PSU output.

sunNimNotificationObject is a reference to the first column of the appropriate row of the *entPhysicalTable* in the ENTITY-MIB, and the *sunNimNotificationChangedOID* is a reference to the state attribute which has changed (in this case the *sunNimEquipmentOperationalState*). The *sunNimNotificationOldInteger* and *sunNimNotificationNewInteger* values represent the old and new integer enumerations of the changed state.

A *sunNimEquipmentAlarm* trap is also generated to indicate a PSU failure, as shown in TABLE 3-11:

TABLE 3-11 Netra t 1400/1405 sunNimEquipmentAlarm – PSU Output Failure

sunNimEquipmentAlarm	
Variable	**Value**
sunNimNotificationEventId	*<unique numeric identifier>*
sunNimNotificationTime	*<date> <time>*
sunNimNotificationObject	*entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<instance>*[1]
sunNimNotificationAdditionalInfo	`0`
sunNimNotificationAdditionalText	
sunNimNotificationPerceivedSeverity	`indeterminate(1)`
sunNimNotificationProbableCause	`powerProblem(58)`
sunNimNotificationSpecificProblem	PSU *<n>* output voltage outside expected range
sunNimNotificationRepairAction	`false(2)`

1. *<instance>* indicates the row of the *entPhysicalTable* associated with the failed PSU ouyput.

Hence there are five mechanisms to detect PSU output status:

1. Poll *sunNemBinarySensorCurrent* of the PSU's output.
2. Poll *sunNimEquipmentOperationalState* of the PSU.
3. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemBinarySensorCurrent* of the PSU output.
4. Receiving a *sunNimStateChange* trap corresponding to the change of *sunNimEquipmentOperationalState* of the PSU.
5. Receiving a *sunNimEquipmentAlarm* trap with a *sunNimNotificationProbableCause* of `powerProblem(58)`. Note that this will only indicate failure and not any subsequent recovery.

LOMlite Alarms and LEDs

The LOMlite alarms and LEDs are modeled using the NemAlarmDevice class. The alarms provide both LED indicators on the chassis front panel, and switch outputs via the LOMlite card's Alarms Service Port connector. Each of these are modeled separately. The fault indicator is associated with a LED on the front panel. The alarms/fault LEDs are identified by their *entPhysicalDescr* of 'Alarm 1 LED', 'Alarm 2 LED', 'System LED', or 'Fault LED'. The alarm switch outputs are identified by their *entPhysicalDescr* of 'Alarm 1', 'Alarm 2', or 'System Alarm'. On a DC system (Netra t 1400) there are also two LEDs on the front panel indicating the state of the A/B DC inputs to the PSUs. The LEDs are identified by their *entPhysicalDescr* of 'Supply A LED' or 'Supply B LED'.

The NemAlarmDevice class used to represent the alarms/LEDs uses the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemAlarmTable*

NemAlarmDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the LOMlite alarms/LEDs as shown in TABLE 3-12:

TABLE 3-12 Netra t 1400/1405 sunNemPhysicalTable – LOMlite Alarms and LEDs

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`alarm(2)`

The *sunNemAlarmTable* is configured as shown in TABLE 3-13:

TABLE 3-13 Netra t 1400/1405 sunNemAlarmTable – LOMlite Alarms and LEDs

sunNemAlarmTable	
Variable	**Value**
sunNemAlarmType	`switch(1)` for the alarm switch outputs, `visible(3)` for the LEDs
sunNemAlarmState	`off(2)`, `steady(3)` or `alternating(4)`

Note that the system alarm (Alarm 3) is modeled according to the convention that any associated alarm indicator is ‘on’ when the system is active with the ASR watchdog running correctly, and ‘off’ otherwise. Its state can also be changed manually. This is the reverse of the state reported by the alarms utilities, which report trhe status of Alarm 3 directly rather than that of the derived system indicator.

LOMlite Alarms / Fault LEDs State Changes

Should the state of any of the alarms, or the LEDs change, the *sunNemAlarmState* in the corresponding *sunNemAlarmTable* entry will change. An SNMP *sunNimAttributeChange* trap is generated in response to this attribute change.

The *sunNimAttributeChangeInteger* trap will take the form shown in TABLE 3-14:

TABLE 3-14 Netra t 1400 / 1405 sunNimAttributeChangeInteger – LOMlite Alarms and LEDs

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemAlarmTable.sunNemAlarmEntry.sunNemAlarmState.<*instance*>[1]
sunNimNotificationOldInteger	`off(2)`, `steady(3)` or `alternating(4)`
sunNimNotificationNewInteger	`off(2)`, `steady(3)` or `alternating(4)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the alarm device.

Hence there are two mechanisms to detect alarm / fault LED state changes:

1. Poll the alarm / LED's *sunNemAlarmState*.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of *sunNemAlarmState*.

ASR Watchdog

The ASR watchdog is modeled using the NemWatchdogDevice class. It is identified by the *entPhysicalDescr* of 'Watchdog'. It is only modeled if it is enabled. If it is disabled, once the agent is running it will be removed from the model. Similarly, it will be added to the model if it is enabled.

The NemWatchdogDevice is represented using the following extensions to the *entPhysicalTable*:

- *sunNimEquipmentTable*,
- *sunNemMIB.sunNemMIBObjects.sunNemPhysicalTable*, and
- *sunNemMIB.sunNemMIBObjects.sunNemWatchdogTable*

NemWatchdogDevice entries in the *entPhysicalTable* have an *entPhysicalClass* of `other(1)`.

The *sunNemPhysicalTable* is configured for the Netra t 1400/1405 ASR watchdog as shown in TABLE 3-15 and TABLE 3-16:

TABLE 3-15 Netra t 1400/1405 sunNemPhysicalTable – ASR Watchdog

sunNemPhysicalTable	
Variable	**Value**
sunNemPhysicalClass	`watchdog(3)`

The *sunNemWatchdogTable* is configured as shown in TABLE 3-16:

TABLE 3-16 Netra t 1400/1405 sunNemWatchdogTable – ASR Watchdog

sunNemWatchdogTable	
Variable	**Value**
sunNemWatchdogAction	`statusOnly(1)` or `systemReset(3)`
sunNemWatchdogTimeout	<*variable*>

ASR Watchdog Changes

Should either of the attributes in the *sunNemWatchdogTable* change, an SNMP *sunNimAttributeChange* trap is generated.

The *sunNimAttributeChangeInteger* trap will take one of the forms shown in TABLE 3-17 and TABLE 3-18:

TABLE 3-17 Netra t 1400/1405 sunNimAttributeChangeInteger – ASR Watchdog

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogAction.<*instance*>[1]
sunNimNotificationOldInteger	`statusOnly(1)` or `systemReset(3)`
sunNimNotificationNewInteger	`statusOnly(1)` or `systemReset(3)`

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog.

TABLE 3-18 Netra t 1400/1405 sunNimAttributeChangeInteger – ASR Watchdog

sunNimAttributeChangeInteger	
Variable	**Value**
sunNimNotificationEventId	<*unique numeric identifier*>
sunNimNotificationTime	<*date*> <*time*>
sunNimNotificationObject	entPhysicalTable.entPhysicalEntry.entPhysicalIndex.<*instance*>[1]
sunNimNotificationChangedOID	sunNemWatchdogTable.sunNemWatchdogEntry.sunNemWatchdogTimeout.<*instance*>[1]
sunNimNotificationOldInteger	<*variable*>
sunNimNotificationNewInteger	<*variable*>

1. <*instance*> indicates the row of the *entPhysicalTable* associated with the ASR watchdog.

Hence there are two mechanisms to detect such attribute changes:

1. Poll the appropriate ASR watchdog attribute.
2. Receiving a *sunNimAttributeChange* trap corresponding to the change of the appropriate ASR watchdog attribute.

The intended mode of operation is that a watchdog timer expiry is configured to reset the system. Should the *sunNemWatchdogAction* attribute be set to `statusOnly(1)`, the state of the watchdog can be monitored by observing the state of the system alarm (Alarm3). This will have its *sunNemAlarmState* set to `steady(3)` when the watchdog is active and has not timed out.

If you change the state of Alarm 3 directly, it will not be possible to detect watchdog timeouts using this mechanism.

TABLE 3-19 Netra t 1400/1405 Containment Model

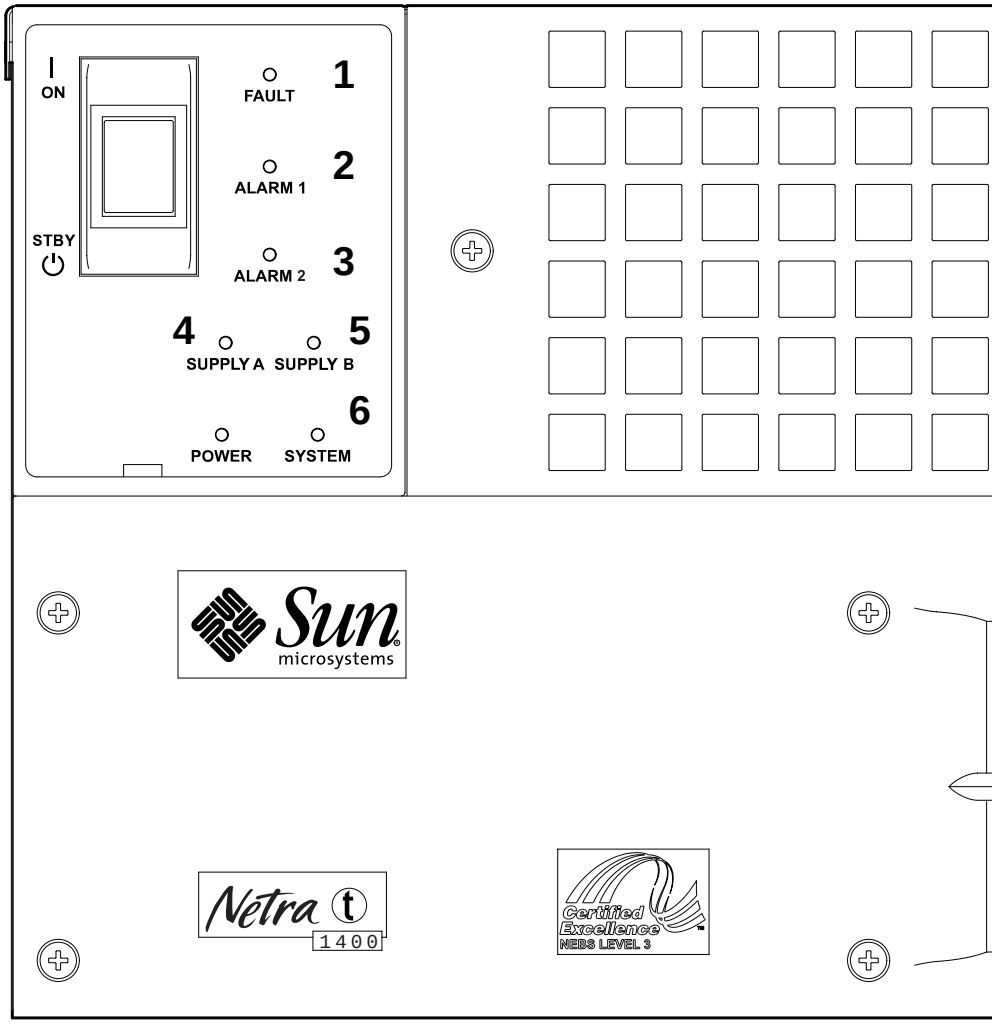
Ohysical Hierarchy/Model Description	NIM/NEM Class	Refer to
Netra t 1400/1405 (4 x UltraSPARC-II <*speed*> MHz) Chassis	NimChassis	
➥ **Fault LED**	NemAlarmDevice	FIGURE 3-1 Item 1
➥ **Alarm 1 LED**	NemAlarmDevice	FIGURE 3-1 Item 2
➥ **Alarm 2 LED**	NemAlarmDevice	FIGURE 3-1 Item 3
➥ **System LED**	NemAlarmDevice	FIGURE 3-1 Item 4
➥ **Supply A LED**	NemAlarmDevice	FIGURE 3-1 Item 5
➥ **Supply B LED**	NemAlarmDevice	FIGURE 3-1 Item 6
➥ System Fan Assembly	NimEquipment	
➥ **System Fan 1**	NemFan	FIGURE 3-2 Item 7
➥ **Fan 1 Tachometer**	NemNumericSensor	
➥ **System Fan 2**	NemFan	FIGURE 3-2 Item 8
➥ **Fan 2 Tachometer**	NemNumericSensor	
➥ Disk Bay Assembly	NimEquipment	
➥ Disk 0	NimEquipmentHolder	FIGURE 3-2 Item 9
➥ <*disk0*>	NimCircuitPack	
➥ Disk 1	NimEquipmentHolder	FIGURE 3-2 Item 10
➥ <*disk1*>	NimCircuitPack	
➥ Disk 2	NimEquipmentHolder	FIGURE 3-2 Item 11
➥ <*disk2*>	NimCircuitPack	
➥ Disk 3	NimEquipmentHolder	FIGURE 3-2 Item 12
➥ <*disk3*>	NimCircuitPack	
➥ Removable Media Bay Assembly	NimEquipment	
➥ Drive 4	NimEquipmentHolder	
➥ <*cdrom*>	NimCircuitPack	FIGURE 3-2 Item 13
➥ Drive 6	NimEquipmentHolder	
➥ <*tape*>	NimCircuitPack	FIGURE 3-2 Item 14
➥ Power subframe and power distribution board assembly	NimEquipment	
➥ PSU Slot 1	NimEquipmentHolder	
➥ PSU 1	NimCircuitPack	
➥ **Power Supply 1**	NemPower Supply	FIGURE 3-3 Item 15
➥ **PSU 1 Input A**	NemBinary Sensor	
➥ **PSU 1 Input B**	NemBinary Sensor	
➥ **PSU 1 Output**	NemBinary Sensor	
➥ PSU Slot 2	NimEquipmentHolder	
➥ PSU 2	NimCircuitPack	
➥ **Power Supply 2**	NemPower Supply	FIGURE 3-3 Item 16
➥ **PSU 2 Input A**	NemBinary Sensor	
➥ **PSU 2 Input B**	NemBinary Sensor	
➥ **PSU 2 Output**	NemBinary Sensor	

TABLE 3-19 Netra t 1400/1405 Containment Model *(Continued)*

Ohysical Hierarchy/Model Description	NIM/NEM Class	Refer to
➥ PSU Slot 3	NimEquipmentHolder	
➥ PSU 3	NimCircuitPack	
➥ **Power Supply 3**	NemPower Supply	FIGURE 3-3 Item 17
➥ **PSU 3 Input A**	NemBinary Sensor	
➥ **PSU 3 Input B**	NemBinary Sensor	
➥ **PSU 3 Output**	NemBinary Sensor	
➥ Motherboard	NimEquipment	FIGURE 3-2 Item 26
➥ CPU Slot 0	NimEquipmentHolder	
➥ CPU Module 0	NimCircuitPack	
➥ *<cpu0>*	NimEquipment	FIGURE 3-2 Item 18
➥ CPU Slot 1	NimEquipmentHolder	
➥ CPU Module 1	NimCircuitPack	
➥ *<cpu1>*	NimEquipment	FIGURE 3-2 Item 19
➥ CPU Slot 2	NimEquipmentHolder	
➥ CPU Module 2	NimCircuitPack	
➥ *<cpu2>*	NimEquipment	FIGURE 3-2 Item 20
➥ CPU Slot 3	NimEquipmentHolder	
➥ CPU Module 3	NimCircuitPack	
➥ *<cpu3>*	NimEquipment	FIGURE 3-2 Item 21
➥ CPU Fan Assembly	NimEquipment	
➥ **System Fan 3**	NemFan	FIGURE 3-2 Item 22
➥ **Fan 3 Tachometer**	NemNumericSensor	
➥ **System Fan 4**	NemFan	FIGURE 3-2 Item 23
➥ **Fan 4 Tachometer**	NemNumericSensor	
➥ EBus Slot	NimEquipmentHolder	
➥ Alarms Card	NimCircuitPack	FIGURE 3-2 Item 24
➥ Lights Out Management Serial Connector	NimTerminationPoint	FIGURE 3-3 Item 26
➥ Alarms Service Port Connector	NimTerminationPoint	FIGURE 3-3 Item 25
➥ **Alarm 1**	NemAlarmDevice	
➥ **Alarm 2**	NemAlarmDevice	
➥ **System (Alarm 3)**	NemAlarmDevice	
➥ Watchdog	NemWatchdog	
➥ PCI 66 Slot 1	NimEquipmentHolder	
➥ *<PCIcard1>*	NimCircuitPack	FIGURE 3-3 Item 27
➥ PCI Slot 2	NimEquipmentHolder	
➥ *<PCIcard2>*	NimCircuitPack	FIGURE 3-3 Item 28
➥ PCI Slot 3	NimEquipmentHolder	
➥ *<PCIcard3>*	NimCircuitPack	FIGURE 3-3 Item 29
➥ PCI Slot 4	NimEquipmentHolder	
➥ *<PCIcard4>*	NimCircuitPack	FIGURE 3-3 Item 30

TABLE 3-19 Netra t 1400/1405 Containment Model *(Continued)*

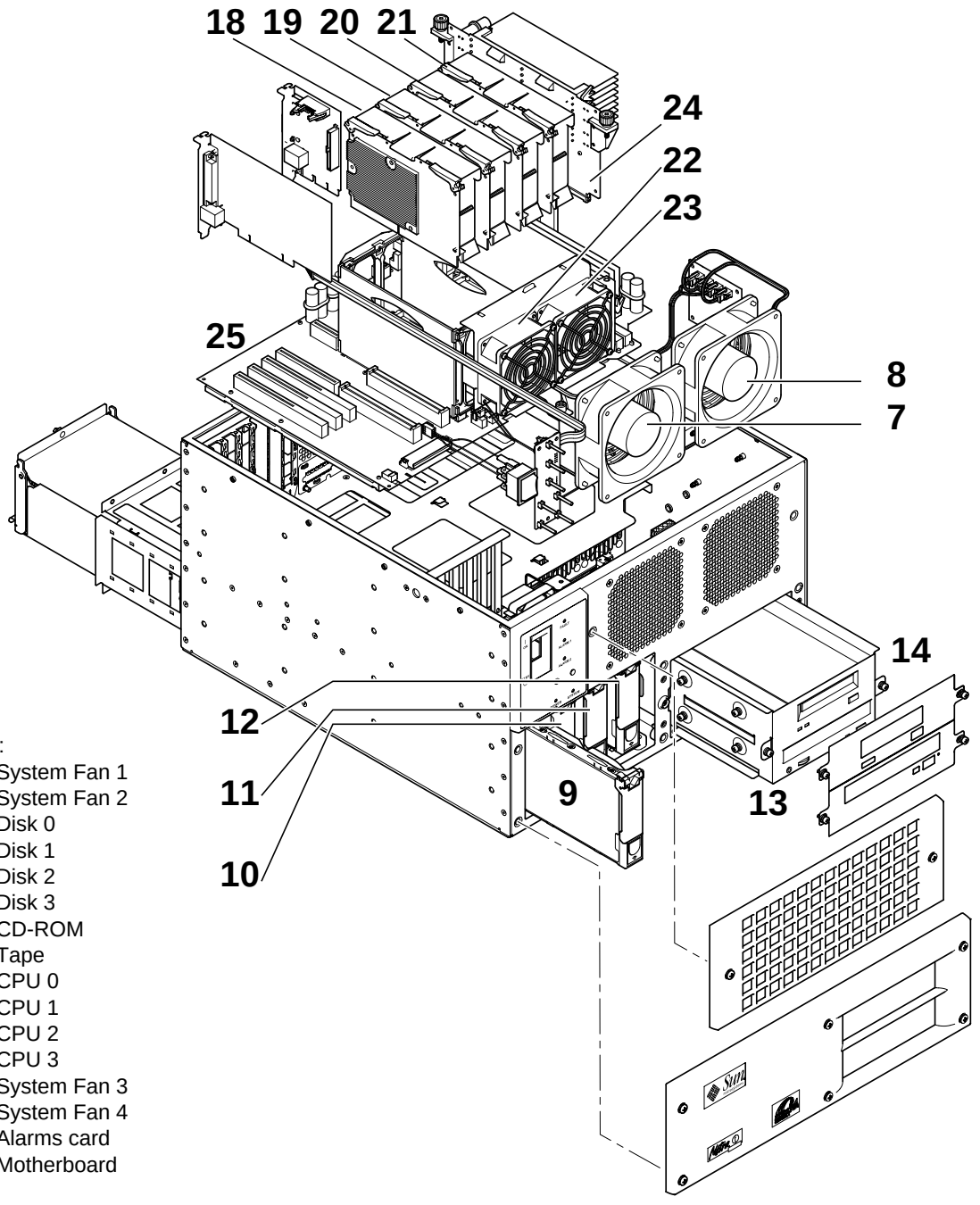
Ohysical Hierarchy/Model Description	NIM/NEM Class	Refer to
➥ TPE	NimTerminationPoint	FIGURE 3-3 Item 31
➥ Serial A	NimTerminationPoint	FIGURE 3-3 Item 32
➥ Serial B	NimTerminationPoint	FIGURE 3-3 Item 33
➥ Parallel	NimTerminationPoint	FIGURE 3-3 Item 34
➥ SCSI	NimTerminationPoint	FIGURE 3-3 Item 35



Key:

1 Fault LED
2 Alarm 1 LED
3 Alarm 2 LED
4 Supply A LED (not Netra t 1405)
5 Supply B LED (not Netra t 1405)
6 System LED

FIGURE 3-1 Netra t 1400/1405 Component Identification – Front Panel



Key:
7 System Fan 1
8 System Fan 2
9 Disk 0
10 Disk 1
11 Disk 2
12 Disk 3
13 CD-ROM
14 Tape
18 CPU 0
19 CPU 1
20 CPU 2
21 CPU 3
22 System Fan 3
23 System Fan 4
24 Alarms card
25 Motherboard

FIGURE 3-2 Netra t 1400/1405 Component Identification – Internal

Key:
15 PSU 1
16 PSU 2
17 PSU 3
25 LOM Serial Connector
26 Alarms Service Port Connector
27 PCI 1
28 PCI 2
29 PCI 3
30 PCI 4
31 TPE
32 Serial A
33 Serial B
34 Parallel
35 SCSI

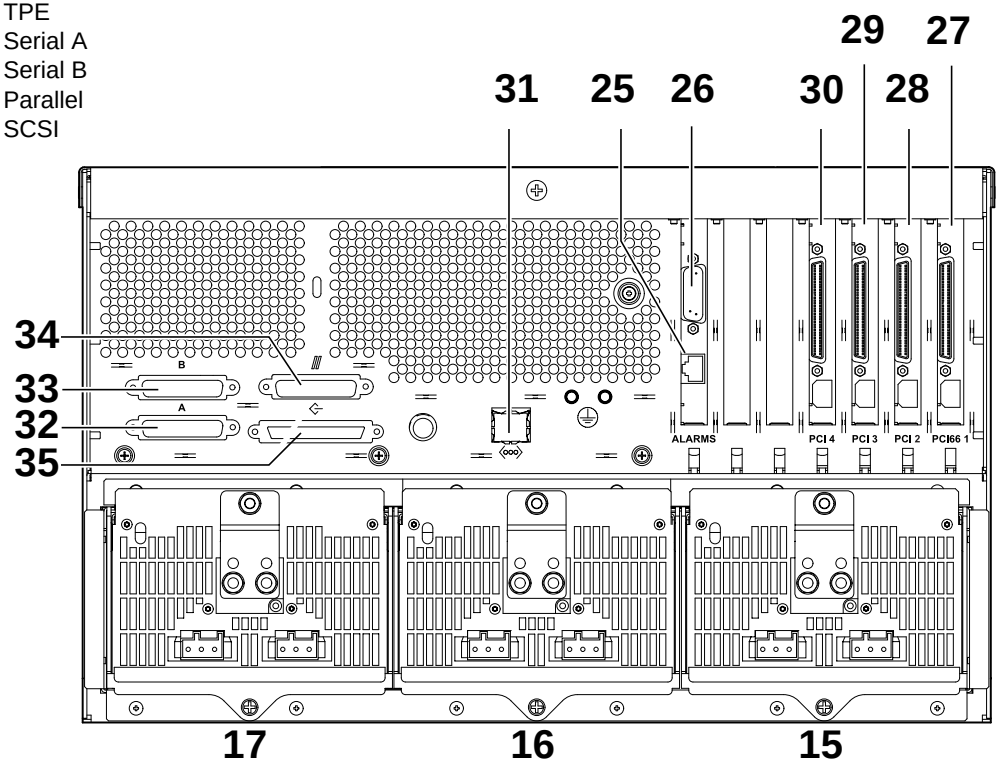


FIGURE 3-3 Netra t 1400/1405 Component Identification – Rear Panel

Index

I

L

M

N

O

P

S

T

V

W