Netra™ ft 1800 User's Guide

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 Fax 650 969-9131

Part No.: 805-4529-10
Revision A, February 1999

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

Contents

Tables

Figures

Preface

This document describes the day-to-day use of the Netra™ ft 1800 system.

All steps required to install the hardware, operating system and other software are described in the *Netra ft 1800 Installation Guide*.

Who Should Use This Guide

This guide is intended to be read by technicians and system administrators who are responsible for the regular maintenance and configuration of a Netra ft 1800 system. Installation engineers and system developers may also find information of use in this guide.

Readers should be familiar with the Solaris™ operating environment.

How This Guide Is Organized

The guide is arranged as follows:

Chapter 1, ”System Overview“, describes the system and how it works.

Chapter 2, ”System Configuration and Administration“, provides an overview of the tools and processes involved in configuring and administering the system.

Chapter 3, ”Sun StorEdge Volume Manager“, describes how the Sun™ StorEdge Volume Manager™ is used in the Netra ft 1800 environment.

Chapter 4, ”Using the Configuration Management System (CMS)“, describes the interfaces to the Configuration Management System and how to use it to manage a Netra ft 1800 system.

Chapter 5, ”Core Processing Subsystem“, provides reference information about the fault tolerant core processing subsystem.

Chapter 6, ”Power Subsystem“, provides reference information about the power supply.

Chapter 7, ”Fault Tolerant Network Subsystem“, provides reference information about the Ethernet LAN and PCI subsystems.

Chapter 8, ”Drive Subsystems“, provides reference information about the mass storage subsystems, which include disks and removable media.

Chapter 9, ”Console, Alarms and Fans“, describes the control, alarm and fans module, and the use of the consoles, alarms and environmental monitors.

Chapter 10, ”PCI Module Subsystem“ describes the system’s support for PCI carriers, which enable you to use standard PCI cards or HotPCI cards to extend the machine’s capabilities.

Chapter 11, ”Split Mode“, explains split mode and how to use it.

Chapter 12, ”Replacing Modules“, describes how to remove and replace faulty hardware.

Appendix A, ”CMS Man Pages“ contains man pages for the CMS user utilities.

Appendix B, ”Open Boot PROM Information“, describes the features and commands that are unique to the Netra ft 1800’s Open Boot PROM.

The Glossary contains terms and acronyms related to fault-tolerant computing and the Netra ft 1800 server.

Related Books

- *Netra ft 1800 Hardware Release Notes* (806-0179-10)
- *Netra ft 1800 Software Release Notes* (805-4527-10)
- *Netra ft 1800 Service Manual* (805-4528-10)
- *Netra ft 1800 CMS API Developer’s Guide* (805-5870-10)
- *Netra ft 1800 CMS Developer’s Guide* (Part No. 805-7899-10)
- *Netra ft 1800 Developer’s Guide* (805-4530-10)
- *Netra ft 1800 Hardware Reference Manual* (805-4531-10)
- *Netra ft 1800 Installation Guide* (805-4533-10)

- *Netra ft 1800 Reference Manual* (805-4532-10)
- *Netra ft 1800 Safety and Compliance Manual* (805-7019-10)
- *Sun StorEdge Volume Manager 2.5 User Guide* (Part No. 805-1603-10)
- *Sun StorEdge Volume Manager 2.5 System Administrator's Guide* (Part No. 805-1607-10).

Sun Documentation on the Web

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
```

Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

```
docfeedback@sun.com
```

Please include the part number of your document in the subject line of your email.

What Typographical Changes Mean

The following table shows the type changes and symbols used in this guide.

TABLE P-1 Typographic conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`
Open Boot Prompt	`ok>`

Symbols

The following symbols mean:

Note – A note provides information which should be considered by the reader.

Caution – Cautions accompanied by this attention icon carry information about procedures or events which if not considered may cause damage to the data or hardware of your system.

Caution – Cautions accompanied by this hazard icon carry information about procedures which must be followed to reduce the risk of electric shock and danger to personal health. Follow all instructions carefully.

CHAPTER **1**

System Overview

Netra ft 1800 is a fault tolerant computer which is designed to withstand failure in any single component.

All electrical components, including motherboards, are *hot-replaceable*. This means that they can be replaced while the system continues to run. It is also possible to configure components redundantly in software to provide fault tolerance at system and component level.

Netra ft 1800 runs the Solaris™ 2.6 operating environment or compatible versions. It is application-binary compatible with the standard range of Ultra™ Enterprise™ 450 systems from Sun.

This chapter provides an overview of the components and architecture of a Netra ft 1800 system, and of the main aspects of system configuration.

1.1 Netra ft 1800 System Components

The following components provide the functionality of a Netra ft 1800 system:

- The chassis.
- Motherboards.
- Hardware units called *modules*, which provide the hardware functionality of the system. These are also referred to within the Configuration Management System (CMS) software as *field replaceable units* (FRUs).
- System I/O connections which form part of the motherboards and modules.
- Software drivers for the modules installed in the system. The drivers are specially *hardened* to provide increased resilience.
- *Subsystems*, that is, groupings of hardware and software which provide fault tolerant functionality in specific areas, for example, networking and mass storage.

The Configuration Management System (CMS) maintains a model of the system configuration and controls the response of the system if any of its components fails. You also use the CMS to set the required fault tolerant configuration.

This section describes the overall hardware architecture of the system and provides an overview of its parts and the way it operates. See Chapter 2 “System Configuration and Administration” for an overview of how to configure the system. See Chapter 4 “Using the Configuration Management System (CMS)” for details of the CMS interface and its use.

1.1.1 Hardware Architecture

The basic configuration of the Netra ft 1800 consists of the chassis and two motherboards. This configuration can run as a live, fault tolerant system with the addition of the following modules:

- Two CPUset modules, which run the Solaris 2.6 operating environment or compatible versions
- Two disk chassis
- Two removable media modules, at least one of which contains a CD-ROM drive
- Two console, alarms and fans (CAF) modules (control panels with network connectors)
- Two power supply units (PSUs) per motherboard

The motherboard has slots for each class of module. The required minimum two modules must be inserted in slots on different motherboards.

Note – Each slot must contain an appropriate blank if it does not contain a module to maintain compliance with EMI standards and the integrity of the cooling design.

TABLE 1-1 shows the complete list of classes of module available for the Netra ft 1800:

TABLE 1-1 Module Classes

Module class	Function	Maximum number in system
CPU	Main processor	2
PSU	Power supply unit	6
DSK	Disk chassis	2
HDD	Hard disk drive	12
CAF	Console, alarms and fans	2

TABLE 1-1 Module Classes *(Continued)*

Module class	Function	Maximum number in system
PCI	PCI cards	16
RMM	Removable media module (CD-ROM drive and optional tape drive)	2
MBD	Motherboard	2

Hard disk drive (HDD) modules are inserted in the disk chassis (DSK module), not directly into the motherboard. See Section 1.1.3 “Modules” on page 1-5 for a full description of the module classes.

Network connections are provided via the CAF. PCI cards are optional, but allow for greater flexibility in configuring fault tolerant network connections.

A CPUset module is a processor and memory module that contains one to four UltraSPARC™ II processor modules and up to 4 Gbytes of memory. The CPUset module is hot replaceable, like all Netra ft 1800 modules.

An MBD (motherboard) and a CPUset module forms the basis of a *side* of the system. The sides can run separately in *split* mode, or in *lockstep* to provide *fault tolerant* redundancy. The two sides are linked by a bridge that allows the processors to use devices on both motherboards. A fault tolerant system can be regarded as not having sides: the relationship between the MBD and CPUset module becomes relevant only when the system is not in lockstep.

Each motherboard has up to three power supply units (PSUs), each of which has dual –48V nominal inputs (range is –40V to –75V). Two inputs per motherboard are required. The PSUs on each motherboard can be configured in 2+1 redundancy. Failure of a single input will not affect the system tolerance to module failure. Each PSU belongs to one motherboard only, and to one side of the system in split mode.

The system administrator can configure the system in split mode via a console which can be connected to either side in split mode.

The system operator can monitor the length of time that a module has been installed and other information related to its life-expectancy. In general, however, the Netra ft 1800 is designed to deal with system events itself and to notify the system operator, who can then take action without interrupting the service.

The system provides alarm notification of failures. See Chapter 9 “Console, Alarms and Fans”.

Full details of the Netra ft 1800 hardware are given in the *Netra ft 1800 Hardware Reference Manual*.

FIGURE 1-1 shows the front view of a Netra ft 1800 system with all the possible modules in place. The power inputs are at the rear of the system.

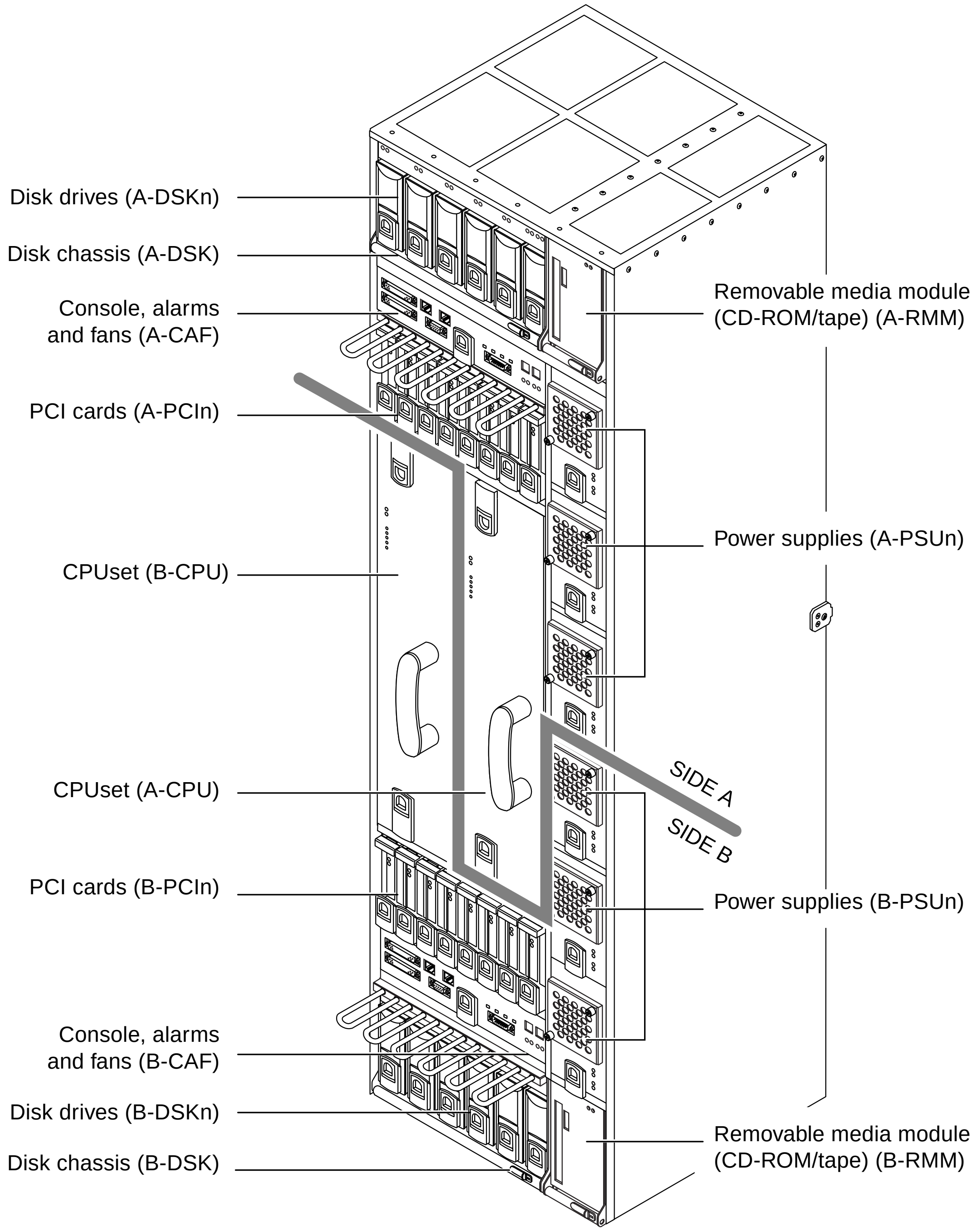


FIGURE 1-1 Netra ft 1800 System

1.1.2 Chassis and Motherboards

The chassis of the Netra ft 1800 has two motherboards attached. The motherboards are hot replaceable in fault tolerant mode, but must otherwise both be present while the system operates. One motherboard is identified as A-MBD and the other as B-MBD.

The motherboards have connectors for all the other modules in the system. There are slots in the chassis for each type of module. Each slot belongs to the motherboard to which it has a physical connection. It has a label which indicates its *location*.

The location of each module consists of the following:

- A or B, corresponding to the motherboard in which the module is inserted
- The class of module (except for HDD modules)
- The number of its slot (numbered from 0).

For example, the first PSU slot on motherboard A has the location A-PSU0.

Hard disk drive (HDD) modules are not inserted directly into the motherboard, but into a disk chassis. The location of an HDD module is the number of its slot within the chassis, for example, A-DSK0 for the first slot in the disk chassis on A-MBD.

1.1.3 Modules

In a Netra ft 1800 system, a *module* provides specific hardware functionality. One class of module provides each type of functionality. For example, processing is provided by CPUset modules and power supply by PSU (power supply unit) modules. Modules of the same class are externally identical and fit into a slot of a unique type on the chassis, with a connection to the motherboard. Each module and each slot is labeled with a mnemonic for its class. Modules fit only in the correct position in slots of the correct class.

See TABLE 1-1 for a list of the module classes available.

Physical disk storage is provided by HDD modules. An HDD module is housed in the disk chassis (DSK). HDD modules reside in locations A-DSK0 to A-DSK5 and B-DSK0 to B-DSK5.

Each module, except for an HDD module, has an EEPROM which contains the information required for the system to validate the module’s class, its peer and container compatibility, history, and other module-specific information. Information about an HDD module is stored in the DSK module’s EEPROM.

To be available to the system, modules must be physically inserted in the correct slots, identified to the CMS (*configured*) and *enabled* by the CMS. When a module is enabled, its drivers are online. See Section 4.3 “Simple Configuration Changes” on page 4-6 for information on how to enable a module.

Each module, except the motherboards and disk chassis, has a unique identifier within the CMS. This consists of its class followed by a space and a sequence number, starting from 0. For example, a system can have a maximum of six PSU modules, so a PSU module has an identifier in the range PSU 0 through PSU 5.

Note – The CMS identifier is completely separate from the location of the slot in which the module is inserted, and remains the same whatever the module’s current location. CMS identifiers contain a space; locations do not.

The motherboards have identifiers A-MBD and B-MBD. The disk chassis have the identifiers A-DSK and B-DSK.

Environmental monitors monitor the ambient temperature of all modules. The CPUset, CAF, PSU and RMM modules have their own fans, whose speed is also monitored. The environmental monitors send a message to the system when the speed or temperature is outside the acceptable range for the module.

Each module has a *Power* LED which shows when it is injected and powered on, and a *Fault* LED which indicates that it has failed. When a module fails, the system responds according to its current configuration, for example, by transferring the services that run on the failed module to another module. You can then replace the failed module, or correct the problem that caused it to fail and restart it.

All modules are *hot replaceable*. This means that you can replace them simply and safely while the system is running. See Chapter 12 “Replacing Modules” for details.

1.1.4 System I/O

System I/O expansion is via four hotPCI buses, two on each motherboard. Up to four PCI modules can be plugged into each hotPCI bus. A PCI module can be a native hotPCI bus design or can contain a standard format PCI card. This allows the system to take advantage of the functions available on off-the-shelf PCI cards.

For standard PCI cards, I/O cables plug on to the standard connectors on the PCI ISA panels, which are visible at the front of the I/O module. Console, modem, Remote Control Processor (RCP), alarm and Ethernet connections are made at the front of the machine via the CAF module.

There are locations for Fast / Wide SCSI disks in two six-slot SCSI disk chassis with individual drives. One of the built-in SCSI controllers on each motherboard connects to the disk chassis on that motherboard. Each motherboard also provides two connections for 10BaseT or 100BaseT Ethernet in the CAF module. The disk and Ethernet connections do not use the hotPCI slots.

1.1.5 Devices

Each module has an associated software device driver. You do not normally need to interact directly with devices to configure or control modules. The CMS starts and stops the devices when it enables and disables modules, and monitors their state.

The device drivers for the modules are specially *hardened* to isolate them and the operating environment from external causes of failure.

1.1.6 Subsystems

A *subsystem* is an object defined in the CMS that configures fault tolerant functionality for the system. A subsystem defines the behavior of a set of modules and their drivers.

Each subsystem provides one or more fault tolerant services. For example, a fault tolerant network subsystem provides the Ethernet interface for a fault tolerant system.

Subsystems are configured from modules that are already *enabled*. When a module in a subsystem fails, the subsystem configuration includes a policy for handling the failure.

Most subsystems must also be enabled using the CMS.

The processor subsystem is based on a CPUset module and provides processing and central control. For full details, see Chapter 5 “Core Processing Subsystem”.

The Ethernet subsystem provides fault tolerant network connections. It is configured from an appropriate combination of PCI controllers in PCI modules and motherboard Ethernet controllers (accessible via the CAF). For full details, see Chapter 7 “Fault Tolerant Network Subsystem”.

The disk subsystem provides internal mass storage. It consists of a disk chassis which can contain up to six HDD modules. Once the chassis and HDD modules are online, they can be configured as virtual volumes using Sun StorEdge Volume Manager. For information about virtual volume configuration see Chapter 3 “Sun StorEdge Volume Manager”. For information about the disk storage modules, see Chapter 8 “Drive Subsystems”.

The console subsystem provides fault tolerant serial connections, including a fault tolerant console connection. See “Console Subsystem” on page 9-5 for a list of the components of this subsystem.

1.2 How a Netra ft 1800 Operates

One or two MBD modules and one or two CPUset modules provide the central processing for a Netra ft 1800 system. An MBD module and a CPUset module form the basis of a side of the system, which can handle all the processing for the system. A side consists of an MBD and an CPUset module, and some or all of the following features associated with it:

- Console, alarms and fans (CAF module)
- Network connections (Ethernet via the CAF modules and appropriate PCI modules)
- Disk chassis (DSK) with up to six HDD modules inserted
- Removable media module (RMM module)
- Power supply (up to three PSU modules)

The Configuration Management System (CMS) software tracks and configures these modules and manages the system’s response to their failures. Individual subsystems or sets of modules can be configured redundantly to provide fault tolerance.

The Netra ft 1800 system can be configured in the following modes:

- Fault tolerant

  All hardware modules are replicated with the possible exception of the removable media module (RMM) and PCI modules. The CPUset modules run in lock-step. If one CPUset module fails, the other continues processing.

- Split mode

  A fully redundant system is brought temporarily out of synchronization and runs as two separate systems in order to upgrade and test software. One side continues processing while the other is upgraded.

Application software is binary-compatible between Netra ft 1800 systems with different levels of hardware redundancy.

1.2.1 Initial Startup

The initial configuration of the Netra ft 1800 system is determined by the hardware that is present when the system is first booted after installation. The CMS identifies the modules in the slots on the motherboard from the information in their EEPROMS. It creates objects for all the modules that are present when the system starts to boot and attempts to enable the corresponding modules.

The objects created on initial startup can be modified and further objects can be added, using the `cmsconfig` utility.

After installation, the Netra ft 1800 comes up by default in fault tolerant mode. This is achieved as follows:

1. The CPUset module on the first motherboard to be powered on boots in a way similar to other systems. It starts the modules that it is configured to use and begins to operate normally.

   Each CPUset module can be configured to use any of the other modules in the system (except the motherboard). Modules on both sides are usable as soon as the first CPUset module boots.

2. The second CPUset module then comes into lock-step with the first side to provide fault-tolerant operation.

The process by which a CPUset module comes into lock-step is called *processor re-integration* (PRI).

1.2.2 Normal Fault Tolerant Operation

When the system operates in fault tolerant mode, both CPUset modules receive exactly the same inputs and perform the same processing. Their inputs and outputs are checked to ensure that they are in lock-step. If they are found not to be in lock-step, an *out-of-sync* event is said to occur. If one CPUset module fails, the other continues processing.

The system does not need to be restarted completely under normal circumstances. Software can be upgraded by splitting the system if necessary. All modules including motherboards can be replaced without shutting the system down.

1.2.3 Split Systems

When a system is *split*, its sides (each motherboard and the CPUset on it) operate as separate computers. One side continues to operate with the same identity as it had before; the other side reboots with a new identity. The system is no longer fault tolerant at the processor level, and fault tolerant subsystems cannot be configured across the sides.

One side of a split system can be brought down without affecting the operation of the other side. This makes it possible to test and upgrade the operating environment without affecting the availability of the service.

When one side of the system is rebooted, it reads the information about the modules in the configuration from the CMS, verifies from the EEPROMs in the modules that they have not changed, and attempts to enable them.

See Chapter 11 “Split Mode” for details of how to split a system.

1.2.4 Module Failure and Replacement

Each module has environmental monitors which detect the conditions likely to lead to failure, namely fan malfunction and overheating. When these conditions are detected, the monitors send a message to the CMS, which takes appropriate action. The CMS normally notifies the system operator that the module has failed. The module’s *Fault* LED is lit.

The CMS also notifies the system operator and switches the module’s *Fault* LED on if the hardened device driver software detects a hardware failure.

The system will continue to operate, using alternative resources if configured to do so, until the module is replaced.

The system operator must replace the module as described in Chapter 12 “Replacing Modules”. This involves the following steps:

1. Disabling the failed module.
2. Replacing the failed module physically with a correct replacement.
3. Enabling the replacement module.

When the replacement module is enabled, it resumes the processing performed by the failed module.

CHAPTER **2**

System Configuration and Administration

This chapter provides an overview of the main tasks involved in configuring and administering a Netra ft 1800 system.

2.1 System Configuration

The Netra ft 1800 has a software Configuration Management System (CMS) which controls and monitors the hardware in the system. You use the CMS to perform the following tasks:

- *Configure* a module, that is, make the CMS aware that a module is present in a specified location
- Where appropriate, *Enable* a module, that is, power on a configured module, start its devices and link it into a subsystem or service
- *Disable* a module, that is, unlink a module from a subsystem or service, stop its devices and power it off
- Define a *subsystem* to provide fault tolerant functionality

A module or subsystem is available to the system only once it has been enabled by the CMS.

The CMS configures a specific module in a specified slot. See Section 1.1.2 “Chassis and Motherboards” on page 1-5 for information about slots and Section 1.1.3 “Modules” on page 1-5 for information about modules.

You use the main CMS command, `cmsconfig`, to perform the configuration tasks related to fault tolerance. A set of associated utilities is available to perform supporting tasks. See Chapter 4 “Using the Configuration Management System (CMS)” for details of these utilities.

The main applications other than the CMS that you use to configure parts of a Netra ft 1800 system are:

- Sun StorEdge Volume Manager, used to configure mass storage volumes for the system; see Chapter 3 “Sun StorEdge Volume Manager” for details
- The split mode utilities, used to move the system to and from split mode and manage ownership of the modules between the sides while in split mode; see Chapter 11 “Split Mode” for details

2.1.1 CMS Objects

The CMS works with software objects that represent the modules in specific locations, or combinations of modules configured as subsystems.

FIGURE 2-1 illustrates the relationship between the physical system and the objects in the CMS.

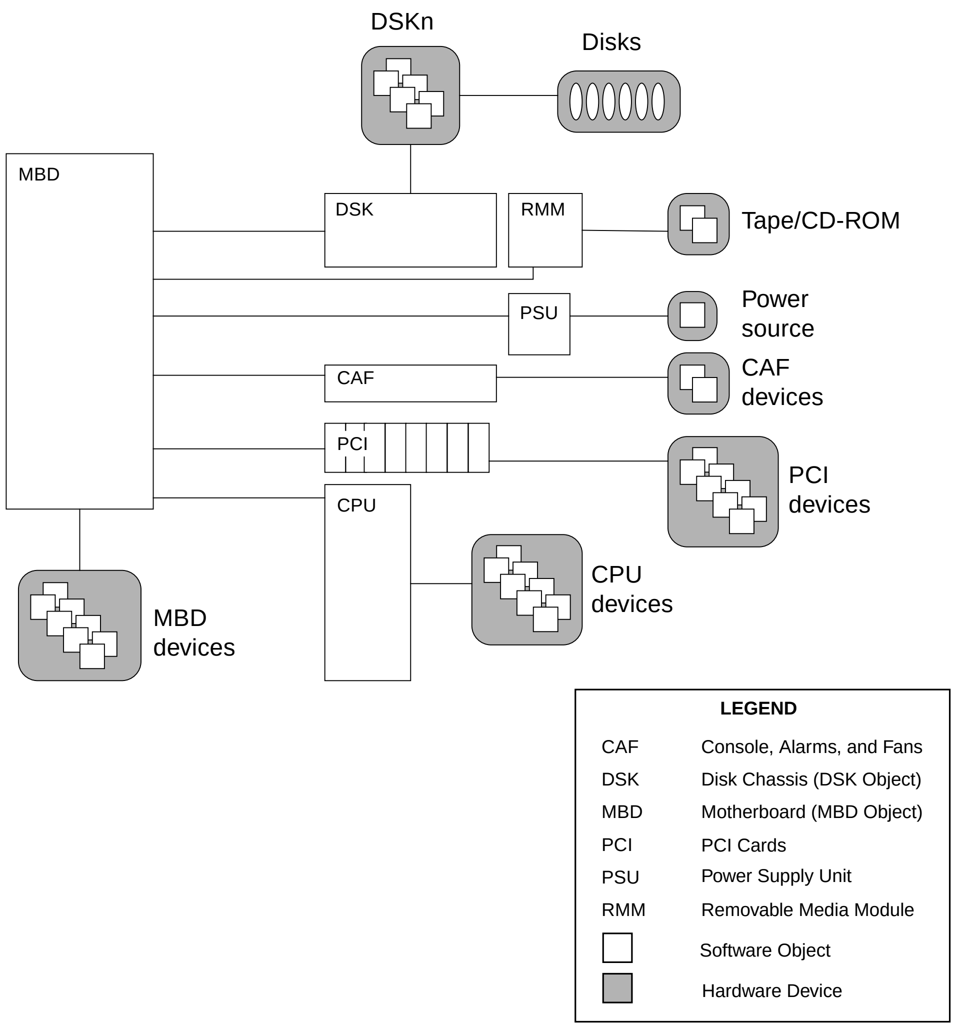


FIGURE 2-1 How the CMS Models the Physical System

CMS objects are represented in FIGURE 2-1 as white boxes. Devices are represented as smaller white boxes within the shaded boxes, which represent hardware modules. There is a one-to-one relationship between modules and CMS objects. The CMS objects control the devices within the corresponding module.

FIGURE 2-2 illustrates the relationship between the physical system and the subsystem objects in the CMS.

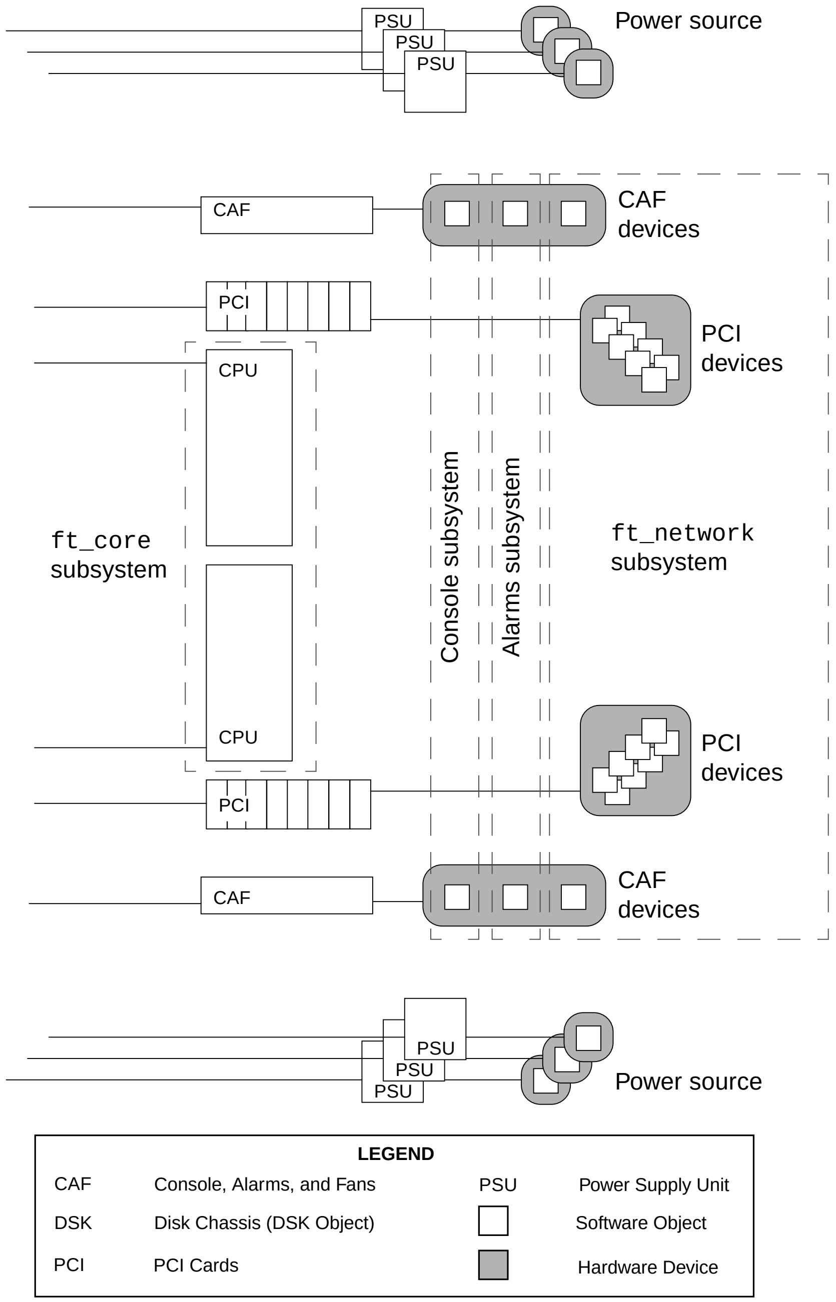


FIGURE 2-2 How the CMS Models Subsystems

The gray dashed lines show the subsystems. Power subsystems consist of PSUs on the same side configured redundantly. The `ft_core` subsystem consists of the two CPUset modules configured to respond to an out-of-sync event and to re-integrate in a specified manner. The `ft_network` subsystem consists of network connections on either side configured redundantly.

The objects that can be configured on a Netra ft 1800 system are shown in TABLE 2-1.

TABLE 2-1 CMS Objects

Object	Represents	Type
`MBD`	Motherboard	module
`CPU`	CPUset	module
`CAF`	Console, alarms and fans	module
`PSU`	Power supply unit	module
`PCI`	PCI card	module
`DSK`	Disk chassis	module
`HDD`	Disk drive	module
`RMM`	Removable media module	module
`ft_network`	Fault tolerant network subsystem	subsystem
`ft_core`	Fault tolerant core processing subsystem	subsystem
`sm`	Fault tolerant console subsystem	subsystem

The objects have common attributes and attributes specific to the type of module or subsystem which they represent, as described in the chapters in this manual that cover the subsystems. The common attributes that can be set by the user via the CMS are shown in TABLE 2-2.

TABLE 2-2 Object Attributes

Field	Values
`action`	`disable|enable`
`location`	Slot in which module is inserted
`fault_acknowledged`	`no|yes`
`user_label`	User settable label for module instance
`part_number`	The required part number of the module

The `action` attribute represents the action on the module that the user has last requested via the `cmsconfig` utility. To enable a module, you set the `action` property to `enable`; to disable a module, you set the `action` property to `disable`. This property remains set across reboots.

You set the `location` attribute to notify the system that a module is inserted in the specified slot.

The `fault_acknowledged` attribute reflects whether you have acknowledged a faulty module using the `cmsfix` utility.

The `user_label` attribute is displayed alongside the object's class name in the `cmsconfig` and `cmsfix` utilities. You can use this attribute to identify individual modules, for example, specific types of cards in PCI modules. The object cannot be accessed using the `user_label` attribute.

The `part_number` attribute stores the part number that was present in the EEPROM the first time the module was enabled. The CMS uses this number to check that the same version of the module is present when it attempts to enable it. You can delete the value of this attribute if you do not require this check before the module is enabled, for example, if you replace the module with an upgraded version.

TABLE 2-3 shows the *system attributes* of CMS objects:

TABLE 2-3 Generic System Attributes of CMS Objects

Field	Values
`description`	The class of module
`serial_number`	The serial number of the module
`faulty`	`no|yes`
`software_fault`	`no|yes`
`present`	`no|yes`
`info`	Information about the status of the module
`busylock`	`no|yes`

System attributes are written automatically by the CMS. You cannot alter them. They contain either fixed information (for example, the class of module) or variable information (for example, the module's current status).

The `description` attribute contains a permanent description of the module.

The `serial_number` attribute stores the serial number that was present in the EEPROM when the module was first enabled. This number identifies the individual module uniquely.

The `faulty` attribute specifies whether a hardware fault has occurred in the module.

The `software_fault` attribute specifies whether a software fault has occurred in the module.

The `present` attribute specifies whether the system can access the physical module. This attribute can be reset by the environmental monitor.

The `info` attribute contains messages about the module from the system.

Objects can have other system attributes that reflect the status of the module or subsystem in the system. These are described in the chapters on the individual subsystems. System attributes are marked with an asterisk (*) when they appear in tables of attributes in these chapters.

The *constituents* of objects are displayed by the CMS. Constituents are other objects (modules, interfaces or drivers) which are referenced by the object to provide part of its functionality. For example, the disk chassis has disk (HDD) modules as its constituents. Some constituents are configured automatically by the system when the constituent module is inserted and configured. Others must be assigned by setting an attribute of the object. See the chapters on the individual subsystems for details.

2.1.2 CMS Object States

The `state` field of a CMS object reflects the state within the system of the corresponding module or subsystem. It is displayed in the `cmsfix` and `cmsconfig` utilities. The value in the `state` field is changed as a result of changes to the module, which may be a result of either user action or something that occurs in the module.

The standard states for modules are shown in TABLE 2-4.

TABLE 2-4 CMS Object States

State	Means	Changes to
`initial`	The location is not initialized.	`not_present`, `disabled`
`not_present`	The location is not assigned to any module.	`disabled`
`disabled`	A module is configured in the location, but is not enabled.	`busy`, then `enabled` or `enable_failed`
`enabled`	The module is available to the system.	`busy`, then `disabled` or `disable_failed`

TABLE 2-4 CMS Object States *(Continued)*

State	Means	Changes to
`disable_failed`	The system has attempted to disable the module but failed.	`busy`, then `disabled`
`enable_failed`	The system has attempted to enable the module but failed.	`busy`, then `enabled`
`busy`	The module is in transition.	`enabled`, `enable_failed`, `disabled` or `disable_failed`

When an object changes its state, it changes from its first state to `busy`, then from `busy` to its new state.

If the system cannot enable a module, this may be due to a hardware problem in the module, in which case the `faulty` attribute has the value `yes` and the *Fault* LED is lit. Alternatively, the module is not powered on, or the driver for the device is not correctly installed. If the problem is in the driver software rather than hardware, the *Fault* LED is not lit, although the `faulty` attribute still has the value `yes` .

If the system cannot disable a module, this is generally because it has been unable to take some or all of the module's devices offline, for example, because an application or service is still using them.

2.2 System Administration

Each side of a Netra ft 1800 is a standard Solaris 2.6 system, which provides services in the usual way. The main system administration tasks, such as software administration and system upgrades, are performed as described in the Solaris documentation. The CMS does not deal with services, although it can be used indirectly to manage the devices which services use. The system can run in either fault tolerant mode, when both sides operate in lockstep to provide a single fault tolerant system, or in split mode, when the sides operate as two separate systems.

See the *Solaris 2.6 System Administrator's Guide* for information about Solaris system administration tasks. See Chapter 11 "Split Mode" for information about splitting and merging a fault tolerant system.

While the Netra ft 1800 system is running, you can perform the following tasks to administer the fault tolerant hardware:

- Monitoring the system using the `cmsfix` utility and via the system logs. See Section 4.1 "The cmsfix Utility" on page 4-2.
- Dealing with module failure and replacing modules during routine maintenance. See Chapter 12 "Replacing Modules" for details.

- Reconfiguring the system's fault tolerance.
  See Chapter 5 "Core Processing Subsystem" and Chapter 7 "Fault Tolerant Network Subsystem" for details.
- Upgrading software.
  See Chapter 11 "Split Mode" for details.

2.2.1 Dealing With Module Failure

When a module fails, the following typically occurs:

1. The module fails physically.
2. The CMS becomes aware that the module is no longer functioning correctly. Its status is marked as `faulty`. Its *Fault* LED is lit.
3. The CMS takes actions defined to power down the module. For example, it can transfer a service to an alternative module in the fault tolerant subsystem.
4. The system operator ensures that nothing is using the devices supported by the module.
5. The system operator disables the module.

Note – Before it is possible to disable the module, any services that use it must be closed down.

---

6. The system operator replaces the module physically with the correct replacement, or fixes any other hardware problem.
7. The system operator enables the module.

See Chapter 12 "Replacing Modules" for full details of this process.

2.2.2 Replacing Modules During Routine Maintenance

A module can be replaced even if it has not failed, for example, to use a different sort of PCI card in the system, or as a precautionary measure when the end of the module's expected life is approaching. The steps involved are typically the following:

1. The system operator ensures that nothing is using the devices supported by the module.
2. The system operator uses the CMS to disable the module.

3. The system operator replaces the module physically with the correct replacement and powers it on.

4. The system operator enables the module.

2.2.3 Changing the System's Fault Tolerance

In some circumstances, the requirements for fault tolerance, or the specific modules available for use in the system, can change. The fault tolerant subsystems can be reconfigured while the system is running to specify different actions when a failure occurs. See the chapters on the individual subsystems for details.

2.2.4 Upgrading the System Software

You can upgrade the software (operating environment or applications) to meet changing requirements. Depending on the software to be upgraded, the split mode feature can be useful. The system becomes temporarily non fault tolerant when the system is split, but the loss of service usually associated with software upgrades is minimal.

In general terms, software upgrades involve:

1. Splitting the system, leaving one side running the existing services. See Section 11.1 "Splitting a System" on page 11-2.

2. Performing the required upgrade and testing on the new side.

3. Gradually migrating services from one side to the other. This normally involves transferring ownership of the modules on which the services are installed. In general, you install new or upgraded applications on disks on the *losing* side, then merge the system and re-create the disk mirror to include the upgraded disks. You install an upgraded operating environment on disks on the losing side, then merge the system to use the configuration of the old losing side. See Chapter 11 "Split Mode" for details.

4. Re-establishing the disk mirror and the other fault tolerant subsystems.

5. Merging the system. See Section 11.2 "Merging a Split System" on page 11-12.

You should consider the fallback options when planning an upgrade. In particular, you should decide, if the software upgrade fails on one side, whether it is better to have the upgraded system running non-redundantly, or the non-upgraded system running redundantly with a recovered system.

2.3 Environmental Monitoring

The Netra ft 1800 includes a facility that monitors the environmental state of the modules and reports changes from the acceptable state to a log or to the screen.

The environmental monitor reports whenever a module undergoes a change of condition, from a normal state to an unacceptable state or the reverse. It monitors the following states:

- The ambient temperature of the module
- The speed of the module's fan

In addition, there are sensors on each of the CPUs inside a CPUset module. These monitor the temperature within the CPU.

Messages from the environmental monitors are written to the status log, the module's EEPROM and the module's `info` attribute. When an unacceptable state occurs, the CMS takes appropriate action, typically switching on the *Fault* light on the module. See "Environmental Failures" on page 2-13 for details of environmental error messages.

Note – The environmental monitor does not shut down modules automatically. The behavior of modules when the environmental monitor reports an alarm is determined by the CMS configuration.

2.4 Status Log

The status log for the Netra ft 1800 is in the file `/var/opt/SUNWlogo/u4ftlog.status`.

The file contains messages that identify module failures in a form similar to the following:

```
class number in location not found
class number in location not correct FRU class
class number in location has previous fault
class number in location failed peer compatibility check
class number in location failed Motherboard compatibility check
event_date: [high|low] temperature
event_date: [high|low] fan speed
```

The placeholder *class* contains the class of a module, the placeholder *number* contains the sequence number of the module, and the placeholder *location* contains the location of the module.

2.5 Failures and Error Messages

The following sections lists possible failures and their associated messages. Messages are displayed on either or both of the following:

- The status log
- The `info` field of the module.

Some messages are displayed on the console. These messages are also stored in the status log or in a module's `info` field.

Note – When a module has detected a fault, the red *Fault* LED is lit.

2.5.1 EEPROM Validation Failures

If the CMS validation on the EEPROM fails, it writes one of the following messages to the CAF or PCI module `info` field:

```
FRU not found in location
FRU in location is not of class class
FRU in location has a previous fault
FRU in location has failed peer compatibility checks
```

`FRU in` *location* `has failed MBD compatibility checks`

`FRU in` *location* `conflicts with host identity`

The corresponding messages in the status log are as follows:

class number `in` *location* `not found`

class number `in` *location* `not correct FRU class`

class number `in` *location* `has previous fault`

class number `in` *location* `failed peer compatibility check`

class number `in` *location* `failed Motherboard compatibility check`

Indicates that the host identity stored in one MBD module is different from the identity stored in the other MBD module.

The CMS will not continue to configure a module which it cannot find, or which is of the wrong class or incompatible with the MBD. The CMS will only configure a module that fails compatibility checks if it is the one working module, or if forced by the user. The CMS will continue configuring a module that reports a previous unfixed fault.

2.5.2 Environmental Failures

If the environmental characteristics of a module deviate from normal, the CMS will cause the module to be faulty. It writes one of the following messages to the module EEPROM:

class number `in location` *location* `has a [high|low] temperature`

class number `in location` *location* `has a [high|low] fan speed`

The corresponding messages in the status log are:

event_date`: [high|low] temperature`

event_date`: [high|low] fan speed`

It writes one of the following messages as appropriate into the `info` attribute of the module object:

```
cpu X temperature is too [high|low]
ambient temperature is too [high|low]
fan X has a [high|low] speed
```

In the last message, X represents the identifier of one of the CPUs in the CPUset module, in the range 0-4.

CHAPTER 3

Sun StorEdge Volume Manager

Sun StorEdge Volume Manager is a suite of software designed to handle online storage management, specifically the configuration and administration of disk-based storage without interruption of availability. The Netra ft 1800 uses the disk mirroring capabilities of Sun StorEdge Volume Manager to provide disk fault tolerance.

Sun StorEdge Volume Manager uses a redundant array of inexpensive disks (RAID), where duplicate copies of the data, or information about the data stored on the system, are held. This duplication makes it possible to regenerate data in the event of a disk failure.

Sun StorEdge Volume Manager provides the following benefits:

- Maximized system availability
  Sun StorEdge Volume Manager provides online administration including disk replacements, load balancing, plus extensions of file systems and databases.
- Improved performance
  - Disk striping and RAID features increase I/O throughput.
  - Performance monitoring aids load balancing.
- Simplified administration
  - Easy-to-use facilities such as the online backup facilities.
  - Represents storage as a pool of disk space without physical limitations.

Sun StorEdge Volume Manager is a complex product that is completely separate from the Netra ft 1800 software. You must have a good understanding of Sun StorEdge Volume Manager to use it on the Netra ft 1800.

The Sun StorEdge Volume Manager documentation describes the normal use of the Volume Manager. This chapter describes the special considerations involved when using Sun StorEdge Volume Manager to configure and manage disk storage for Netra ft 1800. Please see the *Netra ft 1800 Installation Guide* and the Sun StorEdge Volume Manager documentation for information about installing and configuring Sun StorEdge Volume Manager on the Netra ft 1800.

This section should be read in conjunction with the Sun StorEdge Volume Manager documentation to gain an understanding of the detailed requirements and constraints of using Sun StorEdge Volume Manager on the Netra ft 1800. It is not a user's guide for the Sun StorEdge Volume Manager product.

3.1 Using Sun StorEdge Volume Manager

3.1.1 Installing, Upgrading and Configuring Sun StorEdge Volume Manager

You install and upgrade Sun StorEdge Volume Manager on the boot disk of the Netra ft 1800 as described in the *Sun StorEdge Volume Manager 2.5 Installation Guide* and configure it as described in the *Sun StorEdge Volume Manager 2.5 System Administrator's Guide*. See the *Netra ft 1800 Installation Guide* for full details of special considerations for installing Sun StorEdge Volume Manager on the Netra ft 1800.

3.1.2 Replacing an HDD Module Used by Sun StorEdge Volume Manager

Sun StorEdge Volume Manager and the CMS work independently. Sun StorEdge Volume Manager can manage only disks that are enabled by the CMS. The CMS can disable only disks that the Volume Manager does not hold open. This means that in order to replace an HDD module (a hard disk), you must remove any storage allocated on it.

The following procedures use the Volume Manager menu interface wherever possible. See the *Sun StorEdge Volume Manager 2.5 System Administrator's Guide* for full details of the Volume Manager interfaces.

Note – You must replace a boot disk with a disk of similar geometry. To replace a boot disk with a disk of a different geometry, move all volumes from the boot disk to an alternative disk, then remove the old boot disk.

▼ Adding a Disk

1. **Insert the HDD module (disk) in the disk chassis.**

   See "To Replace a Module" on page 12-10.

2. **Configure and enable the HDD module.**

   See "Adding a Module to the CMS" on page 4-7.

3. **Identify the device in the disk**

   Use `cmsconfig` to view the `Disk` attribute of the DSK module that contains the disk device.

4. **Check that Volume Manager is aware of the disk:**

   Volume Manager should become aware of the disk automatically. In the rare event that this does not happen, use the following command:

```
# vxdctl enable
```

5. **Bring the disk under Volume Manager control:**

   Enter the `vxdiskadm` utility, select item 1 and follow the instructions.

6. **Allocate storage on the disk as required using your preferred method.**

▼ Removing a Disk

1. **Identify any storage allocated on the disk**
2. **Remove or relocate the storage.**
3. **Remove the disk from Volume Manager control:**

   Enter the `vxdiskadm` command, select item 3 or 4 and follow the instructions.

4. **Disable the HDD module.**

   See "Removing a Module" on page 4-9.

5. **Remove the HDD module from the disk chassis.**

   See "To Remove a Faulty Module" on page 12-4.

3.1.3 Error Recovery

3.1.3.1 Failures During the Boot Process

The Netra ft 1800 boots from boot disks encapsulated in Sun StorEdge Volume Manager without the need for operator intervention, in all but the most exceptional circumstances. Use of the `vxaltstale` script will allow the system to boot automatically from alternate boot disks when the primary or specified boot disk has been detected as stale.

There are rare circumstances such as system file corruption that can cause the system to fail to boot. These are covered in Appendix B, “Recovery”, in the *Sun StorEdge Volume Manager 2.5 System Administrator's Guide*.

3.1.3.2 Disk Failures During Operation

Techniques for recovering from disk failures are fully documented in the *Sun StorEdge Volume Manager 2.5 User's Guide* and *System Administrator's Guide*.

Note – Use of the Sun StorEdge Volume Manager hot-relocation feature on a Netra ft 1800 that runs in fault tolerant mode is not recommended. If a disk failure occurs, the storage is automatically reallocated, not necessarily in a fault tolerant manner. When the faulty disk is replaced, the storage is not automatically re-allocated back to the original configuration. This operation would need to be performed manually, and might prove difficult to achieve.

CHAPTER **4**

Using the Configuration Management System (CMS)

This chapter introduces the main aspects of the CMS for users who need to configure the fault-tolerance of the Netra ft 1800 and to hot-replace faulty modules.

The Configuration Management System (CMS) provides system management for the Netra ft 1800 system. All activities take place without interrupting the operating environment. Applications can continue to run provided that you do not disable any resource they are using. Users of the applications and services on the system need not know that you are using the CMS.

The CMS provides facilities for managing the fault tolerance of the system, except for disk storage. You must use Sun StorEdge Volume Manager to configure fault tolerant disk storage. See Chapter 3 "Sun StorEdge Volume Manager" for information.

The CMS provides facilities to manage and maintain the system as follows:

TABLE 4-1 CMS Tasks

To do this...	Use this utility:
Replace a failed module	`cmsfix`
Configure system fault tolerance	`cmsconfig`
Add modules to the system	`cmsconfig`
Check a module's *Fault* LED	`cmsledctl`
View the contents of a module's EEPROM	`cmsfruinfo`

The `cmsfix` utility provides all the facilities needed to handle the replacement of a failed module. `cmsfix` provides a user interface which requires no detailed knowledge of the CMS modules. It is described in more detail on page 4-2.

The x`cmsfix` utility is an X-based version of `cmsfix`.

The `cmsconfig` utility provides all the facilities that a system user needs to change the configuration of the machine. `cmsconfig` is an interface to the CMS primarily intended for use by OEMs and VARs. It is described in more detail on page 4-4.

The other utilities are described in the man pages. Appendix A "CMS Man Pages" contains all the man pages for the CMS utilities.

The CMS utilities are located in the directory `/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin`. Add this path to the system's path list if it is not already included.

See the *Netra ft 1800 Developer's Guide* (Part No. 805-4530-10) for full information about writing device drivers. See the *Netra ft 1800 CMS API Developer's Guide* (Part No. 805-5870-10) for full information about the CMS API.

The following section describes the main functions of the CMS.

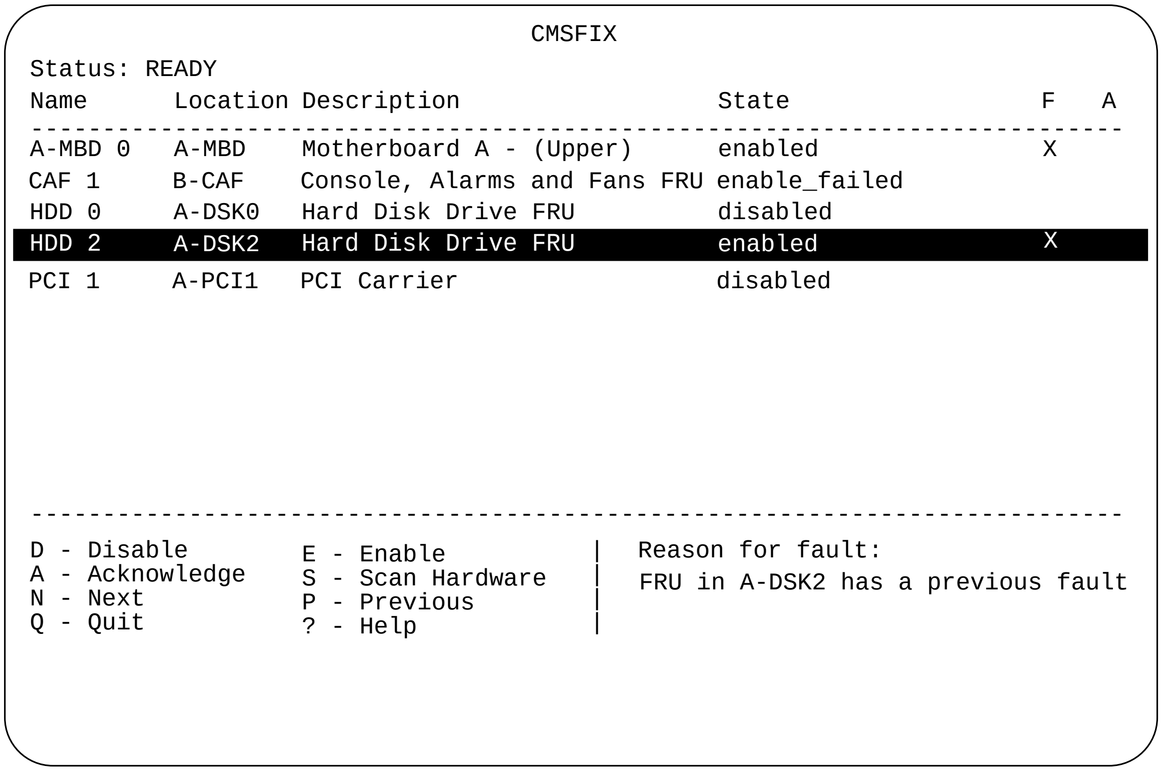
4.1 The `cmsfix` Utility

Use the `cmsfix` utility to handle the replacement or repair of a failed module. Messages that signal a failure appear in the system log. (See Section 2.4 "Status Log" on page 2-11 for details.) Chapter 12 "Replacing Modules" describes in detail how to use `cmsfix` to replace modules.

`cmsfix` displays the names and status of the modules and subsystems that are enabled and that are not functioning as required. A typical `cmsfix` display is shown in FIGURE 4-1.

You can also use cmsfix from the command line. See the the manual page `cmsfix`(1M) on page A-5 or online for full details of the command.

`xcmsfix` is an X-windows based alternative to `cmsfix`. See the manual page `xcmsfix`(1M) for full details.



```
                                    CMSFIX
Status: READY
Name      Location Description                 State                 F   A
-----------------------------------------------------------------------------
A-MBD 0   A-MBD    Motherboard A - (Upper)     enabled               X
CAF 1     B-CAF    Console, Alarms and Fans FRU enable_failed
HDD 0     A-DSK0   Hard Disk Drive FRU         disabled
HDD 2     A-DSK2   Hard Disk Drive FRU         enabled               X
PCI 1     A-PCI1   PCI Carrier                 disabled







-----------------------------------------------------------------------------
D - Disable          E - Enable          |  Reason for fault:
A - Acknowledge      S - Scan Hardware   |  FRU in A-DSK2 has a previous fault
N - Next             P - Previous        |
Q - Quit             ? - Help            |
```

FIGURE 4-1 Typical `cmsfix` Display

Each line in the main section (between the horizontal rules) contains the details of a module which has been configured in the system. The fields are shown in TABLE 4-2.

TABLE 4-2 `cmsfix` Parameters

Field	Contains
Name	The name of the module (class followed by sequence number).
Location	The side, class and position of the slot.
Description	A text description of the module.
State	The state of the module.
F	X if the module is faulty.
A	a if the module's fault is acknowledged.

To select a module, use the arrow keys to highlight it.

The key in the lower left part of the screen contains the commands for use with `cmsfix` and their abbreviations. Type the character to the left of each command to run the command on the selected module. The commands are shown in TABLE 4-3.

TABLE 4-3 `cmsfix` Commands

Key	Command	Use
D	Disable	Disable the module so that it is no longer available to the system.
E	Enable	Enable the module so that it is available to the system again.
A	Acknowledge	Acknowledge the fault in the module.
S	Scan hardware	Scan the hardware currently present in the system and update the display.
N	Next	View the next screen of the `cmsfix` display.
P	Previous	View the previous screen of the `cmsfix` display.
Q	Quit `cmsfix`	Quit `cmsfix.`
V	View	View attribute field for module. Default is `description`.
?	Help	View the `cmsfix` help.

See the `cmsfix`(1M) manual page (page A-5) for a full description of each of the commands.

4.2 The `cmsconfig` Utility

The `cmsconfig`(1M) utility allows you to view or modify the attributes and constituents of objects. The main restriction is that you may not change the value of any system attribute field. System attribute fields are marked with an asterisk (*) in tables of attributes in this guide.

You can also use `cmsconfig` as a fault-management tool to change the status of an object. For example, when you install a replacement for a failed module, you can re-integrate the replacement module into the system by setting the `action` attribute of the module to `enabled`. However, fault-management tasks can usually be more easily performed using the utility `cmsfix`.

`cmsconfig` displays the details of the modules and subsystems currently configured in the system. A typical `cmsconfig` display is shown in FIGURE 4-2.

```
Item Name          Fault Loc    State                   Page 1 of 2
-------------------------------------------------------------------------------
0    A-MBD 0             A-MBD  enabled
1    B-MBD 0           X B-MBD  enabled      FRU in B-MBD has previous fault
2    CAF 0               A-CAF  enabled
3    CAF 1               B-CAF  enabled
4    CPU 0               A-CPU  enabled
5    CPU 1               B-CPU  enabled
6    DSK 0               A-DSK  enabled
7    DSK 1               B-DSK  disabled
8    HDD 0               A-DSK0 enabled
9    HDD 1               A-DSK1 enabled
10   HDD 2               B-DSK0 initial
11   HDD 3               B-DSK1 initial
12   PCI 0               A-PCI0 enabled
13   PCI 1               A-PCI1 enabled
14   PCI 2               A-PCI2 enabled
15   PCI 6               A-PCI6 enabled
16   PCI 7               A-PCI7 enable_faile device instance failed to attach
17   PCI 8               B-PCI0 disabled
18   PCI 9               B-PCI1 disabled
19   PCI 10              B-PCI2 disabled

(H)elp, (I)nclude, (E)xclude, (S)elect, (P)age, (V)iew, (Q)uit or <Number> ?
```

FIGURE 4-2 Typical `cmsconfig` Display

Each line in the main section (between the horizontal rule and above the bottom line) contains the details of a module which has been assigned a location in the system. The fields are shown in TABLE 4-4.

TABLE 4-4 `cmsconfig` Parameters

Field	Contains
Item	The item number of the module in the current screen
Name	The name of the module (class followed by sequence number)
Fault	Contains X if the module is faulty. (The *Fault* LED on the module is lit.)
Location	The side, class and position of the slot
State	The state of the module

To see all the attributes of the object, type the number in the Item field.

The key in the bottom line of the screen contains the commands for use with `cmsconfig`. Type the character in parenthesis in each command followed by Return to run the command. The commands are shown in TABLE 4-5.

TABLE 4-5 `cmsconfig` Commands

Key	Command	Use
H	Help	See `cmsconfig` help
I	Include	Include objects in the display
E	Exclude	Exclude objects from the display
P or PN	Page	View the next page of the `cmsconfig` display
PP	Page	View the previous page of the `cmsconfig` display
S	Select	Jump to the detail of the specified object. For example, S HDD 2 displays the attributes of HDD 2.
T	Top	Jump to the top-level (initial) page.
Q	Quit	Quit `cmsconfig`
V	View	Specify the default view of the `cmsconfig` display's text field. Default is `info` field.

See the `cmsconfig`(1M) manual page (page A-2) for a full description of each of the commands, and of the arguments for the I and E commands.

4.3 Simple Configuration Changes

The examples in this section illustrate how to inform the CMS about configuration changes. These procedures apply whether the hardware changes are trivial or complex. This section does not provide hardware details for all possible configuration changes. Instead, the examples here are cases where the hardware installation work is trivial, in order to illustrate how the CMS interfaces work.

See Chapter 12 “Replacing Modules” for information about replacing modules physically.

See the chapters on the individual modules for details of module- or subsystem-specific configuration tasks for each subsystem.

4.3.1 Adding a Module to the CMS

Once the module is physically present, use `cmsconfig` to configure and enable it so that is it available to the live system.

1. **Identify the slot that contains the module.**

   A label on the chassis identifies each slot.

2. **Start `cmsconfig` if it is not already running:**

```
# /usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsconfig
```

   The display includes all the modules in the system. See Section 4.2 "The `cmsconfig` Utility" on page 4-4 for an illustration. You enter the commands in the rest of this procedure at the `cmsconfig` prompt.

3. **Select a CMS object for the new module.**

   a. **Display all the modules of the required class:**

```
i name
```

   name is the class of the module, for example, PCI. You must enter it in uppercase.

   b. **Exclude the modules that are already part of the current configuration:**

```
e all present
```

   The modules remaining are those added by the previous command. Each line in the display corresponds to a CMS object that is available for describing the new module.

   c. **Select an object from the display.**

   Enter the item number of an object (displayed in the left-hand column of the display). The object's attributes are displayed.

   The object's `location` is `NULL`.

4. **Specify the slot location for the object.**

   Enter the item number for the `location` attribute of the object. `cmsconfig` displays the possible values for the `location` attribute.

   Select the location of the module by entering the corresponding item number.

The list of the object's attributes is displayed. The object's state is now `disabled`.

5. **Enable the module.**

   Enter the item number for the `action` attribute. `cmsconfig` displays the possible values for the `action` attribute.

   Select `enable` by entering the corresponding item number.

   The list of the objects' attributes is displayed. Press `t` to return to the top level menu.

   The CMS now attempts to enable the new module. If the CMS is successful, the attribute `state` changes from `disabled` to `busy`, and then to `enabled`. Press Return to redraw the display with the most up-to-date values of the attributes.

   As part of the process of configuring the module, the system attempts to bring the devices in the module online. The module's *Power* LED is lit if it succeeds.

   If the CMS fails to enable the module, for example, because the necessary devices fail, the module's `state` attribute changes from `disabled` to `busy`, and then to `enable_failed`. Its `info` attribute contains diagnostic information. There are a number of possible reasons for this:

   - The module is faulty
   - A software problem has occurred
   - The module is incompatible with another module in the system, or with the motherboard
   - The module is not of the class specified in the CMS

   The `info` attribute of the module contains information which can point to the sources of the problem.

   To exit from `cmsconfig`, type `q` repeatedly until you see the system prompt.

4.3.2 Removing a Module

Use `cmsconfig` to remove a module from the live system by disabling it and unconfiguring it.

1. **Start `cmsconfig` if it is not already running:**

```
# /usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsconfig
```

   The display includes all the modules in the system. See Section 4.2 "The `cmsconfig` Utility" on page 4-4 for an illustration. You enter the commands in the rest of this procedure at the `cmsconfig` prompt.

2. **Identify and select the module.**

   Enter the item number of the module (displayed in the left-hand column of the display).

   The object's attributes are displayed.

3. **Disable the module.**

   Enter the item number for the `action` attribute of the object. `cmsconfig` displays the possible values for the `action` attribute.

   Select `disable` by entering the corresponding item number, and wait until the module is disabled. The `state` attribute changes from `enabled` to `busy`, and then to `disabled`. Press Return to redraw the display with the most up-to-date values of the attributes.

   When a module is disabled, its *Power* LED is off. It can be removed physically from the system if required. See "To Remove a Faulty Module" on page 12-4 for details of removing a module.

   If the module is not disabled successfully, its state attribute changes from `enabled` to `busy`, and then to `disable_failed`. Identify the cause (most likely, an application that is still using the module) and resolve the problem. The `info` attribute will contain diagnostic information.

   If a fault tolerant network subsystem is using the module, the CMS unlinks the subsystem from the module when it disables it. The CMS will relink the subsystem if the module is enabled again.

4. **Optionally, set the location of the module to `NULL`.**

   If you do not wish to replace a module which you have disabled, you can release the CMS object assigned to it.

   Enter the item number for the location attribute of the module. `cmsconfig` displays the possible values for the `location` attribute.

Select `NULL`.

Note – If you remove a module physically *before* you disable it, the CMS treats this as a failure of the module and takes appropriate action, which includes marking the module as faulty, and logging and reporting a fault.

4.3.3 Moving a Module to a New Location

If you wish to move a module to a new location, but keep the same identity for the module, follow the steps in this section.

1. **Start `cmsconfig` if it is not already running:**

```
# /usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsconfig
```

All the modules in the system are displayed.

2. **Enter the item number of the module (displayed in the left-hand column of the display).**

The module's attributes are displayed.

3. **Disable the module.**

Enter the item number for the `action` attribute of the object. `cmsconfig` displays the possible values for the `action` attribute.

Enter the item number of `disable` and wait until the module is disabled. The `state` attribute will change from `enabled` to `busy`, and then to `disabled`. To monitor this change, press Return on the `cmsconfig` display.

When a module is disabled, its *Power* LED is off. It can be removed physically from the system if required.

If the module is not disabled successfully, its state attribute changes from `enabled` to `busy`, and then to `disable_failed`. Identify the cause (which is most likely to be an application that is still using the module) and resolve it. The `info` attribute will contain diagnostic information.

4. **Move the module physically and change the CMS configuration as required.**

   a. **Remove the module and insert it in its new slot location.**

b. **Change the CMS configuration:**

   Enter the item number for the `location` attribute of the object. `cmsconfig` displays the possible values for the `location` attribute.

   Select the new location of the module by entering the corresponding number.

5. **Enable the module again.**

   Enter the item number for the `action` attribute of the object. `cmsconfig` displays the possible values for the `action` attribute.

   Select `enable` by entering the corresponding number. The module is enabled and its devices are online. The module's *Power* LED is on.

4.4 Fault-Tolerant Configuration Changes

The examples in the previous section were of simple configuration changes because they do not affect the other objects in the system. Other configuration changes may be more complex. The chapters on the specific subsystems contain information about configuration changes which affect the fault tolerance of the system. See also Section 7.4 "Configuring `ft_network`" on page 7-7.

CHAPTER 5

Core Processing Subsystem

This chapter contains software information for the core processing subsystem of the Netra ft 1800 system. The core processing subsystem consists of CPUs and attendant memory that form the “brains” of the Netra ft 1800.

5.1 Overview

The core processing subsystem consists of a fault tolerant configuration of the motherboard (MBD) and processor (CPUset) modules and their associated devices which allows hot-plug replacement of these modules. It also allows control of the way in which the system is split into two sides which run separately and the way in which it is re-integrated into a single system.

The core processing subsystem includes one or two CPUset modules. If there are two CPUset modules, they can run redundantly in fault tolerant mode. If there is only one CPUset module, which can be in either location, fault tolerant operation is not possible.

5.2 Hardware and Software Environment

A-CPU and B-CPU identify the slot locations used for CPUset modules. These modules contain all the main processing power and memory capacity of a Netra ft 1800 system.

5.2.1 Overview

When a Netra ft 1800 is powered up and configured to be in combined mode, the first CPUset module to power up becomes the working CPUset module. If there are two CPUset modules in the machine, the second CPUset module powers up, then waits for an instruction from the CMS to integrate.

When a CPUset module is hot-replaced, it is powered on automatically by the hardware, but waits for an instruction from the CMS to integrate.

The `splitadm` command is used to make the CPUset modules run as two separate systems. See Chapter 11 “Split Mode” for details.

There are software devices associated with the CPUset and MBD modules. These are brought on and off line by the CMS objects associated with the hardware. They should not be brought online and offline by any other means.

5.3 Processor CMS Objects

The core processing subsystem is represented in the CMS by the `ft_core` object, which represents the fault tolerant core of the machine. This does not correspond to a specific module. It is configured in software to control the operation of motherboards, CPUset modules and associated devices in a fault tolerant configuration. The motherboards and CPUset modules are also represented by CMS objects.

The `ft_core` object is aware of whether the two CPUset modules are in sync. It is responsible for the following:

- Initiating the integration of the second CPUset module at boot-up
- Auto-reintegration of the system on some out-of-sync events
- Auto-reintegration of the system in response to a spurious fault
- Reintegration of the system when a CPUset module is hot-replaced
- Supplying the parameters for reintegration

You interact with the `ft_core` object to find out whether the system is running in combined or single mode. You can also find out if a processor integration is pending or happening and if the system has given up attempting to reintegrate a failing CPUset module automatically. You can set the processor integration schedule and configuration parameters via the `ft_core` object.

5.3.1 The `ft_core` Subsystem

Each `ft_core` subsystem has the generic attributes shown in TABLE 2-2 and the attributes shown in TABLE 5-1.

TABLE 5-1 `ft_core` Subsystem Fields

Field	Values
`*description`	`FT Processor and Core`
`pri_stop_time_msecs`	`200|500|1000|2000`
`pri_timeout_seconds`	`0|20|30`
`pri_cpu_resource_percent`	`30|40|50|60|70|80|90|100`
`pri_start_time`	`now|0000|0100|0200|0300|0400|0500|0600|0700|0800|0900|1000|1100|1200|1300|1400|1500|1600|1700|1800|1900|2000|2100|2200|2300`
`*ft_status`	`null|A_running|B_running|fault_tolerant`
`*op_status`	`out_of_sync|pending_pri|integrating|in_sync`
`*oos_type`	`none|CE|unresolved|fatal|forced`
`*oos_msg`	`Message describing the oos event`
`*split_status`	`null|split|combined`

An asterisk preceding a field name indicates that the field is not user-settable.

`pri_cpu_resource_percent` specifies how much of the system resources (processing power and I/O bandwidth) the reintegration process will use. The possible values range from 30% to 100%. The higher the value the quicker processor reintegration may be; however, this is to the detriment of other processes on the system. The default value is 30%.

`pri_stop_time_msecs` specifies the maximum time (approximately) for which system services will be suspended in the final stages of integration (final PRI). The parameter is measured in milliseconds from 200 to infinity. The higher the number the quicker the CPUset module is integrated; however, this is to the detriment of other processes and devices since no processes can be run and no interrupts can be taken during this time. The default value is 500 milliseconds.

`pri_timeout_seconds` specifies the maximum elapsed time the reintegration process will take before final reintegration is invoked. Normal system services are available during this period.

Note – The clock starts when the target CPUset module accepts a reintegration request. This is normally several minutes after being powered on. If the timeout expires the CPUset module fails to be integrated.

The default time is infinite, in that the process will keep trying until final reintegration can be achieved.

`pri_start_time` is used to specify when the integration process will start. It normally specifies a period when the system is less likely to be busy. The 24-hour clock time format is used for entering the time, that is, 2300 is 11 pm and 0100 is 1 am. The default time to integrate is now, which takes immediate effect.

The preceding variables are called ‘processor reintegration’ variables. This topic is discussed in “Processor Reintegration” on page 5-7.

The `ft_status` attribute reflects the fault tolerant configuration of the system. It is set to `fault_tolerant` when both sides of the system are running in lockstep. In non fault tolerant mode, `ft_status` is set to `A_running` for the CPUset module on side A and `B_running` for the CPUset module on side B.

The `op_status` attribute reflects the status of the processor synchronization:

- `out_of_sync` means that the system has experienced an out-of-sync event and needs to reintegrate. This can occur automatically if the event is of type `CE` (correctable error in memory). Otherwise, user intervention is required, for example, replacement of a CPUset module.
- `pending_pri` means that the system is waiting to reintegrate.
- `integrating` means that integration is in process.
- `in_sync` means that the system is running in lockstep.

The `oos_type` attribute reflects the type of the most recent out-of-sync event. This attribute is `none` when the system is running normally. The other values are shown in TABLE 5-2.

TABLE 5-2 `oos_type` Attribute Values

Value	Description
`CE`	A correctable error has occurred. The system reintegrates without user intervention provided the rate of correctable errors is within a specified threshold.
`unresolved`	An error has occurred. Its cause cannot be identified by software alone.
`fatal`	A fatal error has occurred. User intervention is required to investigate and resolve the problem. This can involve replacing a module, or deciding to try to reintegrate the system.
`forced`	A system user has forced an error.

The attribute oos_msg contains a brief message to help diagnose the most recent out-of-sync event. This is set to none once the system is returned to lockstep.

The attribute split_status is written by the split utility. It reflects whether the system is running in split or combined mode.

5.3.2 The MBD Module

The motherboards are represented in the CMS as A-MBD and B-MBD modules. Unlike other modules, the motherboards can be inserted only in one position, which means that their location attributes are fixed.

Each MBD module has the generic attributes shown in TABLE 2-2 and the attributes shown in TABLE 5-3.

TABLE 5-3 MBD Module Fields

Field	Values
*description	Motherboard A - (Upper) (motherboard A) Motherboard B - (Lower) (motherboard B)
*ambient_temperature	normal|high|low
*i2c_faulty	yes|no

An asterisk preceding a field name indicates that the field is not user-settable.

The A-MBD and B-MBD objects can be disabled only if all their constituent modules are disabled and have their location attributes set to NULL. The constituent modules are as follows:

- RMM
- CAF
- PSU0
- PSU1
- PSU2
- CPU
- DSK
- PCI0
- PCI1
- PCI2
- PCI3
- PCI4
- PCI5
- PCI6
- PCI7

The system user should ensure that the fault tolerant drivers which handle the motherboard are online after the motherboard is repaired.

5.3.3 The CPUset Module

In the CMS, the CPUset module is represented as having the standard states described in Section 2.1.2 “CMS Object States” on page 2-7.

Each `CPUset` module has the generic attributes shown in TABLE 2-2 and the attributes shown in TABLE 5-4

TABLE 5-4 CPUset Object Fields

Field	Values
`*description`	`CPUset FRU`
`*fan0_speed`	`normal|low|high`
`*fan1_speed`	`normal|low|high`
`*cpu0_temperature`	`normal|low|high`
`*cpu1_temperature`	`normal|low|high`
`*cpu2_temperature`	`normal|low|high`
`*cpu3_temperature`	`normal|low|high`
`*ambient_temperature`	`normal|low|high`
`*i2c_faulty`	`no|yes`

An asterisk preceding a field name indicates that the field is not user-settable.

The last eight parameters in in TABLE 5-4 are written by the environmental monitor. See Section 2.5.2 “Environmental Failures” on page 2-13 for details.

The processor reintegration parameters are described in Section 5.3.4 “Processor Reintegration” on page 5-7.

Enabling the CPUset module causes the `ft_core` subsystem to attempt to integrate the processor. Disabling the CPUset module causes the system to become out of sync.

5.3.4 Processor Reintegration

Processor reintegration parameters define how the CPUset module is reintegrated and also how reintegration affects the other processes running on the system. These parameters are as follows:

- `pri_stop_time_msecs`
- `pri_timeout_seconds`
- `pri_cpu_resource_percent`
- `pri_start_time`

These parameters are listed in TABLE 5-1 on page 5-3 and described in the paragraphs that follow that table.

After the reintegration configuration parameters are entered, the CPUset module should be ready to integrate. Select the `action` field and select `enable`. Depending upon the reintegration parameters specified, the system should then start to integrate the CPUset module. This will take from a minimum of several minutes to a maximum of several hours, depending upon the parameters specified and the system load.

5.4 Messages and Indicators

5.4.1 LED Indicators

When a CPUset module has a detected fault, the red *Faul*t LED is lit. Other indications that the CPUset module is not functioning are that the *Diagnostic* (D) LED is not flashing, the *Error* (E) LED is lit, or both. See the *Netra ft 1800 Hardware Reference Manual* for full information about the LEDs on the CPUset module.

5.4.2 Failures and Error Messages

The CPUset module has the standard messages for module failure. See Section 2.5 “Failures and Error Messages” on page 2-12 for details.

5.4.2.1 Out-Of-Sync (OOS) Failures

If a CPUset module becomes faulty, an out-of-sync event occurs. This causes the module to appear in the default displays of the `cmsfix` and `xcmsfix` utilities. The EEPROM of the CPUset module displays any event information returned from the device driver. The `info` attribute of the CPUset module states that the module has gone out-of-sync. This message is also sent to the status log

The `ft_status` attribute of the `ft_core` system changes to `A_running` if B-CPU fails, or to `B_running` if A-CPU fails.

The `op_status` attribute changes from `in_sync` to `pending_pri`.

If `ft_core` module is configured in such a way that it decides not to reintegrate, the `op_status` attribute then changes from `pending_pri` to `out_of_sync`.

If `ft_core` module is configured in such a way that it reintegrates the system, but the reintegration fails, the `op_status` attribute changes from `pending_pri` to `out_of_sync`.

If the reintegration succeeds, the `op_status` attribute changes from `pending_pri` to `in_sync`. The `ft_status` attribute of the `ft_core` system changes to `fault_tolerant.`

5.5 Maintenance

If a CPUset module needs to be removed for repair or maintenance it must be disabled by the CMS. To disable a CPUset module prior to removal, the `action` attribute should be set to `disable.` See Section 4.3.2 “Removing a Module” on page 4-9 for details.

CHAPTER 6

Power Subsystem

This chapter contains software reference information for the power subsystem of a Netra ft 1800 system. The power subsystem provides a configurable fault tolerant power supply appropriate to the requirements of the installed system.

6.1 Hardware and Environment

The Netra ft 1800 is powered by a minimum of four dual-input power supplies, suitable for use with either a –48V (SELV) or –60V (TNV-2) nominal DC electrical supply. Each side is powered by up to three power supply units (PSUs) in 2+1 redundant configuration. The mnemonics for the PSU locations, PSU0, PSU1 and PSU2, appear on the Netra ft 1800 power distribution board.

For information about system and electrical installation refer to the *Netra ft 1800 Installation Guide* (Part No. 805-4533-10).

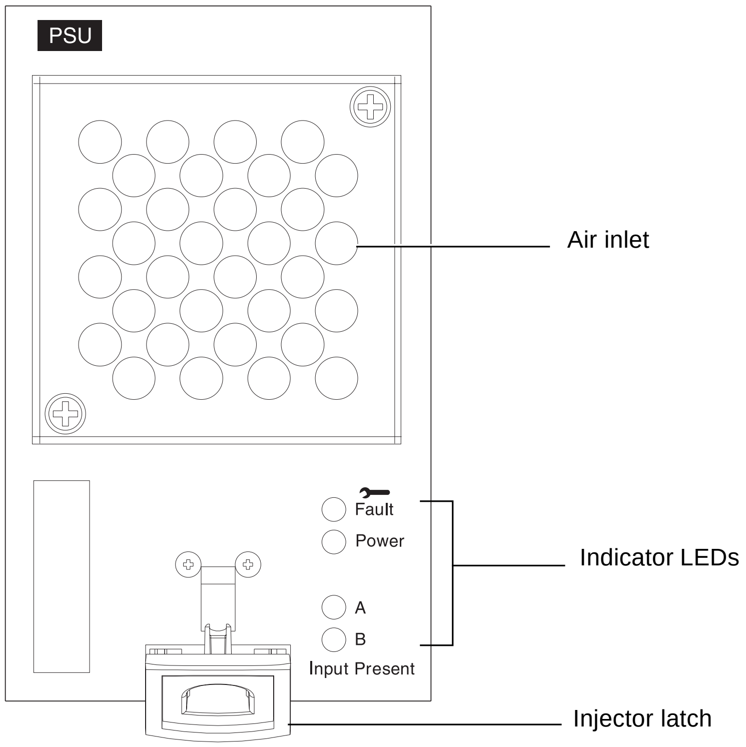


FIGURE 6-1 Power Supply Module Front Panel

6.2 The PSU Module

The PSU is represented in CMS as a PSU object. Each PSU has the standard states, as described in Section 2.1.2 “CMS Object States” on page 2-7. The object has the following fields:

TABLE 6-1 PSU Object Fields

Field	Values
`*description`	`Power Supply Unit`
`*inputA_okay`	`yes|no`
`*inputB_okay`	`yes|no`
`*cpu_rail_okay`	`yes|no`

TABLE 6-1 PSU Object Fields *(Continued)*

Field	Values
`*mbd_rail_okay`	`yes|no`
`*internal_shutdown`	`no|yes`
`*fan_speed`	`normal|low|high`
`*ambient_temperature`	`normal|low|high`
`*i2c_faulty`	`no|yes`

An asterisk preceding a field name indicates that the field is not user-settable.

The system attributes identify environmental and other problems with the PSU.

The attribute `inputA_okay` is set to `yes` when power is present on input A of the PSU, and to `no` when it is not. The attribute `inputB_okay` is set to `yes` when power is present on input B of the PSU, and to `no` when it is not.

The attribute `cpu_rail_okay` is set to `yes` when power is present on the rails to the CPUset module, and to `no` when it is not. The attribute `mbd_rail_okay` is set to `yes` when power is present on the rails to the motherboard, and to `no` when it is not.

The attribute `internal_shutdown` is set to yes if the PSU has shut down for internal reasons.

The remaining system attributes are written by the environmental monitors.

6.3 Messages and Indicators

When a PSU has a detected fault, the red *Fault* LED is lit.

6.4 Maintenance

There are no user-serviceable parts in the system rear access area or the PSUs themselves. User power system maintenance is limited to replacement of a failed PSU. Refer all power system maintenance queries to your local support organization.

CHAPTER 7

Fault Tolerant Network Subsystem

This chapter describes the Netra ft 1800 fault tolerant network subsystem. This provides network connections for the system.

7.1 Hardware and Environment

The fault tolerant network interfaces of a Netra ft 1800 are implemented using pairs of network controllers. The network controllers that make up such a pair are either:

- The motherboard Ethernet controllers which have their physical interfaces located in the CAF module.
- Network controllers in PCI modules. See Chapter 10 “PCI Module Subsystem” for details of the PCI modules.

7.1.1 Hardware Distribution

FIGURE 7-1 shows which slot locations are connected to each PCI bus.

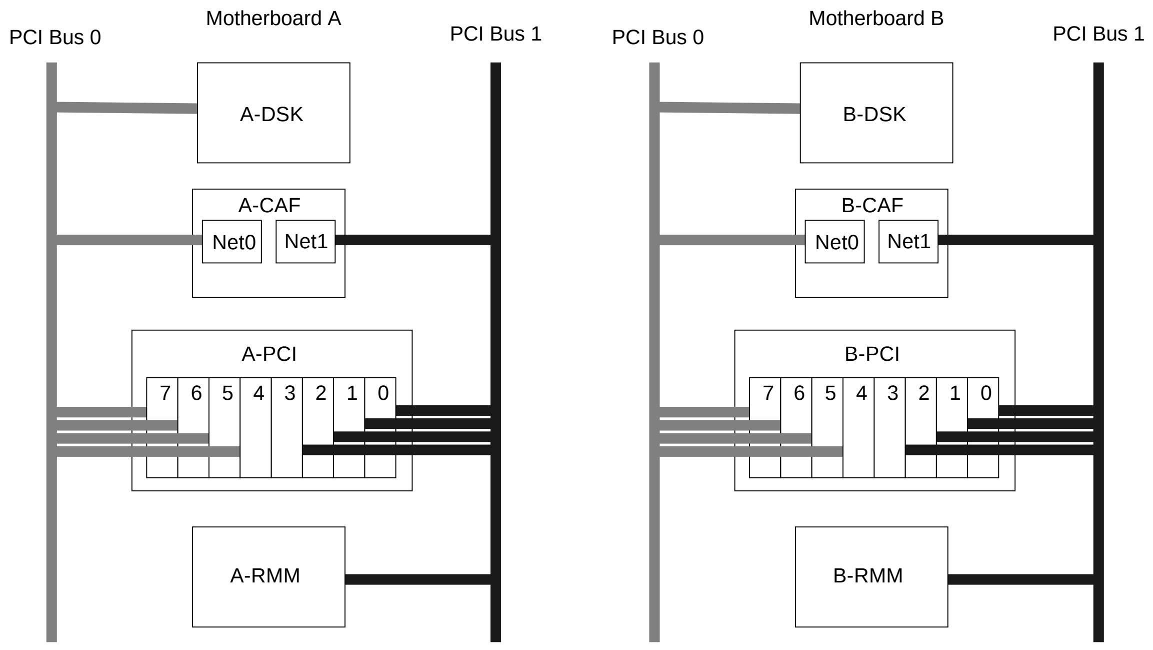


FIGURE 7-1 Slot Location / PCI Bus Distribution

It is recommended that each bus be evenly loaded in order to optimize performance.

For instance, as the RMM is accessed far less than DSK, it makes sense to use the Net1 Ethernet connection on the CAF before using the Net0 connection. Equally, the PCI slots should be populated starting at PCI0 rather than PCI7.

Another consideration is that in order not to compromise fault tolerance, the PCI cards should be scattered rather than grouped. For instance, if eight PCI cards are configured in total, consider placing two on each bus instead of four on each of two buses. Thus the slot locations might be A-PCI0, A-PCI2, A-PCI4, A-PCI6, and the corresponding slots on motherboard B.

7.2 Network Objects

The network subsystem is represented by an `ft_network` object. This does not correspond to a specific module. It is configured in software to reference specified network controllers, which may be either Ethernet controllers in the CAF module or PCI controllers in PCI modules.

7.2.1 Fault Tolerant Network Subsystem

The fault tolerant configuration of network interfaces is managed by the CMS `ft_network` object.

The `ft_network` subsystem has fields as shown in TABLE 7-1.

TABLE 7-1 `ft_network` Subsystem Fields

Field	Values
`*description`	`network multiplexor`
`user_label`	user defined name
`hostname`	hostname for the system
`preferred_controller`	`none|A|B`
`ControllerA_FRU`	`A-CAF0|B-CAF0|A-PCI[0-7]|B-PCI[0-7]|NULL`
`ControllerA_Funct`	`null|Net_0|Net_1|Funct_0|Funct_1|Funct_2|Funct_3`
`ControllerB_FRU`	`A-CAF0|B-CAF0|A-PCI[0-7]|B-PCI[0-7]|NULL`
`ControllerB_Funct`	`null|Net_0|Net_1|Funct_0|Funct_1|Funct_2|Funct_3`
`*devpath`	The pathname of the network device currently in use.
`*link`	The speed of the link currently in use.
`*transceiver`	`internal|external`
`*redundancy`	`not_present|null|non_redundant|dual_redundant`
`*usable_controllers`	`none|A|B|A&B`
`*controller_in_use`	`none|A|B`

An asterisk preceding a field name indicates that the field is not user-settable.

The `ft_network` object references two controllers. The module which contains the controller for each side is specified in the `ControllerA_FRU` and `ControllerB_FRU` fields. The specific controller within the module is specified in the `ControllerA_Funct` and `ControllerB_Funct` fields. The value of each of these fields is `Net_0` or `Net_1` if a CAF module is specified, and `Funct_0` through `Funct_3` if a PCI module is specified.

If you want to change a configuration, you change the `Controller`<*letter*>`_FRU` field first, then change the `Controller`<*letter*>`_Funct` field. The change takes effect only when the `Controller`<*letter*>`_Funct` field is modified.

The `hostname` is used to set up the IP address for the interface. If the hostname given is invalid, the `ft_network` object goes to the state `online_down`. The `info` attribute displays an error message. If the value in `hostname` is valid, the interface is configured and the object goes to the state `online_up`.

When `ft_network` services are created at install time, the value of `hostname` is taken from the `uname` of the system at installation.

The `preferred_controller` allows you to specify a preference between the specified controllers, taking into account external factors, such as the speed of the external network. By default, the `preferred_controller` attribute is set to `none`. You can configure this to either A or B before or after configuring the controllers, while the interface is in any state.

The `controller_in_use` attribute shows which controller is in use. By default, the controller in use will be set to the first controller linked into the service. However, the driver can automatically change it if a problem is detected with the controller.

The `redundancy` attribute reflects the current redundancy of the `ft_network` subsystem. If its value is `not_present`, the subsystem is not configured and cannot come online. If the value is `null`, none of the configured controllers is enabled. If the value is `non_redundant`, only one controller is configured and enabled. If the value is `dual_redundant`, both controllers are configured and enabled. The subsystem can be `online_up` or `online_down` if the attribute is `non_redundant` or `dual_redundant`.

The `ft_network` subsystem has states as shown in TABLE 7-2.

TABLE 7-2 `ft_network` Subsystem States

State	Means
`not_present`	The subsystem cannot be brought online.
`online_down`	The subsystem has redundant controllers configured and enabled, but no hostname.
`online_up`	The subsystem has redundant controllers configured and enabled, and a hostname.

The controllers referenced by the `ft_network` subsystem must be unconfigured before the `ft_network` subsystem can be disabled.

7.2.2 Motherboard Network Controllers

See Section 9.1.2 “CAF Module” on page 9-3 for information about the motherboard network controllers, which interface via the CAF front panel.

7.3 Configuring Network Controllers

A Netra ft 1800 has two network controllers on each motherboard. There are therefore four motherboard network controllers available for use in a combined system, and two in a split system. The motherboard network controllers come online when the motherboard is powered on. They can be used to boot the system.

In addition, there can be up to eight plug-in controllers contained in the slots in the PCI modules. They come online only when the CMS has started. They cannot be used to boot the system. The PCI controllers communicate via their own module.

The motherboard network controllers interface via the CAF module on the same side of the system. The CAF module must be present and have power for any network connection to be established using the motherboard controllers.

A fault tolerant network interface should never use a pair of network controllers that communicate via the same module. A network connection through that interface could not be maintained if the module fails. If the fault tolerant network interface uses a pair of controllers that communicate via different modules, a network connection would be maintained in the event of failure or removal of a single module. Any appropriate combination of controllers can be used. For example:

- One motherboard controller on each side of the system:

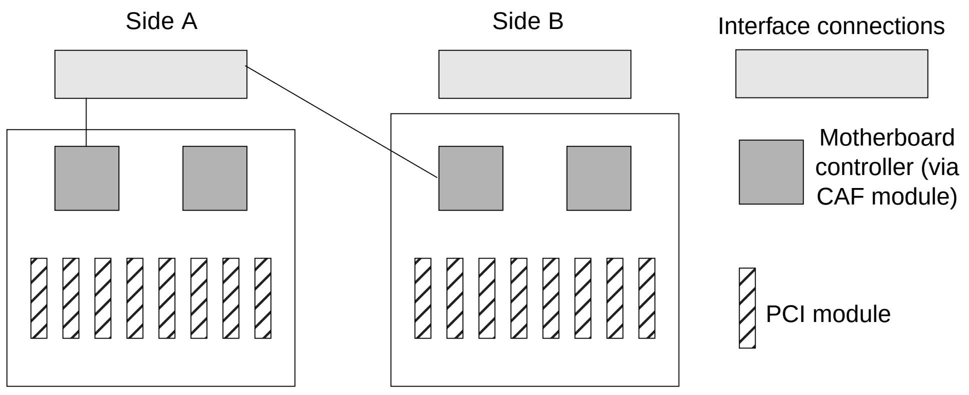


FIGURE 7-2 Network Redundancy on Separate Sides

- One motherboard controller and one PCI controller on the different sides of the system:

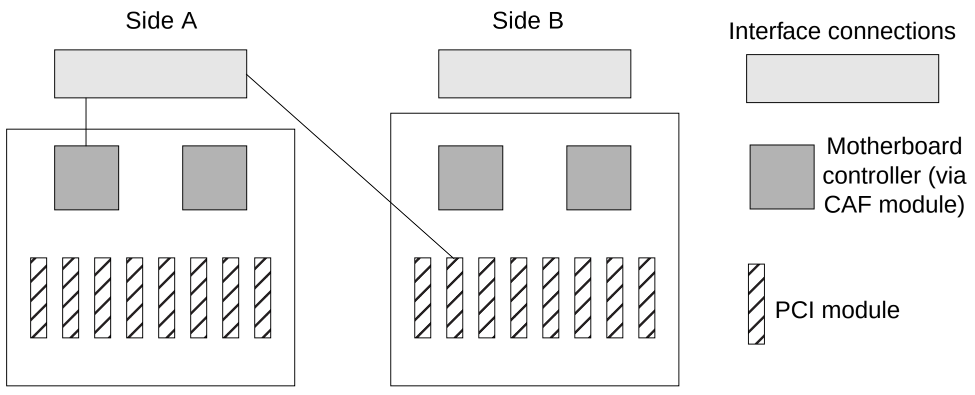


FIGURE 7-3 PCI Redundancy on the Same Side

PCI controllers on different sides (or on the same side) can also be used redundantly, or a motherboard controller on one side and a PCI controller on the other side can be used.

7.4 Configuring `ft_network`

This procedure explains how to configure fault tolerant Ethernets using `cmsconfig`.

1. **Ensure you have a hostname declared in the `/etc/hosts` file.**
2. **Use `cmsconfig` to configure and enable the PCI Ethernet cards that will be used for the fault tolerant network.**

   To be fault tolerant, the interfaces should reside on opposite sides of the machine.

   Note – The CAF Ethernet interfaces are automatically configured and enabled at boot time.

   ---

3. **Include the new `ft_network` object into `cmsconfig`.**

   Type the following at the `cmsconfig` prompt:

   ```
   i ft_network x
   ```

4. **Enter the location and functional interface details for the first network card into the `ControllerA_FRU` and `ControllerA_Func` attributes.**
5. **Enter the details for the second network interface into the `ControllerB_FRU` and `ControllerB_Func` attributes.**
6. **Enter the hostname defined in Step 1 into the `hostname` attribute.**

   No further action is required, as once the CMS detects a valid `hostname` attribute field it will attempt to configure the `ft_network` into the `online_up` state.

7.5 Messages from `ft_network`

Note – When a module has a detected fault, its red *Fault* LED is lit.

The `ft_network` subsystem can write the following messages to the status log:

```
ERROR: failed to link _controllerA
ERROR: failed to link _controllerB
```

The linking of the controller failed, possibly because the controller failed during the linking operarion.

```
WARNING: could not switch controllers
```

The preferred controller cannot be switched into use because it has been determined to be in an unsuitable state (degraded or failed).

```
WARNING: unable to set address
```

The hostname chosen cannot be resolved to an IP address.

```
WARNING: unable to bring interface up
```

The hostname clashes with the hostname of another network interface on the machine.

`WARNING: controller already registered to service` *service*

The service *service* is already using this controller. A controller cannot be used by more than one service simultaneously.

```
WARNING: must have at least one controller
```

An attempt was made to unconfigure the controller that is in use by a non-redundant `ft_network` object in the state `online_up`. The state of the `ft_network` object must be `online_down` or `offline` before this can be done.

```
NOTICE: hostname is not set
```

This notice is given when the state changes from `offline` or `not_present` to `online_down` as a reminder that the hostname needs to be set before the state will go to `online_up`.

```
NOTICE: re-spawning daemon
```

The `ft_network` daemon was killed. The CMS has restarted it.

7.6 Fault Management

When controllers associated with an interface become unusable, the `redundancy` attribute changes to reflect the new level of service.

If a controller becomes unusable then its association with an interface may be broken using `cmsconfig`. A new controller may then be associated to return the interface to its previous level of service.

CHAPTER 8

Drive Subsystems

This chapter contains software reference information for the disk and removable media modules (RMM) of a Netra ft 1800 system. The disk module (DSK) (sometimes referred to as the *disk chassis*) provides the internal physical storage for the system. The RMM typically contains one CD-ROM drive and can have an optional DAT tape drive.

8.1 Disk Chassis

The Netra ft 1800 has a disk chassis (DSK) on each side into each of which up to six hard disk drive (HDD) modules can be inserted. The HDD modules are Ultra SCSI (Fast-20)-based disk drives.

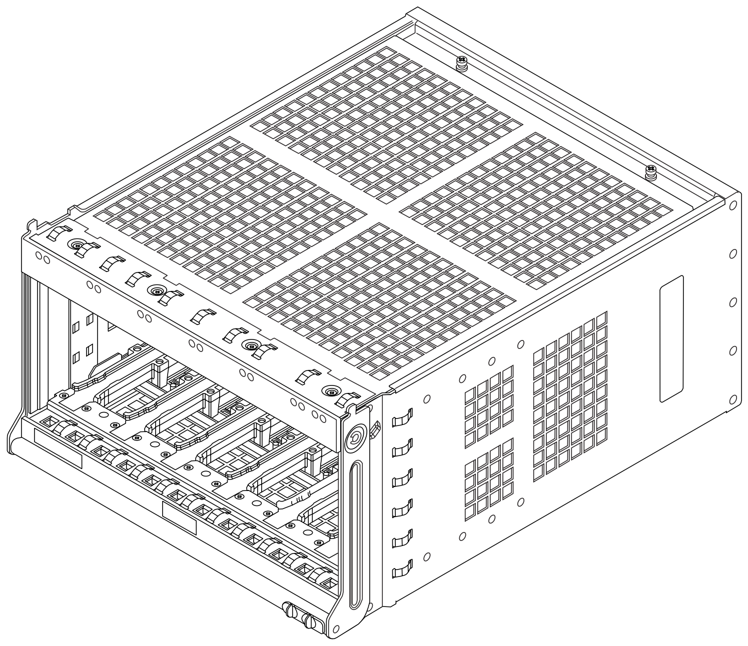


FIGURE 8-1 Disk Chassis (DSK)

The disk chassis is represented in the CMS as a DSK object. Each DSK object has the standard states, as described in Section 2.1.2 "CMS Object States" on page 2-7. The object has the following fields:

TABLE 8-1 DSK Module Fields

Field	Values
`*description`	`Disk Chassis`
`*ambient_temperature`	`normal|low|high`
`*12c_faulty`	`no|yes`

An asterisk preceding a field name indicates that the field is not user-settable.

The DSK object can be unconfigured only if all the constituent HDD objects are unconfigured and have their `location` attributes set to NULL.

8.1.1 Hard Disk Drive Modules

The disk drives are represented in CMS as HDD objects. Each HDD object has the standard states, as described in Section 2.1.2 “CMS Object States” on page 2-7. The object has the following fields:

TABLE 8-2 HDD Module Fields

Field	Values
`*description`	`Hard Disk Drive FRU`
`checksumming`	`yes|no`
`*Disk`	Name of disk

An asterisk preceding a field name indicates that the field is not user-settable.

You set the `checksumming` attribute to `yes` to switch on end-to-end data checksumming for the disk.

The `Disk` attribute contains the system name of the disk. You need this name to identify the disk to Volume Manager, or to other applications that use Solaris system names.

An HDD object can be disabled only if the disk is not in use.

8.1.2 Configuring Disks

An HDD module must be enabled once it is inserted in the disk chassis. See “Adding a Module to the CMS” on page 4-7 for an example of how to enable a new module using `cmsconfig`.

8.1.3 Fault Tolerance Management

Disk fault-tolerance is provided via Sun StorEdge Volume Manager. See Chapter 3 “Sun StorEdge Volume Manager” for details.

8.1.4 Messages and Indicators

When the disk chassis has a detected fault, the chassis red *Fault* LED is lit. When a disk drive has a detected fault, the drive red *Fault* LED is lit.

8.2 Removable Media Module (RMM)

The RMM provides drives for removable storage media.

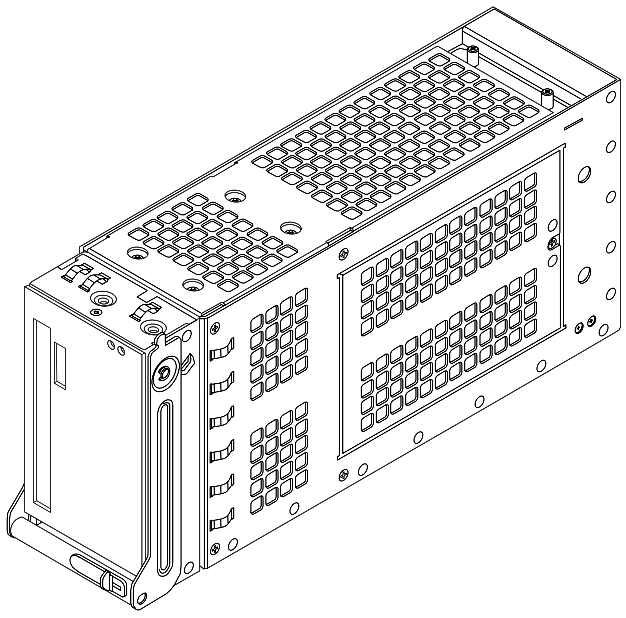


FIGURE 8-2 Removable Media Module Chassis (RMM)

8.2.1 Hardware and Environment

The Netra ft 1800 has a slot for an RMM on each side.

The RMM has locations for two drives. It can contain a CD-ROM drive or a CD-ROM drive and a DAT tape drive. The type of drive is irrelevant to the CMS.

8.2.2 RMM Module

The RMM is represented in CMS as a RMM module. Each RMM object has the standard states, as described in Section 2.1.2 “CMS Object States” on page 2-7. The object has the following fields:

TABLE 8-3 RMM Object Fields

Field	Values
`*description`	`Removable Media Module`
`*Funct_0`	`Name of device`
`*Funct_1`	`Name of device`
`*ambient_temperature`	`normal|low|high`
`*i2c_faulty`	`normal|low|high`

An asterisk preceding a field name indicates that the field is not user-settable.

The `Funct_0` and `Funct_1` attributes contain the system names of the devices in the RMM. `Funct_0` is always the first CD-ROM drive. `Funct_1` is the second device.

The RMM object can be unconfigured only if the drivers in the slots are not in use.

8.2.3 Configuring RMMs

An RMM must be enabled once it is inserted. See “Adding a Module to the CMS” on page 4-7 for an example of how to enable a new module using `cmsconfig`.

8.2.4 Messages and Indicators

When the RMM has a detected fault, the module’s red *Fault* LED is lit.

CHAPTER 9

Console, Alarms and Fans

This chapter describes the console, alarms and fans (CAF) module and the environmental monitoring facility that it provides.

9.1 The CAF Module

The CAF module provides the following:

- Power On and Standby switches
- Main cooling fans for the system
- Alarms interface
- Serial I/O interfaces
- Ethernet interfaces

The switches, LEDs and fans are described in the *Netra ft 1800 Hardware Reference Manual*. The system I/O ports are part of the I/O subsystem, which is described in Chapter 7 “Fault Tolerant Network Subsystem”.

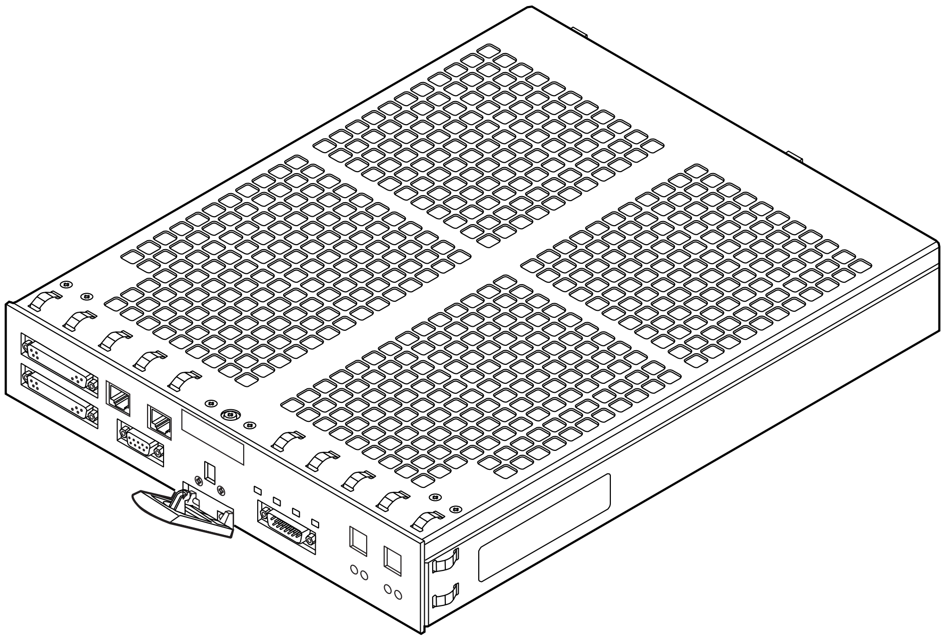


FIGURE 9-1 Console, Alarms and Fans Module (CAF)

This chapter describes the generic features of the CAF module, plus the dual console, the environmental monitor and the alarms.

9.1.1 Hardware and Environment

The CAF includes the hardware that provides the functionality described in this chapter. See the descriptions of the individual components.

The CAF module has its own *Power* and *Fault* LEDs, and repeater *Power* and *Fault* LEDs for the motherboard.

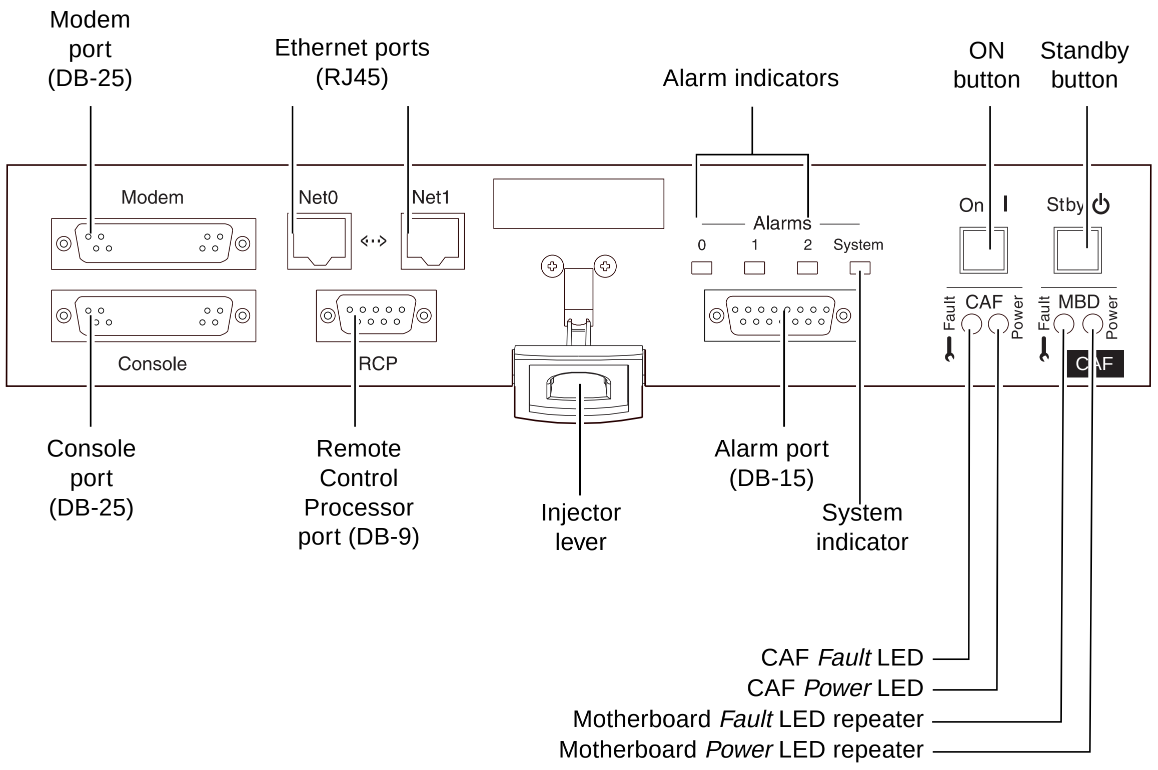


FIGURE 9-2 CAF Front Panel

9.1.2 CAF Module

The CAF is represented in CMS as a CAF object. Each CAF has the standard states, as described in Section 2.1.2 “CMS Object States” on page 2-7. The object has the following attributes:

TABLE 9-1 CAF Module Fields

Field	Values
`*description`	`Console, Alarms and Fans FRU`
`*Console`	Console address
`*Modem`	Modem address

TABLE 9-1 CAF Module Fields *(Continued)*

Field	Values
`*Net0`	Network interface
`*Net1`	Network interface
`*fan0_speed`	`normal|low|high`
`*fan1_speed`	`normal|low|high`
`*fan2_speed`	`normal|low|high`
`*fan3_speed`	`normal|low|high`
`*ambient0_temperature`	`normal|low|high`
`*ambient1_temperature`	`normal|low|high`
`*alarm_0`	Alarm message
`*alarm_1`	Alarm message
`*alarm_2`	Alarm message
`*system_alarm`	Alarm message
`*i2c_faulty`	`no|yes`

An asterisk preceding a field name indicates that the field is not user-settable.

The `Net0` and `Net1` attributes correspond to the labels on the CAF module. The values for these attributes are set up when the CAF is enabled. The CAF attempts to link these network controllers to `ft_network` services to which they are assigned. It must prevent the same controller being linked to more than one service. When the CAF is disabled it attempts to unlink these network controllers from the `ft_network` services. The CAF module can have either of its controllers already linked as part of an `ft_network` service before the CMS starts up at boot time. If the CAF module is not present, and thus not enabled by the CMS, the `ft_network` object will not represent any links that have already been performed. See Section 7.2.1 “Fault Tolerant Network Subsystem” on page 7-3 for details of the `ft_network` object.

The possible values of `alarm_0` through `alarm_2` depend on the fault tolerant status of the alarm subsystem. In fault tolerant mode, the attributes have the values `on` or `off`, reflecting the status of the corresponding alarm. In non fault tolerant mode, the attributes on side B reflect the identities of the alarms assumed by alarm devices as well as their status: `alarm_0` takes the role of `alarm_3`, `alarm_1` takes the role of `alarm_4` and `alarm_2` takes the role of `alarm_5`.

The remaining attributes are written by the environmental monitors.

9.1.3 Messages and Indicators

When a CAF has a detected fault, the red *Fault* LED is lit.

9.2 Consoles

In split mode, each side of a Netra ft 1800 system has its own console. In fault tolerant mode, messages from both sides are displayed on both consoles. The consoles are controlled from the CMS.

9.2.1 Console Subsystem

The consoles are represented in the CMS as the `sm` subsystem. The `sm` subsystem has the following attributes:

TABLE 9-2 Console Subsystem Fields

Field	Values
`*description`	`Serial multiplexer`
`port0`	`NULL|A-CAF:console|A-CAF:modem|B-CAF:console|B-CAF:modem`
`port1`	`NULL|A-CAF:console|A-CAF:modem|B-CAF:console|B-CAF:modem`
`port2`	`NULL|A-CAF:console|A-CAF:modem|B-CAF:console|B-CAF:modem`
`port3`	`NULL|A-CAF:console|A-CAF:modem|B-CAF:console|B-CAF:modem`
`*redundancy`	`null|not_present|non_redundant|dual_redundant|triple_redundant|quad_redundant`
`*usable_controllers`	Names of usable controllers
`*mode_of_use`	`unused|/dev/console|/dev/term/0|/dev/term/1|/dev/term/2`
`*ports_in_use`	List of ports in use

An asterisk preceding a field name indicates that the field is not user-settable.

You set the attributes `port0` through `port3` to specify the device that uses the corresponding port.

The `redundancy` attribute reflects the current redundancy of the `sm` subsystem. If its value is `not_present` or `null`, the subsystem is not configured to be redundant and cannot come online. If the value is `non_redundant`, the subsystem is configured to be redundant, but the controllers are not enabled. It will come online when the controllers are enabled. If the value is `dual_redundant`, the subsystem will come online and run in fault tolerant mode with two available ports. If the value is `triple_redundant`, the subsystem will come online and run in fault tolerant mode with three available ports. If the value is `quad_redundant`, the subsystem will come online and run in fault tolerant mode with four available ports.

The `sm` subsystem has states as shown in TABLE 9-3.

TABLE 9-3 Console Subsystem States

State	Means
`not_present`	The subsystem cannot be brought online.
`busy`	The subsystem is in transition between states.
`offline`	The subsystem has controllers configured but not enabled.
`online`	The subsystem has redundant controllers configured and enabled.

The controllers referenced by the `sm` subsystem must be unconfigured before the `sm` subsystem can be disabled.

9.2.2 Serial I/O

The system contains four UARTs that provide serial I/O via the CAF module. Software provides the facility for multiplexing character streams from these UARTs. When two or more UARTs provide a logical stream, input from any device is multiplexed onto a single stream to the user. When characters are written to the stream they are passed to each UART in the logical set. This facility is used to provide access to the system console from multiple ports.

In a non-split system the default console uses the console port, known as port A, on each of the system CAFs. This behavior is available both within OBP and in UNIX.

9.2.3 Console Configuration

The console can be booted in dual or non-dual mode.

9.2.3.1 Booting a Dual Console

Dual console is configured within OBP by setting the input and output devices to the `u4ftser` node. At the OBP prompt, enter the following:

```
ok> setenv input-device /u4ftser
ok> setenv output-device /u4ftser
ok> reset
```

The system will then boot dual console with console output appearing on port A of both CAFs when the system is non-split. The file system entry for the console can be accessed from either `/dev/console` or `/dev/u4ftser:con`.

9.2.3.2 Booting a Non-Dual Console

One side of a split system can be booted with a non-dual console by setting the input device to one of those shown in TABLE 9-4:

TABLE 9-4 Input Device Settings

Device	Port
`u4-a-ttya`	port A on A-CAF
`u4-a-ttyb`	port B on A-CAF
`u4-b-ttya`	port A on B-CAF
`u4-b-ttyb`	port B on B-CAF

9.3 Alarms

The Netra ft 1800 provides general-purpose alarms and a UNIX-running watchdog which enables applications to indicate a failure condition according to their failure criteria.

The alarms and watchdog functionality is provided by two registers on each motherboard controller, one per motherboard. The alarms are relays that can be used to indicate a fault or exceptional condition and are toggled by writing to bits in the general purpose alarms register on the motherboards. The watchdog timer is reset by writing to the "pat" register within the specified timeout interval.

The outputs for the alarms and watchdog are brought out to LEDs in the CAF on each side. The general-purpose *Alarm* LEDs are amber while the watchdog is monitored by the *System* LED, which is green. The CAF houses the relays.

The alarms can be treated as a fault tolerant pair where alarms 0 through 2 are toggled on both motherboards simultaneously, or they can be treated as non fault tolerant where alarms 0 through 5 are toggled independently on either motherboard. In non-fault tolerant mode, alarms 0 through 2 "belong" to side A (the upper motherboard) and alarms 3 through 5 "belong" to side B (the lower motherboard). The default is to treat the alarms as a fault tolerant pair. The UNIX-running watchdog is a hardware timer that is patted (reset) simultaneously on both motherboards irrespective of the mode of operation. If the watchdog triggers, the *System* LED will be turned off.

9.3.1 Alarms Subsystem

The alarms subsystem is represented in the CMS by the `ft_alarm` object and by the alarm attributes of the CAF object (see TABLE 9-1 on page 9-3).

You use the `ft_alarm` object to control the mode of operation of the alarms subsystem and view its current operational status.

The `ft_alarm` object attributes are shown in TABLE 9-5.

TABLE 9-5 `ft_alarm` Subsystem Fields

Field	Values
`*description`	`alarms multiplexor`
`req_mode`	`FT|non-FT`
`*status`	`redundant|non-redundant`

An asterisk preceding a field name indicates that the field is not user-settable.

Set the `req_mode` attribute to `FT` to send all alarms output to both CAFs. Set the `req_mode` attribute to `non-FT` to send alarms output from each side to the CAF on that side.

The `status` attribute reflects the operational status of the alarms subsystem. When the subsystem is configured to use both sides and one side is not operational, the driver will automatically change the status attribute to `non-redundant`.

TABLE 9-6 shows the `ft_alarm` subsystem states.

TABLE 9-6 `ft_alarm` Subsystem States

State	Means
`unusable`	The alarm device is not available.
`usable`	One or more alarm devices are available. The subsystem will come online when required.

CHAPTER **10**

PCI Module Subsystem

PCI modules contain one or more network controller PCI cards. They have fields as shown in TABLE 10-1.

TABLE 10-1 PCI Module Fields

Field	Values
`*description`	`PCI Carrier`
`*Funct_0`	The system name of the device.
`*Funct_1`	The system name of the device.
`*Funct_2`	The system name of the device.
`*Funct_3`	The system name of the device.
`*ambient_temperature`	`normal|low|high`
`*i2c_faulty`	`no|yes`

An asterisk preceding a field name indicates that the field is not user-settable.

The last two attributes in TABLE 10-1 are written by the environmental monitor.

The attributes `Funct_0` through `Funct_3` correspond to the devices within the PCI module. These devices are read from the EEPROM of the module when it is enabled. The module attempts to link these network controllers to `ft_network` services to which they are assigned. When the module is disabled it attempts to unlink these network controllers from the `ft_network` services. The PCI module never has any of its controllers already linked as part of an `ft_network` service, before the CMS starts up at boot time.

The network controller in a PCI module will come online when that module is enabled.

CHAPTER 11

Split Mode

Split mode minimizes the need for service outages on a Netra ft 1800 system resulting from software upgrades to the operating environment or to applications.

The Netra ft 1800 architecture enables a system running in fault tolerant mode to be split into two independent domains, referred to as *sides*. One side, referred to as the *winning* or *surviving side*, retains the identity of the original system and continues running the same services without interruption. The other side, referred to as the *losing* or *new side*, can be booted with a new system identity.

When the system is split, it is possible to use one side (the losing side) to test the upgrade procedures and services while the other side continues to provide the system services uninterrupted. This removes the possibly lengthy test and verification cycle from the upgrade procedure and reduces the overall service unavailability.

When split, the system consists of two domains, each with dedicated resources. Each domain contains:

- One CPUset module, containing up to four processors
- One motherboard, containing two PCI buses (0 and 1) with connections for four I/O slots, and an EEPROM which is accessible via an I^2C maintenance bus

Each motherboard contains two PCI bridges (one per bus) that provide the interface between the CPUset module and the I/O peripherals. When in split mode, each bridge provides standard PCI bridge functionality and enforces resource ownership, as each is connected to both CPUset modules.

Each side owns available resources by owning slots that can contain modules. The slots can be either *fixed* (ownership of the slot cannot be transferred between sides) or *movable* (ownership of the slot can be transferred between sides).

- Slots that contain PSU, CAF, RMM, MBD and CPUset modules are fixed slots and are always owned by the side to which they are physically attached.
- Slots that contain DSK and PCI modules are movable slots and can be assigned to either side, both before splitting the system and while it is in split mode. The individual HDD modules in a DSK module all belong to the same side as the DSK module.

  Either side can therefore take ownership of a DSK or PCI I/O resource regardless of where the module is located. Note that:
- Ownership of slots cannot be shared between sides.
- The ownership of a slot is remembered but not applied when the system is configured to run in combined or fault tolerant mode.

One side of the Netra ft 1800 system is defined as the *split master* or *master*. You can specify which side is the master before or after splitting the system. The master can always initiate the transfer of ownership of a movable module from one side to the other. The other side can initiate a transfer of ownership only if it is able to negotiate the change with the master. The split mode operation is managed by a daemon, `splitd`, an instance of which runs on each side. The daemons communicate via the Inter-CPUset Network (`icn`) or the regular network connections.

Information about the mode, master and module ownership of a system persists across a reboot of either side. That is, if one or both sides are rebooted:

- a split system remains split
- the same side remains as the master
- modules continue to belong to the same side

Split mode operations are adminstered by the system administrator using the following utilities:

- `splitinfo` - gives the status of split configuration
- `splitconf` - changes the split configuration, including slot ownership
- `splitadm` - performs the split and merge operations

11.1 Splitting a System

When a fault tolerant system is split, one side, referred to as the *winning* side, continues running without interruption, retaining the identity of the original fault tolerant system. The other side can be rebooted by the system administrator and must have a new identity.

You must use Sun StorEdge Volume Manager to reconfigure the disk volumes for the newly created system. See “To Prepare the Disks” on page 11-4 for details.

You split the system either to upgrade the operating environment, or to upgrade applications on the system. If you upgrade the operating environment, you may need to reconfigure the disk partitions on each side to reflect the requirements of the upgraded operating environment. See the installation instructions for the operating environments for details of the disk partitions which it requires.

11.1.1 Before You Split the System

The split mode software must be installed before you try to split the system. The split mode software consists of the packages SUNWspltr and SUNWspltu. The split mode software, and in particular the split daemon `u4ftsplitd`, starts at the first reboot after the package installation.

See the *Netra ft 1800 Installation Guide* for details of software installation.

You can prepare the configuration of the new system so that it will boot with the required components. This involves the following tasks, described in this section:

- editing the `split` daemon and `icn` configuration file (page 11-3)
- preparing the disks under Sun StorEdge Volume Manager control (page 11-4)
- specifying the split master (page 11-4)
- assigning the ownership of the modules (page 11-5)
- quiescing the devices (page 11-6)

▼ To Edit the Daemon and `icn` Configuration Files

You must edit the `u4ftsplitd` configuration file before you split the system. In particular, if the boot disk is mirroring under Sun StorEdge Volume Manager, you must edit the `hostnames` line to specify the hostnames of the two split systems. If you choose `icn` as the interface for inter-daemon communication, you must configure the `icn` driver. See Section 11.4.1 “Configuring the `icn` Driver for Split Mode Operation” on page 11-24 for information on how to do this.

The configuration file for `u4ftsplitd` is `/etc/splitd.conf`.

See the `splitd.conf`(4) man page for details of how to configure the split daemon.

Once you have modified the configuration file, enter the following commands to restart the split daemon with the new configuration:

```
# /etc/init.d/u4ftsplit stop
# splitinfo
```

▼ To Prepare the Disks

If Volume Manager is not used, no action is required.

If Volume Manager is used:

1. **Ensure that the boot disk consists of two mirrored disks, each of which resides on a different side of the system (in a different disk chassis).**
2. **Remove from Volume Manager control all the non-boot disks in the disk chassis that will belong to the losing side.**

   The boot disk *must* remain as an active mirror under Volume Manager control. See Chapter 3 “Sun StorEdge Volume Manager” for details.

▼ To Specify the Split Master

You must identify which side of the system will be the master when the system is split. It is simpler to make the surviving side the master.

To see which side is currently the split master, enter the following command:

```
# splitinfo -m
```

If side A is the master, the system will respond with a message similar to the following:

```
# master = A
```

To change the current split master from A to B, enter the following command:

```
# splitconf -m B
```

See the manual pages for `splitinfo`(1M) and `splitconf`(1M) for full details of these commands.

▼ To Assign the Ownership of the Modules

You can assign ownership of slots that contain movable modules to one of the sides before the system is split.

To view the current ownership of all the slots in the system, enter the following command:

```
# splitinfo -l all
```

To change the ownership of an slot, use the command `splitconf`(1M) with the `-l` and `-o` options, as follows:

```
# splitconf -l location -o [A|B]
```

For example, assume that the `splitinfo` command displays the following list:

```
A-CAF A Fixed
A-PCI0 A Movable
A-PCI1 A Movable
.
.
B-PSU1: B Fixed
B-PSU2: B Fixed
```

The slot A-PCI0 can belong to side B because a PCI is a movable module. In contrast, the slot A-CAF must be owned by side A, the side to which it is physically attached, because the CAF is a fixed module.

To change the current owner of A-PCI0 from A to B, enter the following command:

```
# splitconf -l A-PCI0 -o B
```

To view the current ownership of a the slot, enter the following command:

```
# splitinfo -l A-PCI0
```

The output from this command should now be:

```
A-PCI0 B Movable
```

Note – You must modify the fault tolerant networks that are configured on a combined system before the split command is issued so that they do not include a physical device that will be owned by the losing side. Furthermore, you must have at least one usable network interface that will belong to the loser of a split operation and this interface must be the primary interface of the system, that is, the interface that has its host name equal to the node name.

▼ To Quiesce Devices

You must ensure that there are no processes using the devices associated with the modules that you assign to the new system before you split the system.

Assigning modules that contain open devices to the new system causes the `splitadm` command to fail. The CMS will not disable these modules and the `splitadm` command will time out waiting for them to be disabled.

Note – The `split` daemon communicates directly with the CMS to disable modules that will not be owned by the winning side when the system is being split. This means that you must ensure associated devices are quiesced in such a way that a `cmsconfig` disable action would succeed if it were attempted.

Note – Do not quiesce the boot disk manually under any circumstances if it is mirrored under Sun StorEdge Volume Manager. The system software will quiesce the boot disk safely, if necessary.

11.1.2 Splitting a System

Once the new system is prepared, as described in the preceding sections, the fault tolerant system can be split. The basic command to split the system is:

```
# splitadm split
```

When a Netra ft 1800 system is split, a high-speed communication path exists between the two sides over the PCI buses. The `split` daemon on each side can use this path to communicate with its peer on the other side. This communication is facilitated by a loadable `icn` device driver. The `icn` driver appears to the system as an Ethernet-like device and exports a standard DLPI interface. You must not modify the port numbers in the `splitd.conf` file because they correspond to the ports used by the daemon.

The following sections describe the options available for controlling the way in which the split occurs. The options can be combined on the command line as appropriate. See Section 11.1.2.4 “Example” on page 11-8 and the `splitadm`(1M) man page for details of the options for the `splitadm` command.

11.1.2.1 Specifying Timeouts

The split process attempts to disable all modules that are not owned by the winner before splitting the system. The configuration file of the `split` daemon and the options to the `splitadm` command specify how this is handled. Each disable operation is expected to terminate within a default timeout of 60 seconds or within a user-specified timeout. The timeout is determined in order of decreasing priority:

- by the `-t` option specified to the command `splitadm` (see `splitadm`(1M))
- by the `timeout` entry specified in the split daemon configuration file (see `splitd.conf`(4))

For example, to request a timeout of 30 seconds, enter:

```
# splitadm -t 30 split
```

The timeout value must be expressed in seconds.

If one of the modules cannot be disabled within the time specified, the command prints an error message similar to the following:

```
# Error: failed to disable B-CAF
```

The process then stops. At this point, all the modules that have been successfully disabled will remain disabled and the system will remain in combined (fault tolerant) mode.

You must adjust the module so that it can be disabled before you can split the system. This typically involves quiescing the devices in the module.

If the first phase completes successfully, the system is split and the new system is created.

11.1.2.2 Specifying the Winning Side

The split master side is the winning side by default. If you want the other side to be the winning side, enter the following command to split the system (assuming that the split master is side A):

```
# splitadm -w B split
```

11.1.2.3 Troubleshooting During Splitting

If an error occurs:

1. Check that you have correctly completed the preparation described in Section 11.1.1 “Before You Split the System” on page 11-3.
2. Issue the `splitadm` command again.
3. If it fails again, raise the timeout values to give the modules more time to take their devices offline.

If increasing the timeout values fails, it is likely that there is a busy device that cannot be released which prevents a relevant module from being unconfigured. You will need to quiesce the device before you can bring the module offline. You can use `cmsconfig` to identify the specific module that has failed.

11.1.2.4 Example

To split a system, requesting a timeout of 90 seconds and side B as the surviving side, enter:

```
# splitadm -t 90 -w B split
```

11.1.3 After Splitting the System

Splitting the system does not affect the surviving side, which continues to run with the same identity as before. The whole system, however, is no longer fault tolerant.

If the boot disk is mirrored under Sun StorEdge Volume Manager (that is, you are using the Volume Manager), you can reboot the new system as soon as it has completed its split actions. Two reboots are performed with one `reboot` command. During the first reboot you must provide two Sun StorEdge Volume Manager licence keys. On the second reboot, the system comes up with the new identity that was set up as described in “To Edit the Daemon and `icn` Configuration Files” on page 11-3. The CMS initialization process automatically configures the modules in the new system.

If the boot disk is not mirrored under Sun StorEdge Volume Manager (that is, you are not using the Volume Manager), you must install the new boot disk as described in the *Netra ft 1800 Installation Guide*. You are strongly recommended to mirror the boot disk to avoid the need for this. Furthermore, you must ensure that the configuration files of the new system are set up correctly as described in “To Edit the Daemon and `icn` Configuration Files” on page 11-3.

You can also use `cmsconfig` to enable the modules on the new system as required, including the network connections. Any attempt to enable modules that belong to the other side on to the new system will fail. You must disable all such modules using `cmsconfig`.

11.1.3.1 Transferring Ownership of Modules in Split Mode

You can transfer the ownership of a slot in split mode while both sides are running, or while only the master side is running. You can issue the command for transferring the ownership of a slot from either side when both are running, or from the master at any time:

```
# splitconf -l location -o [A|B]
```

For example, to transfer the ownership of A-PCI0 from A to B so that side B can use the slot, enter the following from either side:

```
# splitconf -l A-PCI0 -o B
```

Provided it is running, the side which must release the slot automatically disables the module in it. The operation fails if the module contains devices which are in use. You must ensure that the module is in a state in which its devices will be successfully taken offline before you attempt to transfer the ownership of the slot.

Using `cmsconfig`, you can enable the module on the side that now owns it.

11.1.3.2 Changing the Master in Split Mode

When `u4ftsplitd` runs on both sides, and there is communication between them, each side can initiate a transfer of slot ownership.

However, one side may need to obtain ownership of a module if the communication between the two sides is broken. For example, if side A crashes, side B may want to acquire some of its resources in order to re-establish the lost services as soon as possible.

If there is no communication between the sides, the side that is not master will refuse any attempt to transfer ownership. If the side requesting the transfer is the master, the transfer will be performed and the other side will learn about the new configuration as soon as the communication is re-established.

In the example, side B will be able to acquire some resources from A only if it is the master side.

The command for changing the master is the same as the one used in combined mode:

```
# splitconf -m side
```

The operation should nearly always succeed, even when the other side is not running. The caller is informed if the other side fails to respond.

As soon as the communication between the two systems is (re)established, they manage to (re)synchronize on a common view:

- If the two systems have a different view of the ownership of a slot, the view of the split master is accepted by the non-master side.
- If both systems believe that they are the master, the system whose `split` daemon has been running for the longer time is deemed to be the master.

If the two sides are not communicating, both sides might believe that they are the master and attempt to accept ownership of the same slots. This can result in a contradictory state. Be careful when re-assigning mastership to avoid this. If it does occur, use the command `splitconf` on the side that has the wrong information about mastership to assign mastership. For example, if side B wrongly believes that it is the master, use the following command on side B:

```
# splitconf -m A
```

11.1.4 Force Option

The `-f` option to `splitadm` forces the system to split even if one or more modules that belong to the new system fail to disable within the specified timeout. Similarly, the `-f` option to `splitconf` forces the transfer of ownership even if the module specified by the `-l` option fails to disable within the specified timeout. Using this option can cause serious loss of data if all devices in the relevant modules are not previously quiesced.

The force option should be used only as the last resort for the split or transfer of ownership in a Netra ft 1800. Use the normal procedures for the split and transfer of ownership operations whenever possible. However, in some circumstances it is not possible to disable some slots and there may, therefore, be no alternative to using the force option. You can obtain the list of devices that failed to disable, and which may, therefore, require the use of the force option, using a combination of `splitinfo` and `cmsconfig`. The force option for the `splitadm` and `splitconf` commands (and their corresponding API functions, `set_domain_attributes` and `set_slot_owner`) can be used with the following restrictions on the subsequent merge operation.

- Splitting the system with the force option

  Assume that the split operation resulted in side A winning the split and retaining the host-id of the combined system, and that the force option was used because the B-DSK device could not be disabled.

  When the two sides are subsequently merged back into a fault tolerant system, you must either make side B win the merge or disable the B-DSK (which contains all the devices affected by the force option) on side A before merging the system. If these devices are needed by the combined system, and the winner was side A, you must reboot the combined system before these devices can be enabled.

- Transferring ownership using the force option

  Assume that a module, for example, B-DSK, is to have its ownership transferred from side A to side B. In this case the subsequent merge winner must be side B. The devices in DSK-B must be disabled on side A if the subsequent merge winner is going to be side A. If these devices are needed by the combined system, and the winner was side A, you must reboot the combined system before these devices can be enabled.

Note – The restrictions described in the preceding paragraphs apply individually and in combination. You must plan very carefully to avoid creating an unusable system (which you will need to shutdown) when you split a system using the force option. In particular, never use the force option to transfer the ownership of the boot disk, and avoid leaving a side of the system without a working CAF module.

11.2 Merging a Split System

You merge a split system to return it to fault tolerant running after you have upgraded software on one side.

11.2.1 Preparing to Merge a Split System

Before you configure the system back to fault tolerant mode, you must decide which side is going to win, and with what identity. The initial identity of the combined system coincides with the identity of the winning side. If you want the winning side to assume the identity of the other side, you must change the identity of the combined system *after* you have merged the sides. Note that a change of identity always implies a reboot.

11.2.2 Merging a Split System

The merge command is:

```
# splitadm -w winner_side merge
```

The winning side keeps running the same services, while the other side is reset and its services are shut down. The CPUset module re-integration is performed automatically within the `merge` command itself.

The winning side must be specified: there is no default.

The resulting combined system has the identity of the winner.

11.2.2.1 Enabling Modules on the Surviving System

To enable the modules that previously belonged to the losing side so that they belong to the combined system, use the CMS command `cmsconfig`(1M).

11.2.2.2 Re-establishing Fault Tolerant Pairs

While the CPUset module re-integration is performed automatically, you must configure all the other fault tolerant pairs manually. See Chapter 2 “System Configuration and Administration” for details.

Note – You must use the `cmsconfig` utility for accurate information regarding the actual state of the system, fault tolerant or not, and the actual state of the system’s modules. You should not rely on the output of `split` commands for such information at this output can reflect the view of the system only from the daemon’s perspective, and it may be inaccurate. For example, it is possible that a system that is going through a re-integration process will be declared combined by the output of a `splitadm` merge command. The `split` commands and their outputs can be relied upon only for information regarding the intended state of the system (either split or configured fault tolerant), the master domain, and slot ownership.

11.3 Using Split Mode for Software Upgrades

Split mode is intended primarily to reduce down-time experienced as a result of upgrades to the application or system software. This section describes the procedure for using split mode to upgrade software and provides an example.

In this example, assume that the Netra ft 1800 system that is going through an upgrade cycle is:

- called bradfield
- running Solaris 2.6, build 6
- in the combined state (that is, configured to be fault tolerant)

Furthermore, assume that you want to upgrade the system, maintaining its identity, to Solaris 2.6, build 7. The initial state of the system software can be confirmed using the following commands:

```
# uname -a
SunOS bradfield 5.6 Build_6 sun4u sparc SUNW,Ultra-4FT
# u4ftvmctl -c
CPUsets running combined
```

The initial configuration file for the `split` daemon on the combined system is set up as follows:

```
# pg /etc/splitd.conf
# The domain-address section.
# Add the primary and alternative hostnames
# (used for inter-daemon comms) of the two sides here.
# Example:
# host_prim host-i0 host-2-i0
# host_alt thishost thishost-2
host_prim bradfield-i0 bradfield-2-i0

# The port-number section
port_address a b 3501
port_address b a 3500
port_address a b 3502
port_address b a 3503

# The misc-section
timeout  60
dolog    yes

# Hostnames section
# add the hostnames of the two sides here. Example:
# hostnames thishost thishost-2
hostnames bradfield bradfield-2
```

Given that the primary mechanism for the inter-daemon communication is selected to be `icn`, you must remember to modify the daemon and icn configuration files before splitting the system. Furthermore, if the system is not using the Sun StorEdge Volume Manager, these files must be set up correctly on the side that is going to lose the split for the correct operation of the system. The split operation will create the correct configuration files if the Volume Manager is used for disk mirroring.

▼ Upgrade Procedure

1. **Prepare the system to split according to the instructions provided in Section 11.1.1 “Before You Split the System”.**

   For the purpose of the example, assume that no preparatory work is needed, other than modifying the configuration files, as the system is going to be split along the natural A and B sides. The corrected daemon configuration file (for both sides) is given above. The corrected (for side A) `icn` configuration file is given below:

```
#!/sbin/sh
#
#ident  "@(#)config.icn0        1.3     98/11/03 SMI"
#
# icn config script
#
PARENT_PATH=/pci@6,2000/u4ftbus@0/u4ioslot@6
DEV_PROPS="reg=0x82803000,0x00000000,0x58000000,0x00000000,0x040
00000"
HOSTNAME=bradfield-i0
```

For side B (which will lose the split operation), the correct icn configuration file is:

```
#!/sbin/sh
#
#ident  "@(#)config.icn0        1.3     98/11/03 SMI"
#
# icn config script
#
PARENT_PATH=/pci@6,2000/u4ftbus@0/u4ioslot@6
DEV_PROPS="reg=0x82803000,0x00000000,0x58000000,0x00000000,0x040
00000"
HOSTNAME=bradfield-2-i0
```

The only difference is in the HOSTNAME entries. Furthermore, if theVolume Manager is used for disk mirroring, there would be no need to create (or correct) the above file.

2. **Split the system using the `splitadm` command as instructed in Section 11.1.2 "Splitting a System".**

   The winner of the split operation will continue to run, but the loser will drop to the PROM. This can be done using the following command:

```
# splitadm -w a split
icn network setup: u4ftmbox icn0 icn.keepalive done.
WARNING: Out-of-sync on CPUset B
domain = A
attributes = split,master
master = A
#
```

The console output above is for side A, that is, the winner of the split operation.

On side B, the console output will show that the system is going through a reset and will be ready for a boot. Some of that output is shown below:

```
<*>

Running C prom.
Prom base address:      0000.01ff.f020.0000
This is cpuset  B

Cpuset fpga download version: 2.4
Button Power ON

CPU0 has assumed the role of Boot CPU

@(#) Sun Ultra 4FT UPA/PCI 3.7 [PROTO-P1b-sd_st: Fusion-B2]
Version 14.0 created 1998/12/21 17:13
.
.
ebus-test ebus-test Sun Ultra 4FT UPA/PCI(2 X UltraSPARC-II
296MHz), No keyboard
OpenBoot 3.7 [PROTO-P1b-sd_st: Fusion-B2], 512 MB memory
installed, Serial #10824173.
Ethernet address 8:0:20:a5:29:ed, Host ID: 80a529ed.

Initializing  505 megs of memory at addr 0; left:  505MB
{0} ok
```

3. **Boot the loser of the split operation following the instructions provided in Section 11.1.3 “After Splitting the System”.**

4. **Install the new version of the software on the loser of the split as instructed in the appropriate installation manual.**

For the purpose of the example, the third and fourth steps of the upgrade procedure are combined. At the boot prompt, issue the following command:

```
{0} ok boot b-cdrom
```

This will cause the boot device to be `b-cdrom0` and the system to generate the output for the installation. You should now install the new software and then boot the system from the newly installed boot device. Some of this interaction and the generated output is shown below:

```
Boot device: b-cdrom  File and args: kadb
Loading ufs-file-system package 1.4 04 Aug 1995 13:02:54.
FCode UFS Reader 1.10 96/10/15 00:57:29.
.
.
Installation complete
Executing SolStart postinstall phase...
Executing finish script "patch_finish"...

Finish script patch_finish execution completed.
Executing JumpStart postinstall phase...

The begin script log 'begin.log'
is located in /var/sadm/system/logs after reboot.

The finish script log 'finish.log'
is located in /var/sadm/system/logs after reboot.

# eeprom boot-device=b-dsk4   diag-device=b-dsk4
# reboot
Jan 12 04:45:34 rpcbind: rpcbind terminating on signal.
syncing file systems... done
NOTICE: resetting <ttymux#0>, cmd 0
NOTICE: resetting <u4ioslot#0>, cmd 0
NOTICE: resetting <u4ioslot#1>, cmd 0
.
.
NOTICE: resetting <u4ioslot#30>, cmd 0
NOTICE: resetting <u4ioslot#31>, cmd 0
rebooting...
Resetting ...

<*>

Running C prom.
Prom base address:      0000.01ff.f020.0000
This is cpuset  B

Cpuset fpga download version: 2.4
Software Power ON

CPU0 has assumed the role of Boot CPU

@(#) Sun Ultra 4FT UPA/PCI 3.7 [PROTO-P1b-sd_st: Fusion-B2]
Version 14.0 created 1998/12/21 17:13
```

```
.
.
Rebooting with command: boot
Boot device: b-dsk4  File and args: kadb
Loading ufs-file-system package 1.4 04 Aug 1995 13:02:54.
Loading: /platform/SUNW,Ultra-4FT/ufsboot
Loading: /platform/sun4u/ufsboot
kadb: kernel/unix
Size: 278101+61104+70432 Bytes
/platform/SUNW,Ultra-4FT/kernel/unix loaded - 0x98000 bytes used
SunOS Release 5.6 Version Build_7 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
NOTICE: Ultra-4FT DDI extensions installed
linking network controllers: hme0 hme1.
configuring network interfaces: pnet0.
Hostname: bradfield-2
Configuring the /devices directory
Configuring the /dev directory
Configuring the /dev directory (compatibility devices)
configuring icn network interfaces
Waiting for CMS objects to online ...
The system is coming up.  Please wait.
checking ufs filesystems
/dev/rdsk/c1t4d0s5: is clean.
Configuring network interface addresses: hme2 pnet0 pnet1.
NIS domainname is slough.uk.sun.com
starting routing daemon.
starting rpc services: rpcbind keyserv ypbind done.
On this screen you can create a root password.
.
.
setting netmask of pnet0 to 255.255.255.0
Setting default interface for multicast: add net 224.0.0.0:
gateway bradfield-2
syslog service starting.
Print services started.
volume management starting.
The system is ready.

bradfield-2 console login:
```

In the above example, the second system has been upgraded to build 7 release and is ready for use. You can login and examine the status of the system using the `cmsconfig` command. You can also get information about the split status of the system using the `splitinfo` command:

```
# splitinfo -dam
domain = B
attributes = split
master = A
```

5. **Prepare the system for the merge operation as instructed in Section 11.2.1 "Preparing to Merge a Split System".**

6. **Merge the split system using the `splitadm` command as instructed in Section 11.2.2 "Merging a Split System".**

   Note that for the upgrade to take effect, the merge winner must be the side that has gone through the software upgrade (that is, the split loser). For the purpose of the example, consider that the preparation is done according to Section 11.2.1 "Preparing to Merge a Split System". You can now issue the following command (on either side) to merge the system:

```
# splitadm -w b merge
```

   Side A will go through a CPU reset and will wait for the PRI. Side B, meanwhile, will continue to provide service and will re-integrate the A CPUset. Some of the output observed on console A is given below:

```
# icn network shutdown: icn.keepalive icn0 icn1 icn2 icn3 u4ftmbox
done.
# INIT: New run level: 6
The system is coming down.  Please wait.
System services are now being stopped.
Print services stopped.
Stopping the syslog service.
syslogd: going down on signal 15
Jan 12 13:02:07 snmpdx: received signal 15
The system is down.

syncing file systems... done
NOTICE: resetting <u4ft-eeprom#0>, cmd 0
NOTICE: resetting <ttymux#0>, cmd 0
.
.
```

```
NOTICE: resetting <u4ioslot#31>, cmd 0
NOTICE: resetting <hme#5>, cmd 0
rebooting...
Resetting ...
```

Side B displays only the following output:

```
# icn network shutdown: icn.keepalive icn0 icn1 icn2 icn3 u4ftmbox
done.
# splitinfo -dam
domain = C
attributes = combined
master = A
```

To confirm that the system is in the combined (that is, fault tolerant) state, you can enter the following command:

```
# u4ftvmctl -c
CPUsets running combined
```

Note – This system is NOT the initial system, bradfield, but is a new system called bradfield-2:

```
# uname -n
bradfield-2
```

Note – You can enable the resources of the side that lost the merge using the CMS but for the purpose of the example, this step is not included.

---

7. **Ensure that the resources (including the data and the boot disk) are mirrored so that you are back to a fully fault tolerant mode of operation.**

   For the purpose of the example, it is assumed that this step is performed.

8. **Change the identity of the system.**

   The new fault tolerant system will have the identity of the side that lost the initial split operation; therefore, its identity differs from that of the original fault tolerant system. You can change the identity of the system and reboot to have the same identity as the original fault tolerant system. This can be done by updating the following files as shown below:

   - /etc/nodename
   - /etc/hostname.*
   - /etc/inet/hosts
   - /etc/net/ticots/hosts
   - /etc/net/ticotsord/hosts

```
# uname -n
bradfield-2
#
# pg /etc/nodename
bradfield
# pg /etc/hostname.*
bradfield
# pg /etc/inet/hosts
# Internet host table
#
127.0.0.1      localhost
129.156.140.64  bradfield-2
129.156.140.62  bradfield     loghost
# pg /etc/net/ticlts/hosts
# RPC Hosts
bradfield     bradfield
# pg /etc/net/ticots/hosts
# RPC Hosts
bradfield     bradfield
# pg /etc/net/ticotsord/hosts
# RPC Hosts
bradfield     bradfield
```

Now you can halt the system using the `halt` command. The following output will be seen on both consoles (assuming that both CAFs have been enabled):

```
# halt
Jan 12 13:21:29 bradfield-2 halt: halted by root
syslogd: going down on signal 15
Jan 12 13:21:34 bradfield-2 snmpdx: received signal 15
Jan 12 13:21:34 bradfield-2 rpcbind: rpcbind terminating on
signal.
syncing file systems... done
NOTICE: resetting <u4ft-eeprom#0>, cmd 0
NOTICE: resetting <ttymux#0>, cmd 0
.
```

```
.
.
NOTICE: resetting <hme#4>, cmd 0
NOTICE: resetting <se#1>, cmd 0
Program terminated
{0} ok
```

You can now reboot the system and bring the new system up:

```
{0} ok **boot**
Boot device: b-dsk4  File and args: kadb
Loading ufs-file-system package 1.4 04 Aug 1995 13:02:54.
FCode UFS Reader 1.10 96/10/1500:57:29.
Loading: /platform/SUNW,Ultra-4FT/ufsboot
Loading: /platform/sun4u/ufsboot
kadb: kernel/unix
Size: 278101+61104+70432 Bytes
/platform/SUNW,Ultra-4FT/kernel/unix loaded - 0x98000 bytes used
SunOS Release 5.5 Version Build_7 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
NOTICE: Ultra-4FT DDI extensions installed
configuring network interfaces: pnet0.
Hostname: bradfield
Waiting for CMS objects to online ...
The system is coming up.  Please wait.
checking ufs filesystems
.
.
volume management starting.
The system is ready.
```

You can log on to the system and see the system configuration and split status:

```
bradfield console login: **root**
Password:
Jan 12 13:34:37 bradfield login: ROOT LOGIN /dev/console
Last login: Tue Jan 12 12:55:37 on console
Sun Microsystems Inc.   SunOS 5.6       Generic August 1997
# **hostid**
80a529ed
# **hostname**
bradfield
# **u4ftvmctl -c**
CPUsets running combined
#
```

11.4 Split Mode Reference Information

11.4.1 Configuring the `icn` Driver for Split Mode Operation

The Inter-CPUset network (`icn`) driver is a multi-threaded, loadable STREAMS driver supporting the connectionless Data Link Provider Interface, `dlpi`(7P), over the Netra ft 1800 PCI bridges. It enables network communication between the two sides of a split Netra ft 1800 system without external cabling. The driver emulates an Ethernet style device, although there are only ever a maximum of two devices connected.

The Netra ft 1800 hardware supports four distinct hardware paths between the two sides of a split system, and the `icn` driver supports separate instances corresponding to each of these four paths. These paths are accessed via the character special device `/dev/icn`.

`icn` instances are paired; that is, `icn0` on side A communicates directly with `icn0` on side B and so on. Each `icn` instance therefore requires its own subnet with the matching instances on each side being the *only* addresses in that subnet. Furthermore, the interfaces should be configured as private with `ifconfig`(1M) to prevent the addresses being published to the outside world.

11.4.1.1 Configuration

Each `icn` instance requires a hostname and an IP address. The IP addresses allocated to the `icn` instances should be taken from IETF RFC 1918, which designates the net address 192.168.0.0 as being available for private networks. The set of recommended IP addresses is shown in TABLE 11-1.

TABLE 11-1 IP Addresses for `icn` Instances

Side A	Instance	Side B
192.168.1.1	icn0	192.168.1.2
192.168.2.1	icn1	192.168.2.2
192.168.3.1	icn2	192.168.3.2
192.168.4.1	icn3	192.168.4.2

Similarly, each active instance requires a matching hostname entry (in `/etc/hosts` or NIS) exactly like other network interfaces. For ease of identification, these should have a consistent form. That is, if side A has hostname hA, and side B has hostname hB then the connection through `icn0` has the names `hA-i0` (on side A), and `hB-i0` (on side B), and so on for `icn`.

The `icn` configuration files can be found in `/etc` and are named `config.icn{0-3}`. Each file contains several entries, one per line. Only the line that starts `HOSTNAME=` should be modified to indicate the hostname this instance should use. An example file is given below.

```
#!/sbin/sh
#
#ident"@(#)config.icn01.398/11/03 SMI"
#
# icn config script
#
PARENT_PATH=/pci@6,2000/u4ftbus@0/u4ioslot@6
DEV_PROPS="reg=0x82803000,0x00000000,0x58000000,0x00000000,0x040
00000"
HOSTNAME=host-i0
```

Note – A configuration file that does not specify a `HOSTNAME` is ignored by the start up and shutdown scripts.

11.4.1.2 Starting Up and Shutting Down `icn`

`icn` should be active only when the system is split. To this end it is normally controlled via the split daemon `u4ftsplitd`. When the system is split (or rebooted in the split state), the daemon executes the *`icn.init`* script (in `/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib`). This script ensures all the necessary drivers are loaded and automatically configures the individual `icn` instances using information in the configuration files.

If the start up succeeds, all the configured `icn` connections will be up and available. No further user actions should be necessary.

An additional script, *`icn.keepalive`* is started automatically by the *`icn.initscript`*. Its function is to re-establish connectivity if the opposite side is rebooted.

Similarly, when the system is merged, the daemon shuts down all the `icn` connections (if possible) via the *`icn.terminate`* script (also in `/usr/platform/SUNW,Ultra-4FT/SUNWcms/lib`).

11.4.1.3 Troubleshooting `icn`

The standard networking utilities (`ifconfig`(1M), `ping` (1M), `netstat`(1M) and `ndd`(1M)) can all be used to interrogate the state of `icn` connections. If you suspect connectivity problems, check the following areas.

- Ensure the interface appears to be known to the system (using `ifconfig`(1M)) and is marked as up with correct IP addresses and other settings. If it is not, check the appropriate configuration file.
- If the configuration files are correct, check the network state with `netstat`(1M).

  Example `netstat` output is:

```
# netstat -I icn0
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
icn0  40945host-i0       host-i0         3       0     1      0     0      0
```

  If the error count `Oerr` is increasing, this typically indicates that connection to the other side has not been established. Examine the status log for any messages from the `icn` driver that indicate a failure to establish the connection.

  Connection failure can occur if the other side is down or the `icn` system is not active on it (*`icn.init`* not run, bad configuration files, or `icn` terminated using *`icn.terminate`*).

  After remedying the cause, connectivity should be established. If it is not, re-initialize the `icn` system using *`icn.terminate`* and *`icn.init`*. If all else fails try rebooting.

- Simple status checks can be performed using `u4ftctl` as follows:

```
 # /usr/platform/SUNW,Ultra-4ft/SUNWcms/lib/u4ftctl -d /dev/icn
status
   (icn#0): Online
   (icn#1): Not initialized
   (icn#2): Not initialized
   (icn#3): Not initialized
```

This indicates that the icn driver is online on instance 0 but no connections have been initiated or established yet. All other instances are not initialized (the value for HOSTNAME in the appropriate configuration file is empty).

```
 # /usr/platform/SUNW,Ultra-4ft/SUNWcms/lib/u4ftctl -d /dev/icn
status
   (icn#0): Online + Exporting + Importing
   (icn#1): Not initialized
   (icn#2): Not initialized
   (icn#3): Not initialized
```

This indicates that two way communication across instance 0 is possible.

- More detailed information retrieval is possible using ndd(1M). Note that this information is very detailed and probably only of use to customer support. A highly abbreviated extract is shown below.

```
 # ndd /dev/icn icn_status_report
   Global
        drv_state:              ICN_ENABLED
        drv_opens:              3
        drv_ninterfaces:        1

   Instance 0
        error count:    1
        icn_smr_size:   524288 bytes
        icn_msgd_lim:   255 entries
        Local Memory
                smr_state:      ICN_SMR_MEM_ONLINE
                smr_enabled:    0
                smr_suspended:  0
        Remote Memory
                smr_state:      ICN_SMR_MEM_ONLINE
                smr_enabled:    0
                smr_suspended:  0
```

11.4.2 `u4ftsplitd` logging

By default, the split daemon logs relevant events and error messages in the status log file, `u4ftlog.status`.

It is possible to disable the logging by editing the line `dolog` from the `u4ftsplitd.conf` configuration file.

The following messages are printed to the log file:

- report_message (notification of split mode-related events and failures)
  - `SPLITD_LOG_LEVEL_HIGH`

    Messages at this level notify the user of severe errors that cause the termination of the split daemon.

```
"nvram read failed"
"nvram type 1 of mode disagreement"
"set eeprom failed"
"nvram type 2 of mode disagreement"
"killed"
```

  - `SPLITD_LOG_LEVEL_MID`

    Messages at this level notify the user of severe errors that prevent the split daemon from completing a requested operation.

```
"nvram write failed"
"set eeprom A failed"
"set eeprom B failed"
"nvram write failed for side A"
"nvram write failed for side B"
"failed to open eintr dev"
"eintr ioctl failed to split"
"failed to open eintr dev"
"ioctl failed on eintr dev"
```

  - `SPLITD_LOG_LEVEL_LOW`

    Messages at this level notify the user of non-severe errors reported by the split daemon.

```
"Unable to connect to the console."
```

- SPLITD_LOG_LEVEL_NONE

  Messages at this level notify the user of events. They should not be confused with error messages.

```
"nvram split_master is A"
"nvram split_master is B"
“Master changed from B to A"
"Master changed from A to B"
```

- report_trace (internal daemon events of failures)

  Trace messages are generated for debugging purposes when a very serious condition occurs in the split daemon.

```
"SET_SPLIT_MASTER timedout when not SPLIT"
"SET_SPLIT_MODE timed out when not STOC"
"send_report: side is not A|B"
"pthread_attr_init failed for net thread"
"Failed to create net thread"
"Inconsistent slot offline state"
"SPLIT_REPORT ignored when not in SPLIT mode"
"SPLIT_REPORT ignored - other side not in SPLIT mode"
"SPLIT_REPORT error: two same sides"
"NEW_MASTER_REQ not expected when not in SPLIT mode"
"NEW_MASTER_RPLY not expected when not in SPLIT mode"
"Request for NEW_MASTER_RPLY not found"
"NEW_MODE_REQ not expected when not in SPLIT mode"
"NEW_MODE_RPLY not expected when not SPLIT or STOC"
"Request for NEW_MODE_RPLY(err) not found"
"NEW_OWNER_REQ not expected when not in SPLIT mode"
"Inconsistent slot offline state"
"NEW_OWNER_RPLY not expected when not in SPLIT mode"
"splitd, received unknown message"
"Split resulted in wrong winner"
"SPLIT_REPORT winner, difs: 0x%x, 0x%x, 0x%x, 0x%x, 0x%x"
"SPLIT_REPORT looser, difs: 0x%x, 0x%x, 0x%x, 0x%x, 0x%x"
"Request for NEW_OWNER_RPLY(%d,%d,%d) not found"
"Corrupt message from net (%d,0x%x)"
```

11.4.3 Using the `split` Library

Instead of the command line, you can use the `split` library to access the split functionality. Refer to the following manual pages for the details of the split API:

- `get_domain_attributes`(3)
- `set_domain_attribute`(3)
- `get_slot_status`(3)
- `set_slot_owner`(3)
- `split_lock`(3)
- `split_unlock`(3)

You must include the file `/usr/platform/SUNW,Ultra4-FT/SUNWcms/include/split_api.h` in your source files, and link to the split library by including lines similar to the following in your makefile:

```
LIBSPLIT         + /usr/platform/SUNW,Ultra4-FT/SUNWcms/lib/
LDLIBS           += -R$(LIBSPLIT) -L$(LIBSPLIT) -lu4ftsplitmt
```

Note – There are two versions of the split library available; a simplified non-thread-safe version (`libu4ftsplit`), and the full thread-safe version (`libu4ftsplitmt`). If you use the simplified version, only the API functions `get_domain_attributes`(3) and `get_slot_status`(3) are available. If you want to use the full API or any of the `set_` routines, you must use the multi-thread-safe version of the library (`libu4ftsplitmt`) *and* compile your application in a thread-safe manner (that is, specify at least `-D_REENTRANT` in the flags to your compiler). In addition, you should follow any other mt-specific actions your compiler requires (see the appropriate compiler documentation for specific details).

CHAPTER **12**

Replacing Modules

This chapter describes how to remove and insert:

- CPUset, PCI, CAF and PSU modules (Section 12.2 “Replacing a Faulty Module” on page 12-3)
  - CPUset modules: “Changing a CPUset Module” on page 12-4
  - PCI modules: “Changing a PCI Module” on page 12-8
  - CAF modules: “Changing a CAF Module” on page 12-7
  - PSU modules: “Changing a PSU Module” on page 12-9
- Drive chassis (Section 12.4 “Replacing a Disk Chassis” on page 12-13)
- Removable media module (Section 12.3 “Replacing an RMM” on page 12-10)
- Motherboards (Section 12.5 “Replacing a Motherboard” on page 12-18)
- Environmental filters (Section 12.6 “Changing the Environmental Filters” on page 12-29).

Caution – The wrist-strap provided must be used when replacing modules, or making cable connections to the rear of the system. The wrist-strap connection point on the Netra ft 1800 system is located on the panel at the bottom rear of the chassis.

Caution – Ensure the system is connected reliably to earth.

12.1 Overview

All modules have their own guides in slots in the chassis, into which they fit exactly. No module will fit into a slot allocated to a different class of module. No module will fit into its own slot if it is upside down.

See the chapters on the individual subsystems for details of hardware compatibility.

12.1.1 Module Indicator LEDs

Most modules have at least the following two indicator LEDs: a green *Power* LED and a red *Fault* LED.

The indicator combinations are shown in TABLE 12-1.

TABLE 12-1 Module LEDs

Power (Green)	Fault (Red)	Description
Off	Off	No power to module, disabled
On	Off	Power on, normal operation
Off	On	Module faulty, needs changing
On	On	Module faulty, but still enabled; must be disabled before it can be removed

12.1.2 Physical Protection of Modules

Each module is encased in metal so that users cannot easily damage the mechanical or electrical components.

Caution – Dangerous voltages, capable of causing death or injury, are present in this equipment.

12.1.3 Module Injector/Ejector Mechanisms

All the modules except the disk chassis (DSK) and RMM have an injector/ejector lever (CPUset modules have two). They are all similar in function and usage. A common feature is a slide which engages and disengages the module's electrical connection to the motherboard, and a lever which physically engages and disengages the module. When the latch is disengaged, a red dot is exposed. This facilitates the identification of unlatched injectors.

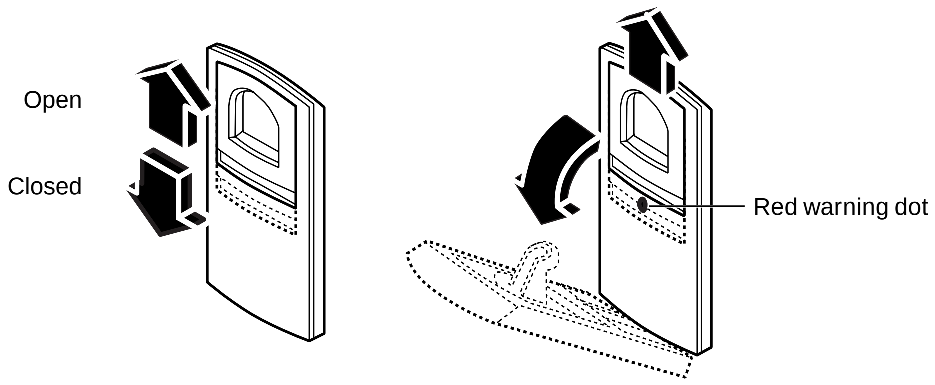


FIGURE 12-1 Module Injector/ejector Lever

The module is disengaged from its electrical connection when the slide is moved towards the rounded end of the lever, exposing the red warning dot.

12.2 Replacing a Faulty Module

In all the following procedures you can use `xcmsfix` instead of `cmsfix`.

Caution – If you are removing or installing a PSU module, open the external power breaker associated with that PSU before physical removal or insertion.

The general procedure refers to CPUset, CAF, PCI and PSU modules. For more specific instructions for the RMM, disk chassis and motherboards, refer to:

- Section 12.3 "Replacing an RMM" on page 12-10
- Section 12.4 "Replacing a Disk Chassis" on page 12-13
- Section 12.5 "Replacing a Motherboard" on page 12-18

▼ To Remove a Faulty Module

1. **Disable the module:**

   a. **Start `cmsfix`.**

```
# cmsfix
```

   The faulty module appears in the list.

   b. **Disable the module.**

   Use the arrow keys to select the module in the list, then type **`D`**.

   When the state of the module changes to `disabled`, the module can be removed. If the module is not disabled successfully, identify the cause (for example, an application service that is still using the module), resolve the problem, and repeat the command to disable the module.

2. **Move the slide in the lever on the module to the disengaged position.**

   This will expose the red warning dot.

Caution – If you are replacing a PSU module, open the external power breaker associated with that PSU before removing the module.

---

3. **Lower the lever.**

   The module will slide out a small amount when the lever is fully lowered.

4. **Slide the module out of its slot, using the handle if there is one.**

   For specific procedures relating to individual modules, refer to:

- "Changing a CPUset Module" on page 12-4
- "Changing a CAF Module" on page 12-7
- "Changing a PCI Module" on page 12-8
- "Changing a PSU Module" on page 12-9

Changing a CPUset Module

CPUset modules have two injector levers which must be operated simultaneously.

As you pull out the CPUset module, the handle in the top panel pops up and must be depressed again manually in order to withdraw the module fully from the chassis (see FIGURE 12-2). Once the handle is clear of the crossbar and has popped up again, it can be used to take the weight of the module.

Caution – CPUset modules are very heavy. The weight warning label on the CPUset is for guidance only. The actual weight of a CPUset depends on its configuration. Both the front and top handles must be used simultaneously once the module has been withdrawn as illustrated in FIGURE 12-2.

On inserting the CPUset module the handle must be depressed in order to push the module fully into the chassis.

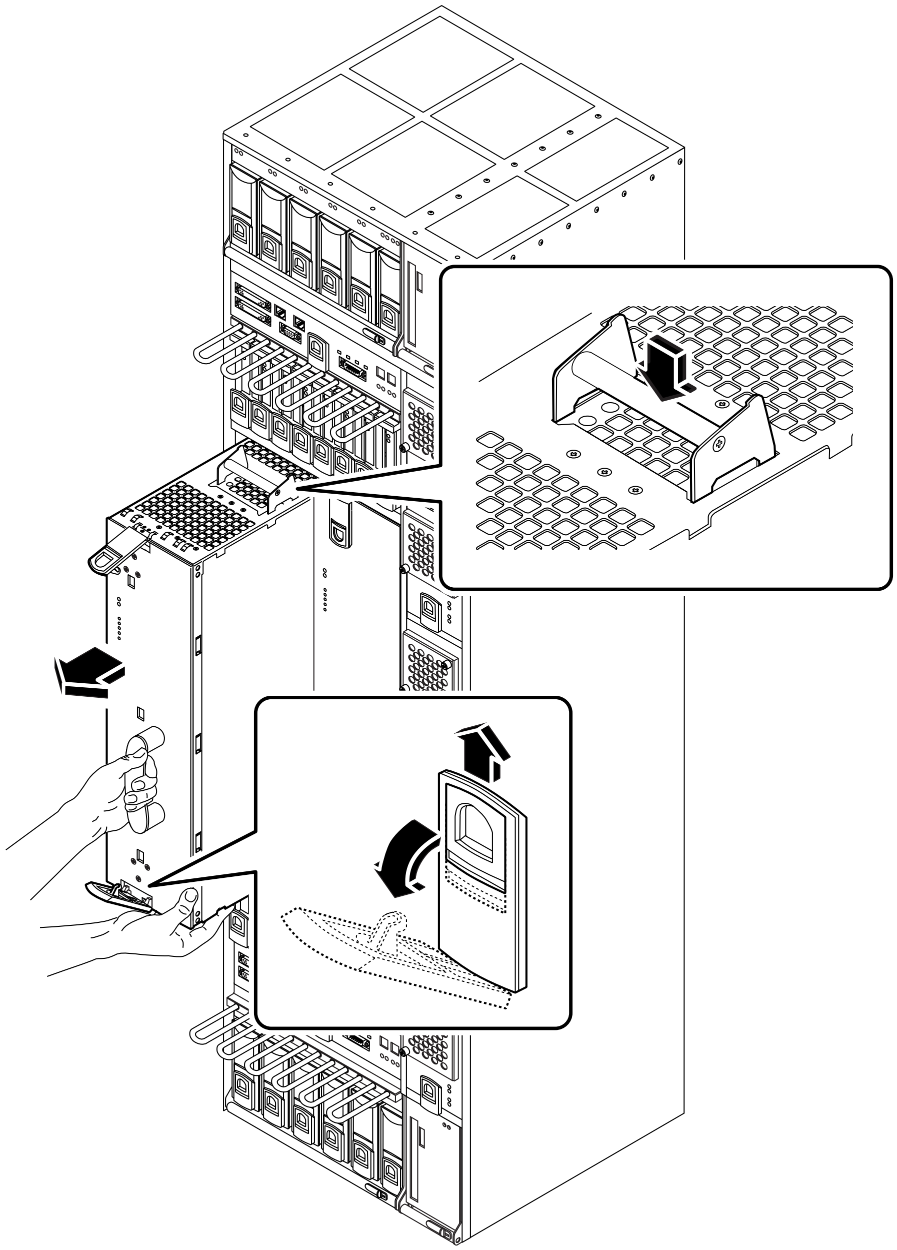


FIGURE 12-2 Removing a CPUset Module

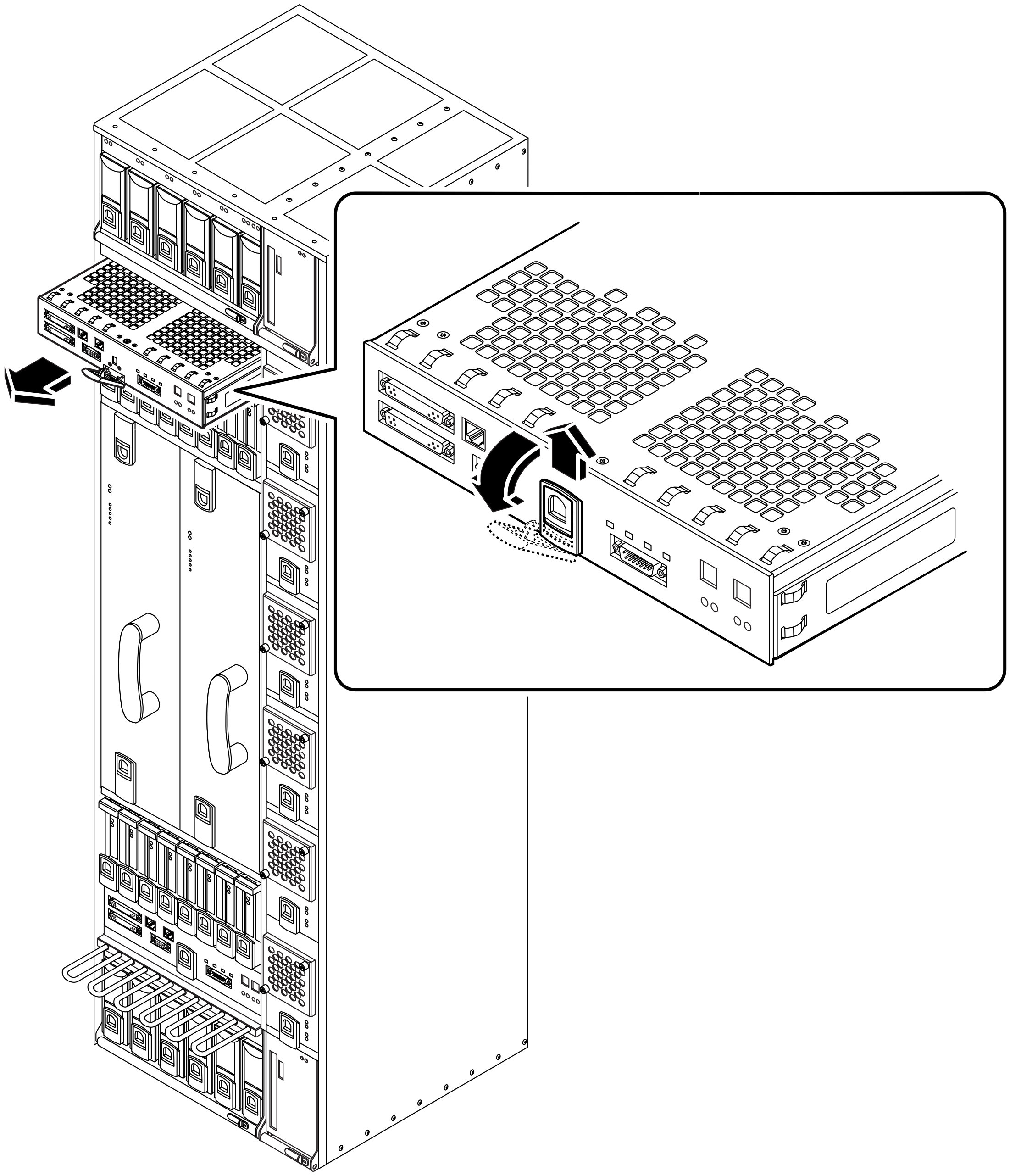


FIGURE 12-3 Removing a CAF

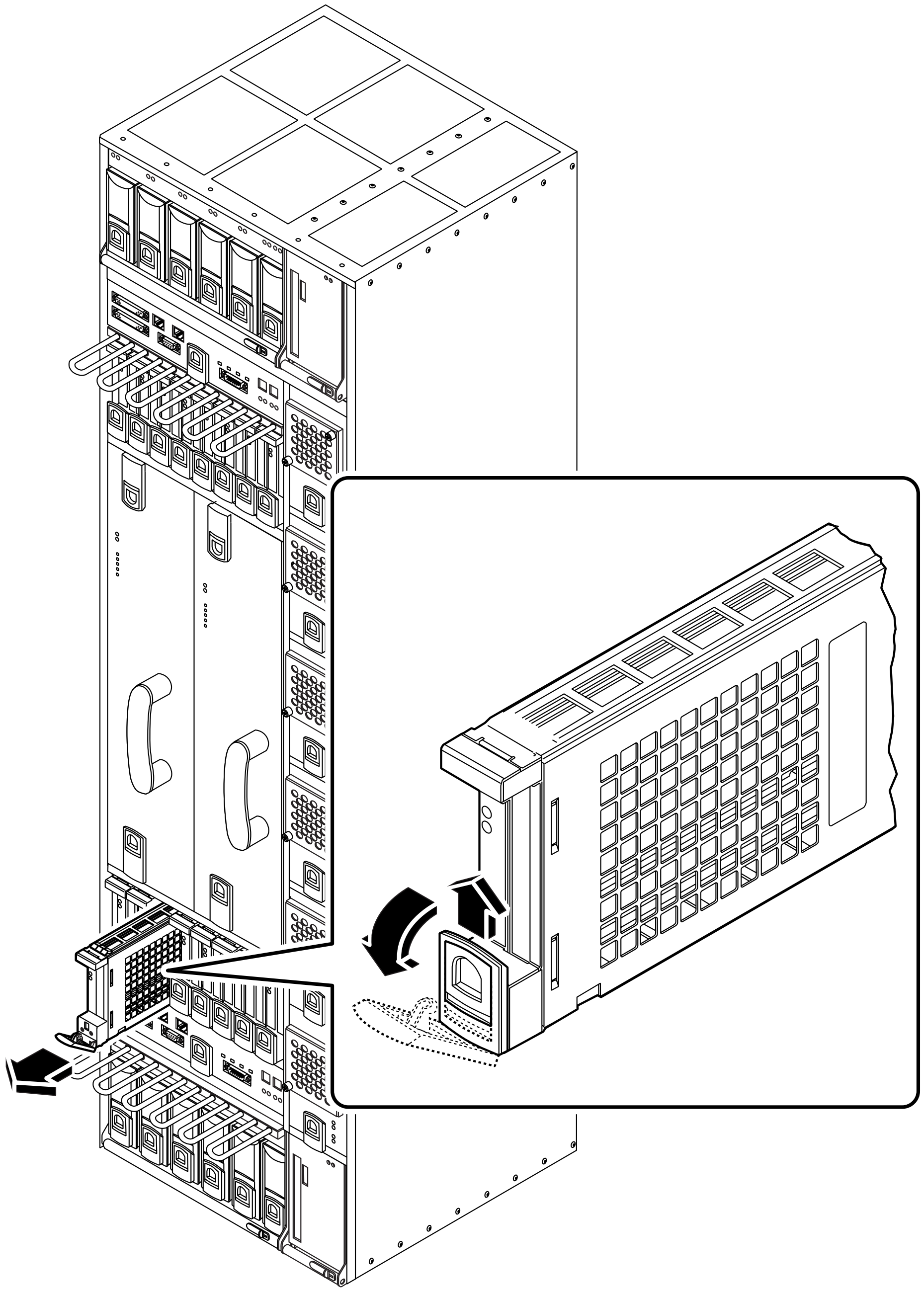


FIGURE 12-4 Removing a PCI Card Carrier

Changing a PSU Module

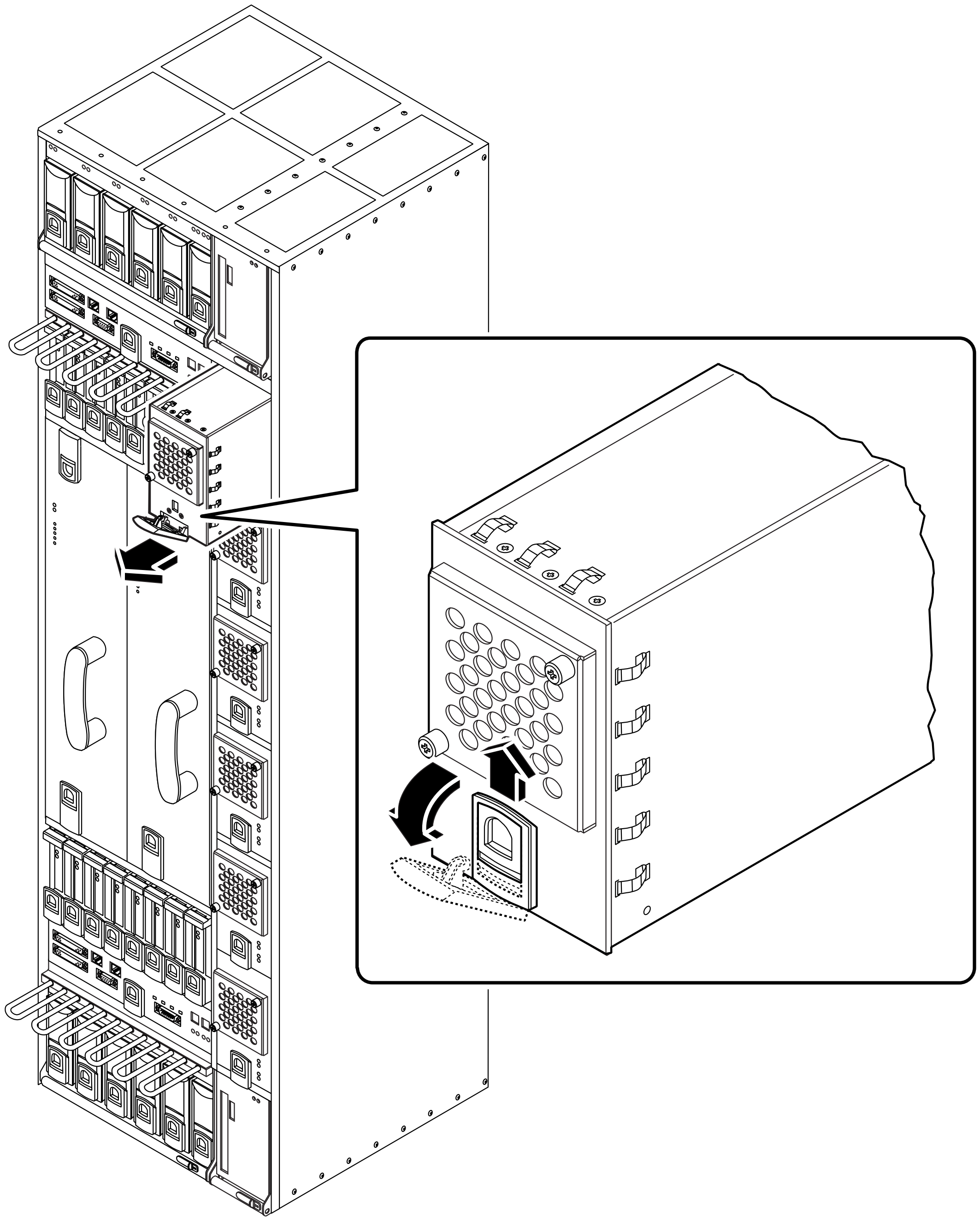


FIGURE 12-5 Removing a Power Supply

▼ To Replace a Module

1. **Slide the module into its slot but not fully home.**

   A module will not fit into a slot designed for a different class of module.

2. **When the lever engages with the chassis, raise it to push the module fully home.**

3. **Move the slide in the lever into the engaged position.**

4. **Configure the module into the system.**

   a. **Start** `cmsfix`**.**

   ```
   # cmsfix
   ```

   A list of modules is displayed, including the new module.

   b. **Enable the module.**

   Use the arrow keys to select the module in the list and type **E**.

   The state of the module changes to `enabled`. The module is now restored.

12.3 Replacing an RMM

RMM modules have a slide with an actuator microswitch on an ejector handle. The slide controls the electrical connection to the motherboard. When the slide is closed (to the right), the electrical connection is engaged; when it is open (to the left), the electrical connection is disengaged. The handle is lifted to disengage the module physically, and lowered to engage it.

▼ To Remove the RMM

1. **Disable the RMM.**

   a. **Start** `cmsfix`:

   ```
   # cmsfix
   ```

   The faulty RMM appears in the list.

b. **Disable the RMM.**

   Use the arrow keys to select the module in the list, then type **`D`**.

   When the state of the module changes to `disabled`, the module can be removed. If the module is not disabled successfully, identify the cause (for example, an application service that is still using the module), resolve the problem, and repeat the command to disable the module.

2. **Slide the latch in the handle of the RMM to the left (towards the 'unlocked' symbol) to disengage the power supply.**

3. **Lift the handle.**

4. **Slide the RMM out of its slot.**

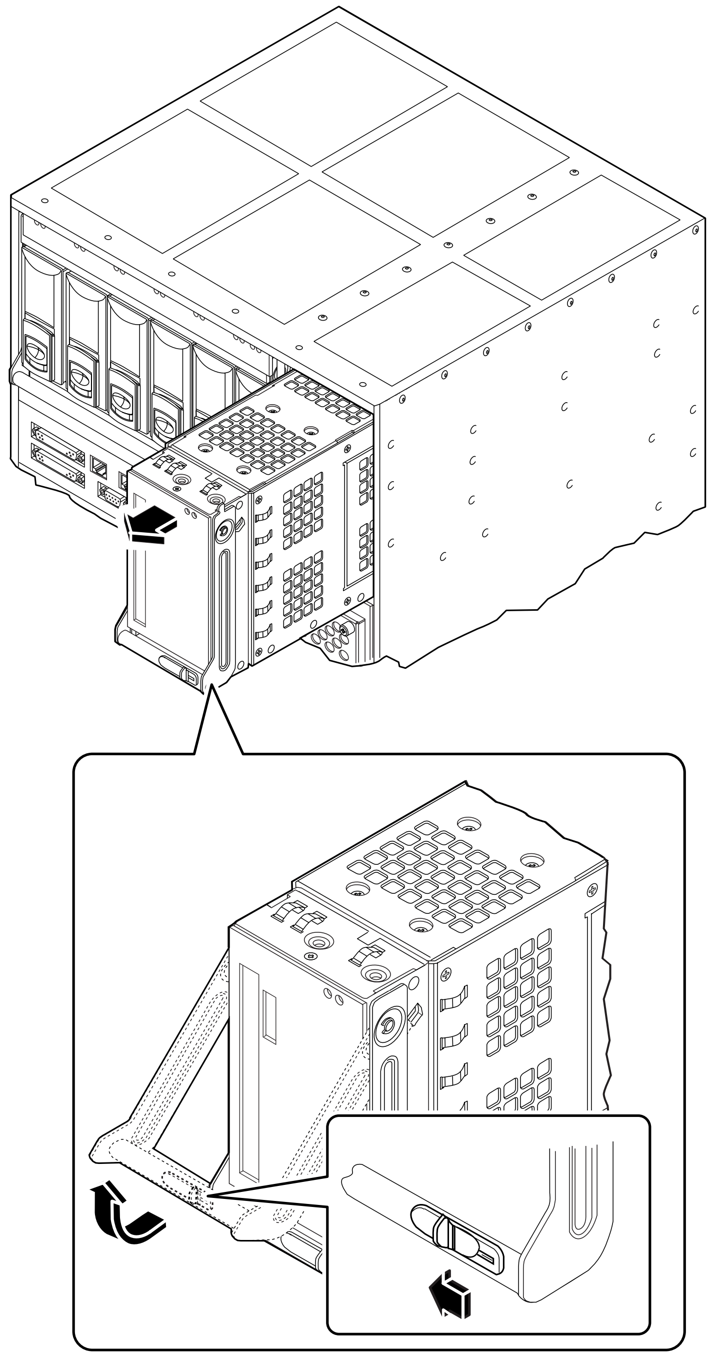


FIGURE 12-6 Removing an RMM Module

▼ To Replace the RMM

1. **Slide the RMM into its slot until it is almost completely home.**

   A module will not fit into a slot designed for a different class of module.

2. **Lower the handle to engage the RMM fully in its slot.**

3. **Slide the latch in the handle to the right (towards the 'locked' symbol) to engage the power supply.**

4. **Enable the RMM.**

   a. **Start** `cmsfix`**.**

   ```
   # cmsfix
   ```

   A list of modules is displayed, including the new RMM.

   b. **Enable the RMM.**

   Use the arrow keys to select the RMM in the list and type `E`.

   The state of the RMM changes to `enabled`. The RMM is now restored.

12.4 Replacing a Disk Chassis

DSK modules have a slide on an ejector handle. The handle is lifted to disengage the module physically, and lowered to engage it.

Note – The slide on the hard disk drive module injection lever does not control the electrical connection. If the disk chassis is powered on, the module will power on as soon as it is inserted, regardless of the position of the slide.

⚠ **Caution –** Always remove all hard disk drive modules before removing a disk chassis. To avoid the risk of data corruption or physical damage, allow at least 30 seconds after powering down an HDD module (hard disk) before physically removing it. Always put a hard disk back in the same location from which you removed it.

▼ To Remove the Disk Chassis

1. **Remove any HDD modules (hard disks) in the disk chassis.**

   If the disk chassis has failed, the modules are already disabled. Remove them as shown in FIGURE 12-7 on page 12-15.

⚠ **Caution –** You must completely remove all the HDD modules before unplugging the disk chassis. Make a note of the location of each HDD module as they must be re-inserted in the same locations.

---

2. **Disable the disk chassis.**

   a. **Start** `cmsfix`:

   ```
   # cmsfix
   ```

   The faulty disk chassis appears in the list.

   b. **Disable the disk chassis.**

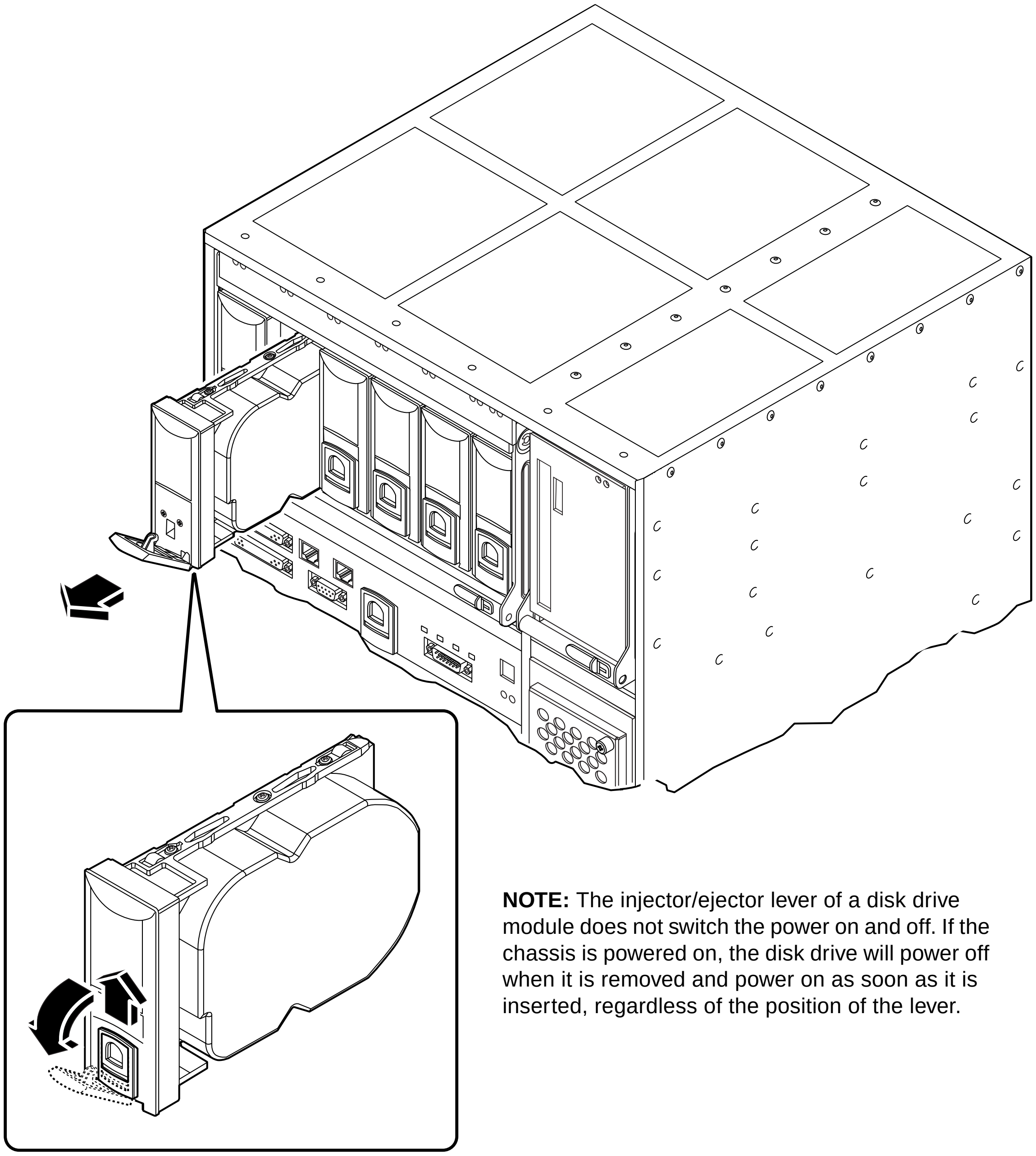
   Use the arrow keys to select the disk chassis in the list, then type **`D`**.

   When the state of the module changes to `disabled`, the module can be removed. If the disk chassis is not disabled successfully, check that any HDD modules in it are disabled and removed.

3. **Slide the latch in the handle of the disk chassis to the left (towards the 'unlocked' symbol) to disengage the power supply.**

4. **Lift the handle.**

5. **Slide the disk chassis out of its slot.**



NOTE: The injector/ejector lever of a disk drive module does not switch the power on and off. If the chassis is powered on, the disk drive will power off when it is removed and power on as soon as it is inserted, regardless of the position of the lever.

FIGURE 12-7 Removing a Disk Drive

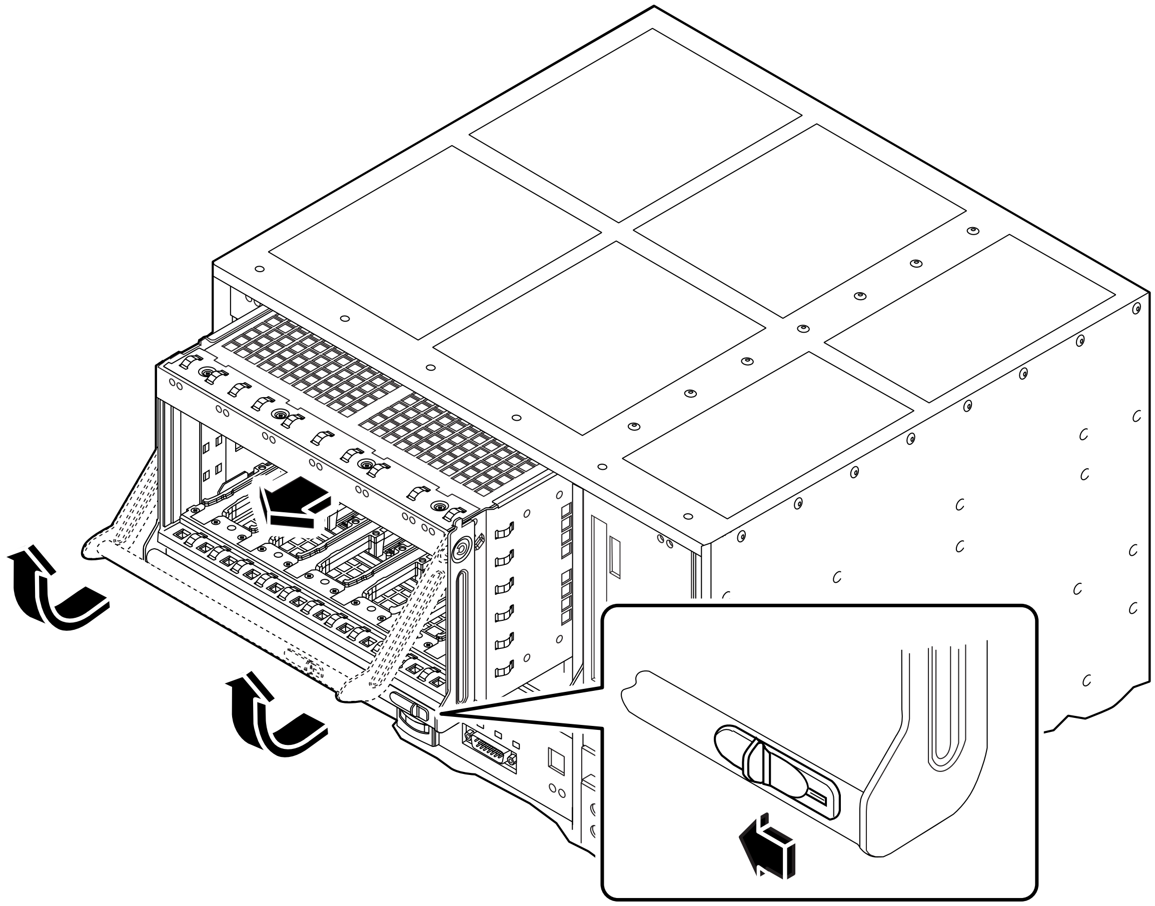


FIGURE 12-8 Removing a DSK Module

▼ To Replace the Disk Chassis

1. **Slide the disk chassis into its slot until it is almost completely in.**
2. **Lower the handle to engage the disk chassis fully in its slot.**
3. **Slide the latch in the handle to the right (towards the 'locked' symbol) to engage the power supply.**
4. **Configure the disk chassis into the system.**

a. Start `cmsfix`**.**

```
# cmsfix
```

A list of modules is displayed, including the new disk chassis.

b. Configure the disk chassis

Use the arrow keys to select the disk chassis in the list and type `C`.

The state of the disk chassis changes to `configured`. You can now replace the individual disk drives.

12.5 Replacing a Motherboard

The upper motherboard (A-MBD) and the lower motherboard (B-MBD) are removed and replaced in almost exactly the same way.

Caution – The wrist-strap provided must be used when replacing modules, or making cable connections to the rear of the system. The wrist-strap connection point on the Netra ft 1800 system is located on the panel at the bottom rear of the chassis (see FIGURE 12-9).

Caution – Only one motherboard should be replaced at a time. If it is necessary to replace both motherboards, complete the full replacement procedure for one motherboard and ensure the system is running correctly before attempting to replace the second motherboard.

Note – The securing screws for motherboard A are black. The securing screws for motherboard B are silver.

Note – All the securing screws are captive and spring-loaded, and require a No. 2 Phillips screwdriver.

Note – The special tools required (CPUset module locking and motherboard ejection tools) are housed in the clips on the outside of the mid cover.

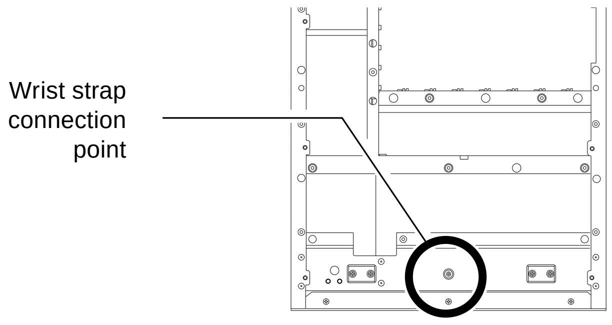


FIGURE 12-9 Wrist Strap Connection Point

▼ To Remove a Motherboard

1. **Remove the replacement motherboard from its packaging.**

   Lay the new motherboard on the black side of its protective antistatic sheet until it is required. You will also need to remove the plastic sleeves from the motherboard securing screws.

2. **Log on as root using the console on the side of the system which is to remain running.**

   You can also `rlogin` as root, assuming the `CONSOLE` line in `/etc/default/login` is not commented out.

3. **Use `cmsfix` or `cmsconfig` to disable all the modules connected to the motherboard which is to be replaced.**

   Refer to Section 4.2 "The `cmsconfig` Utility" on page 4-4

   The PCI cards, HDDs and DSK module and RMM should be disabled first, followed by the corresponding CPUset module and the CAF. Disabling the motherboard automatically disables the associated PSUs.

4. **Disable the faulty motherboard using `cmsfix`.**

```
# cmsfix
```

   The faulty motherboard appears in the list.

   c. **Use the arrow keys to select the faulty motherboard and acknowledge the fault by typing `A`.**

   d. **Disable the motherboard by typing `D`.**

   When the state of the motherboard changes to `disabled`, you can continue the procedure.

5. **Unlock the ejector slides on all disabled modules.**

   The red warning dots will show.

6. **Open the external power breakers associated with the PSUs on the motherboard to be replaced.**

7. **Unplug all the modules which are now disabled.**

Caution – You must completely remove all the HDD modules before unplugging the disk chassis. Make a note of the location of each HDD module as they must be re-inserted in the same locations.

Note – When removing modules on side A, unplug the CPUset *before* unplugging the PCI modules.

There is no need to remove the modules (apart from the HDDs) completely from their slots, or to remove blanking panels from unused slots.

The *Diag* LED on the remaining CPUset will flash slowly during this procedure.

8. **Loosen the four screws that secure the mid cover.**

   Refer to FIGURE 12-10. Lift off the cover and place it out of the way of the work area.

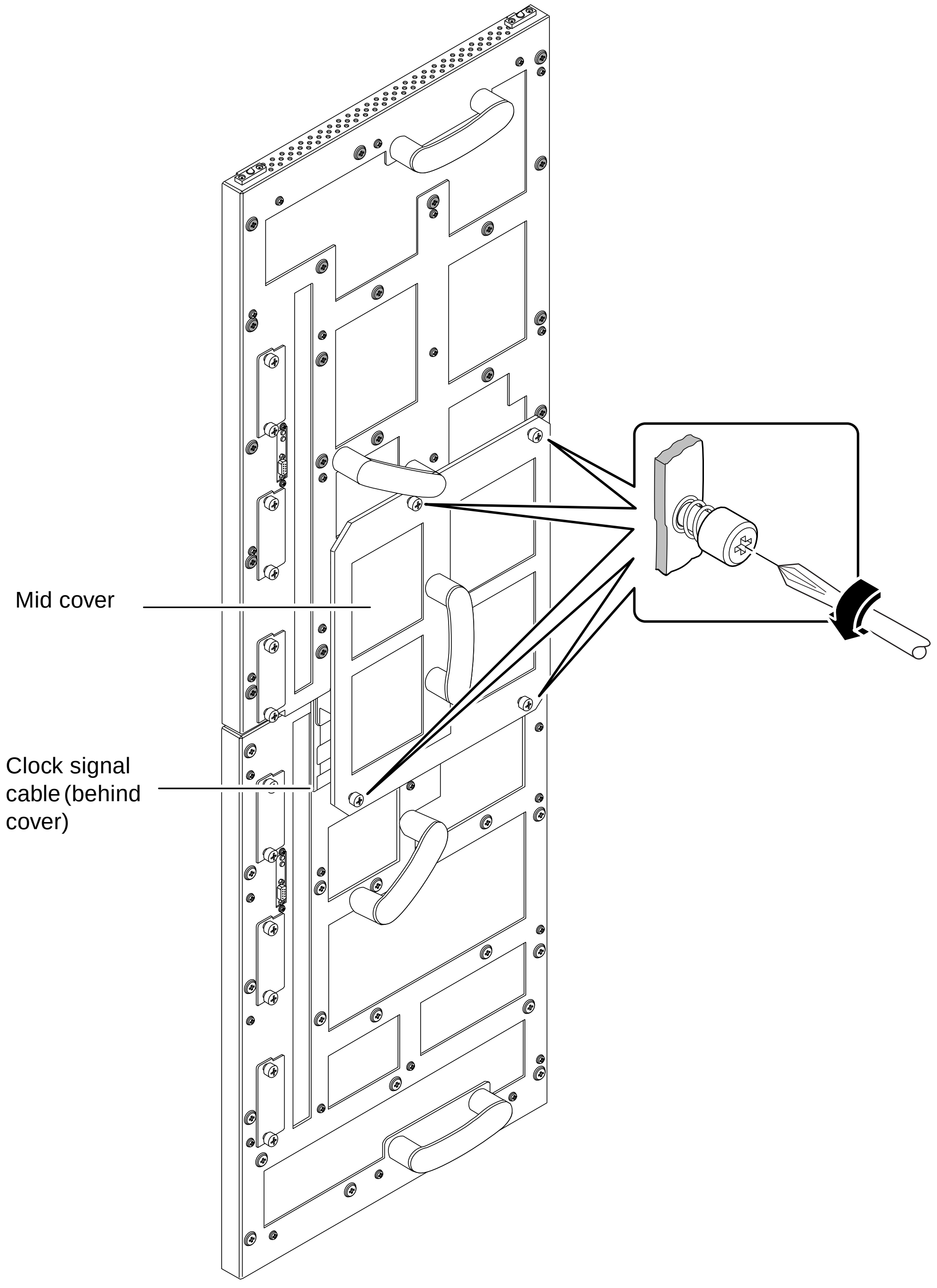


FIGURE 12-10 Location of Mid Cover Securing Screws and Clock Signal Cable

9. **Gently pull off the brass connector that secures the clock signal coaxial cable.**

   Refer to FIGURE 12-10.

Caution – Take care to secure the connector well away from the motherboard.

---

10. **Remove the power inlet connectors from the motherboard to be replaced.**

    Unscrew the two securing screws on each inlet connector, then secure the connectors and cables clear of the rear of the system.

11. **Insert two of the CPUset module locking tools into the holes provided.**

    Refer to FIGURE 12-12. Hand-tighten the tools to secure the remaining CPUset module to the motherboard which is to remain in the chassis.

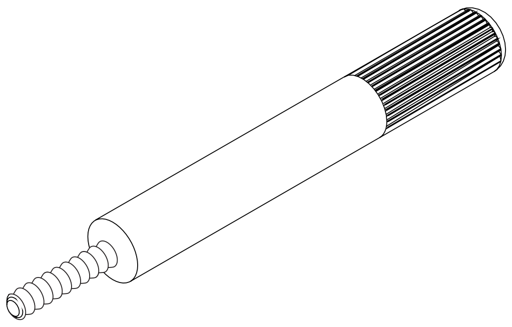


FIGURE 12-11 CPUset Module Locking and Motherboard Ejection Tool

---

Note – The tools have knurled handles to prevent them being overtightened.

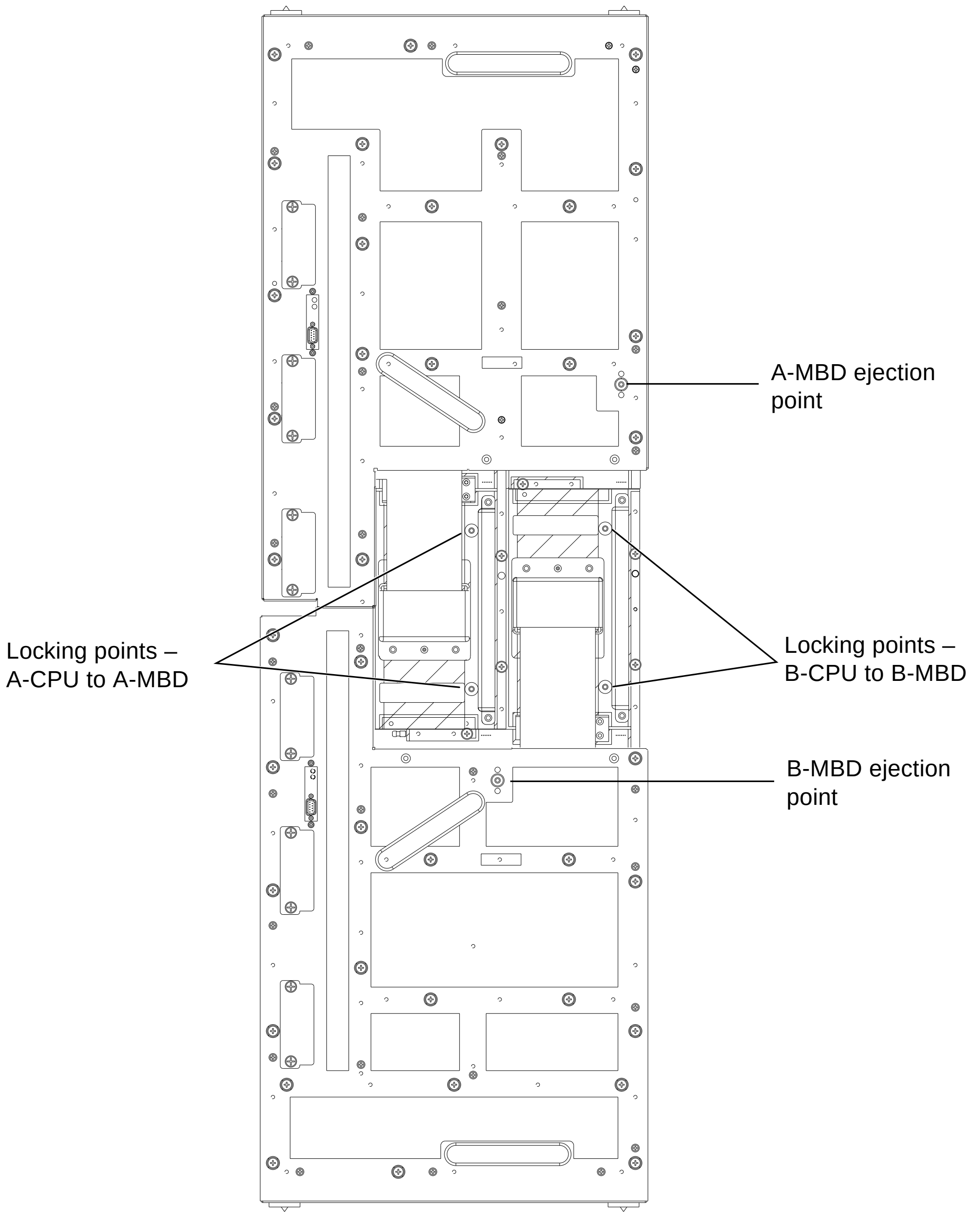

A-MBD ejection point
Locking points – A-CPU to A-MBD
Locking points – B-CPU to B-MBD
B-MBD ejection point

FIGURE 12-12 Location of CPUset Module Locking and Motherboard Ejection Points

12. **Loosen all 22 (A-MBD) or 23 (B-MBD) captive screws securing the motherboard to the chassis (see** FIGURE 12-13**).**

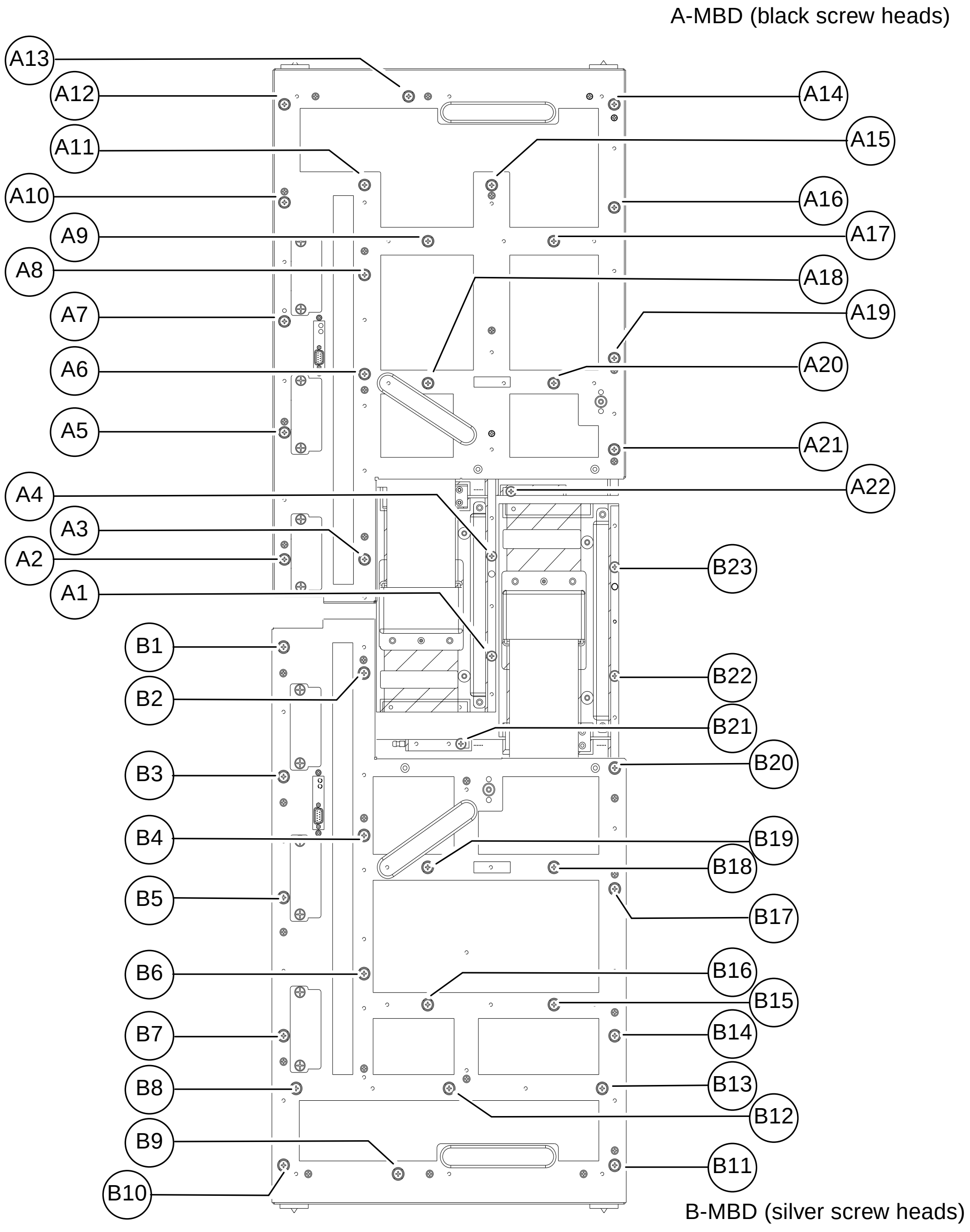


FIGURE 12-13 Motherboard Securing Screws

Note – The securing screws for motherboard A are black. The securing screws for motherboard B are silver.

Ensure all the screws are free of their threads before proceeding.

13. **Insert the motherboard locking tool into the appropriate hole.**

    Refer to FIGURE 12-12 on page 12-23. Gently tighten the tool to lift the motherboard away from the working CPUset module.

    Note – The tool has a knurled handle to prevent it being overtightened.

14. **Using the handles provided, pull the motherboard gently away from the chassis and off the guide pins.**

    Caution – The motherboard weighs 11.34 kg (23 lbs). Make sure that there is a clear area to which you can transfer the motherboard once it is removed from the chassis.

15. **Remove the ejector tool and replace it in the clip on the mid cover.**

▼ To Replace a Motherboard

1. **Locate the three guide pins which position the motherboard.**

   Refer to FIGURE 12-14.

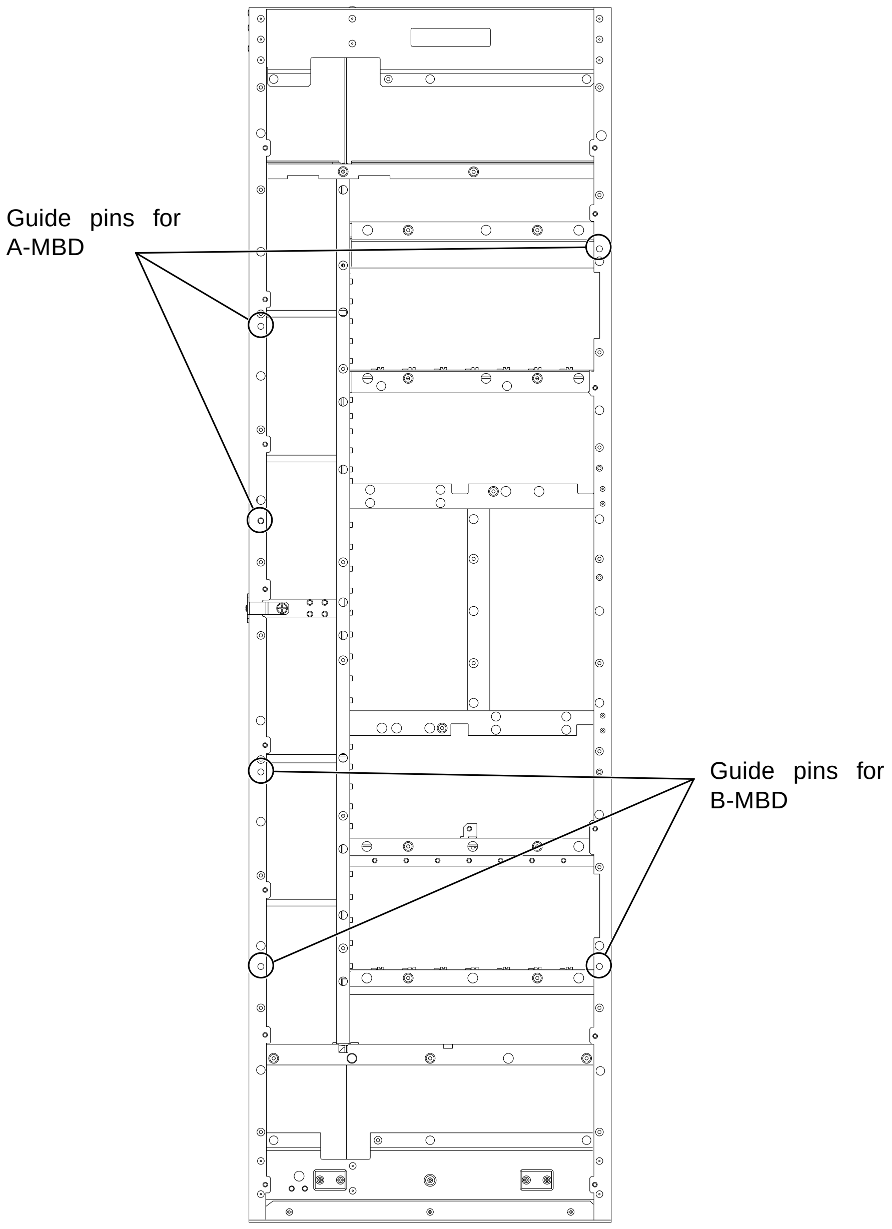


FIGURE 12-14 Location of Motherboard Guide Pins

2. **Lift the motherboard by its handles and feed it gently on to the guide pins.**

Note – Motherboard B can be supported on the two plastic blocks at the base of the chassis, at which point it is in the correct position vertically.

Ensure that the clock signal cable is not trapped and that the brass connector is secured out of the way.

3. **Push the motherboard gently but firmly home.**
4. **Tighten the captive securing screws.**

   Tighten one of the uppermost screws first so that the motherboard is held in position, then tighten the screws around the CPUset module. Do not overtighten the screws: maximum torque permitted is 5.4 Nm (4 lb/ft).
5. **Connect the clock signal cable, pushing the connector home firmly.**

   Ensure the cable itself is routed such that it will not be trapped by the mid cover.
6. **Remove the two CPUset module locking tools and replace them in their clips on the mid cover.**
7. **Insert the power inlet connectors and tighten their securing screws.**

   Do not overtighten the screws: maximum torque permitted is 5.4 Nm (4 lb/ft).
8. **Replace the mid cover, tightening the four captive retaining screws.**

   Do not overtighten the screws: maximum torque permitted is 5.4 Nm (4 lb/ft).
9. **Re-insert the modules that were previously withdrawn and close the injector latches.**
10. **Close the external circuit breakers associated with the PSUs which have been re-inserted.**
11. **Update the EEPROM on the new motherboard.**

    If you have replaced motherboard A, type:

    ```
    # cmsintroduce_mbd A-MBD
    ```

    If you have replaced motherboard B, type:

    ```
    # cmsintroduce_mbd B-MBD
    ```
12. **Use `cmsconfig` to enable the new motherboard.**

    Refer to Section 4.2 “The `cmsconfig` Utility” on page 4-4.

13. **Use `cmsconfig` to enable the associated modules, starting with the CAF and CPUset module.**

    Refer to Section 4.2 “The `cmsconfig` Utility” on page 4-4.

12.6 Changing the Environmental Filters

There are three filter trays at the base of the chassis, two large ones and one small one. These are inserted in place of the sacrificial shipping plinth at installation time.

Each power supply module is also fitted with a filter.

The filters (Filter Kit X-Option No. X6952A) should be changed at least once every six months.

12.6.1 Main System Filters

The trays are secured by thumbscrews and incorporate handles for easy removal and insertion.

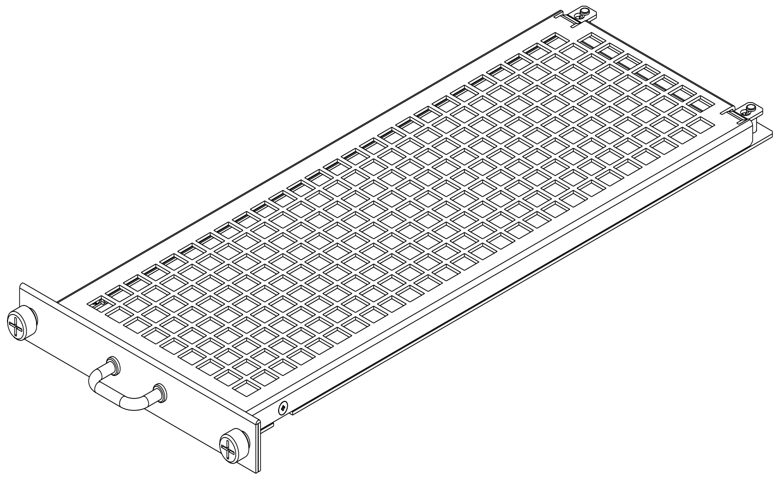


FIGURE 12-15 Large Filter Tray

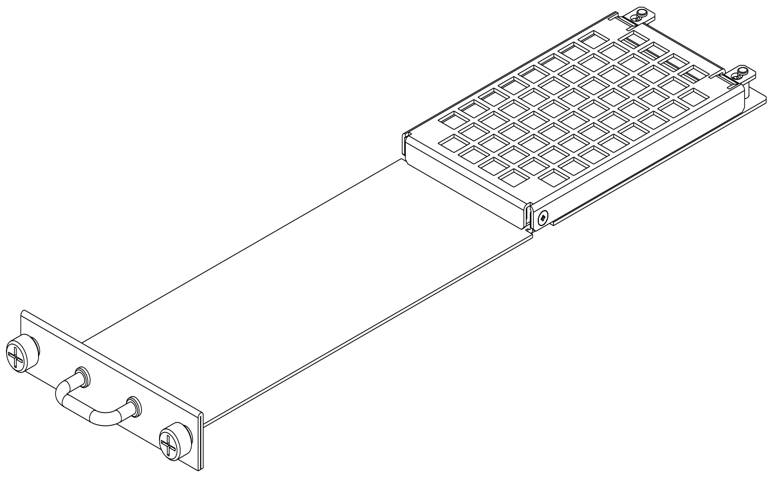


FIGURE 12-16 Small Filter Tray

▼ To Remove a Filter Tray

1. **Unscrew the two thumbscrews on the front of the tray (see** FIGURE 12-15 **and** FIGURE 12-16**)**
2. **Slide the tray out using the handle provided.**

▼ To Replace a Filter

1. **Using a Phillips No.2 screwdriver, remove the two countersunk screws securing the filter housing to the tray body.**

   Refer to FIGURE 12-17 and FIGURE 12-18.

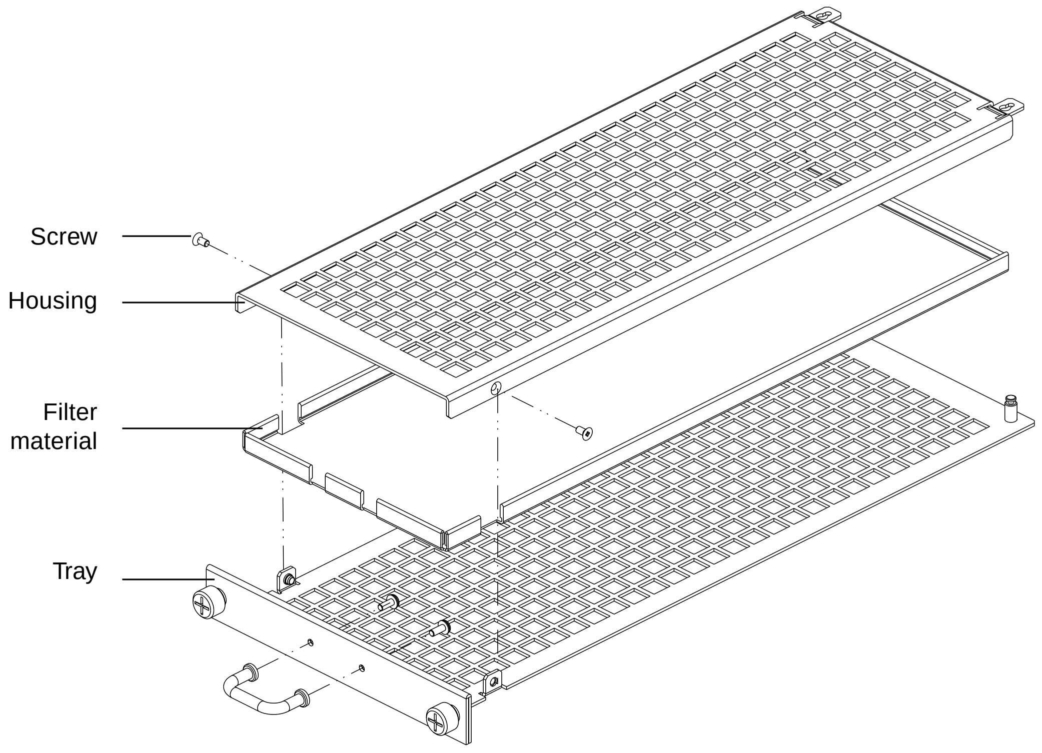


FIGURE 12-17 Dismantling the Large Filter Tray

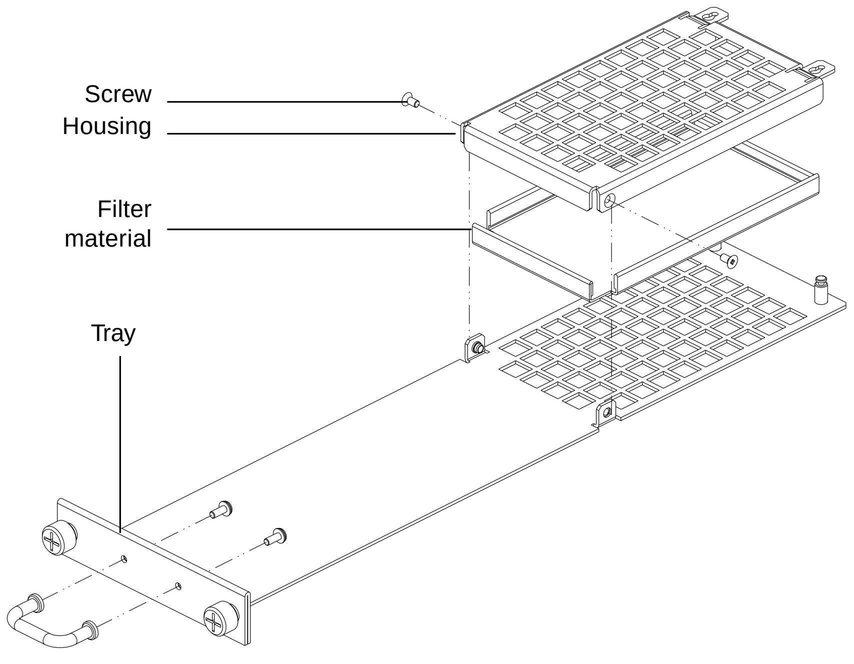


FIGURE 12-18 Dismantling the Small Filter Tray

2. **Slide the filter housing towards the rear of the tray to release it from the grooved posts.**
3. **Lift the filter housing off the tray and remove the filter material.**
4. **Insert the new filter material, supplied as X-Option No.X6952A.**
5. **Place the housing on to the tray with the grooved posts inserted in the keyholes at the rear and push it forwards whilst pressing down slightly.**

   The keyholes should engage with the grooves in the posts.
6. **Insert the two screws and tighten them carefully.**

   Be careful not to overtighten these screws.

▼ To Replace a Filter Tray

1. **Insert the tray into the appropriate aperture at the base of the chassis; the two large trays are fitted on the left and the small tray on the right.**

   It is unimportant which way up the tray is fitted.

2. **Push the tray securely home and hand-tighten the two thumbscrews.**

12.6.2 PSU Filters

The power supply filters are located immediately behind the front air inlet grille of the PSU modules.

▼ To Change a PSU Filter

1. **Unscrew the two filter cover captive screws.**

   You can unscrew these by hand. If they have been over-tightened previously, it may be necessary to use a No.2 Phillips screwdriver to loosen them.

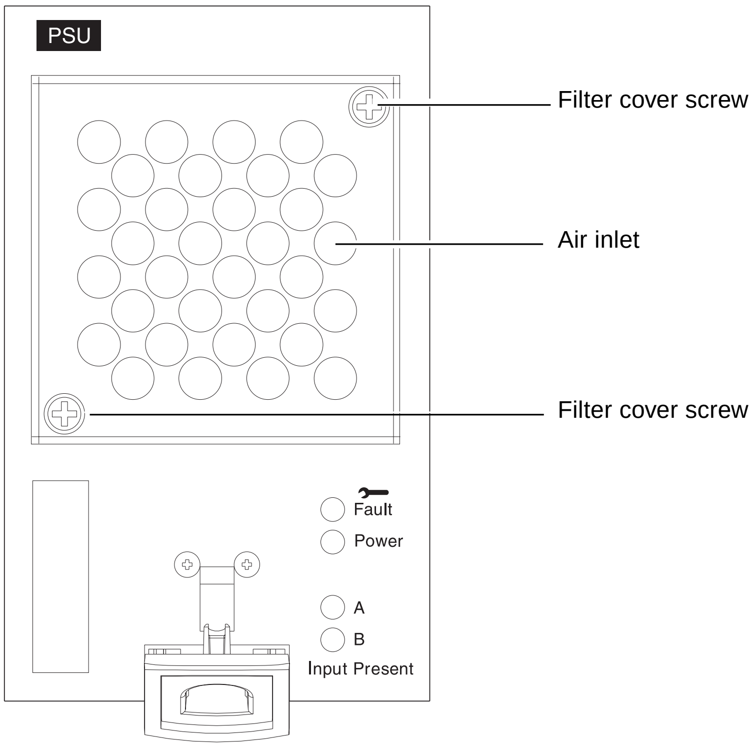


FIGURE 12-19 PSU Filter Cover

2. **Remove the filter cover and the filter material.**

3. **Place a new filter over the fan cover.**

   The filter will be held in place by the air flow generated by the fan.

4. **Replace the filter cover and hand-tighten the two captive securing screws. It is not necessary to use a screwdriver.**

APPENDIX **A**

CMS Man Pages

This appendix contains the man pages for the CMS utilities for the Netra ft 1800. It includes the following man pages:

- `cmsconfig` (1M)
- `cmsfix` (1M)
- `cmsfruinfo` (1M)
- `cmsintroduce_mbd` (1M)
- `cmsledctl` (1M)
- `cmsphonehome` (1M)
- `xcmsfix` (1M)

NAME

cmsconfig - user interface to the Configuration Management System

SYNOPSIS

cmsconfig [-w] [-m *module* **][-n** *number* **] [-a** *attribute* **][-v** *value* **] [-q] [-V] [-s] [-A]**

DESCRIPTION

cmsconfig is the user interface of the Configuration Management System (CMS). It provides the user with access to the description of the system configuration which is maintained by **cmsd**(1M). **cmsconfig** is normally used when reconfiguring a system's hardware. A separate utility, **cmsfix**(1M), is available for fault management and correction.

The hardware of a system is divided by the CMS into modules and subsystems (see **cmsd**(1M) and **cmsdef**(4). **cmsconfig** enables a user to examine a subset of the full range of modules and subsystems in the system. When invoked with no arguments, **cmsconfig** provides the following commands:

I Include a list of modules into the current selection.

E Exclude a list of modules from the current selection.

number
Examine in detail the module with this item number.

P If the list is longer than a screen, page down within the list.

PN If the list is longer than a screen, page down within the list.

PP If the list is longer than a screen, page up within the list.

Pnumber
If the list is longer than a screen, specify a page number within the list.

S Select a named object

T Go to the top-level menu

H or ?
Print a help message describing the available options.

Q Exit from **cmsconfig.**

(These commands are not case-sensitive.)

Pressing RETURN at the prompt redraws the screen. This can be used to check whether any of the modules' states have changed. When specifying a list of modules for **I**and**E**, the following options are available:

dep[endents]*n*
All the dependents of the module whose item number in the current list is *n*.

dep[endents]*n-m*
All the dependents of the module whose item number in the current list is between *n* and *m*, inclusive.

con[stituents]*n*
All the constituents of the module whose item number in the current selection is *n.*

con[stituents]*n-m*
All the constituents of the module whose item number in the current selection is between *n* and *m*, inclusive.

all All modules and objects in the system.

alldep[endents]*n*
: Similar to dependents but also includes the dependents' dependents, and so on.

all*name*
: All modules having the attribute or constituent *name.*

all*name***==***value*
: All modules whose attribute or constituent *name* is equal to *value.*

all*name***!=***value*
: All modules whose attribute or constituent *name* is not equal to *value.*

all default
: A synonym for **all _show_by_default != no**

all present
: A synonym for **all***location***!=NULL**

name All modules whose name starts with *name.*

name number
: The specified module only.

name low - high
: The range of modules from *name low* to *name high.*

To select an item in any of the lists that are displayed, enter its item number. To modify an attribute or constituent of a module, select the module. A list is displayed of the module's attributes and constituents and their current values. Selecting an attribute or constituent displays its possible values. Select a value and press RETURN to set the value. Pressing RETURN without selecting a value on this screen will cause a return to the list of attributes and constituents. In all other cases, pressing RETURN at the prompt will refresh the screen, enabling any changes in the attributes to be seen.

cmsconfig prevents the user from modifying a **system_attribute.** It also checks any modification request against a list of rules taken from the file **/usr/platform/SUNW,Ultra-4FT/SUNWcms/etc/cmsconfig.rule**. These rules stop operations which will seriously affect system integrity (such as switching off the last CPU or power supply) and warn the user before allowing an operation which would cause a loss of functionality in the system.

The following command line options are recognised by **cmsconfig**:

-w Suppress warning messages generated from the rules file.

-m*name*
: Specify a set of modules to be examined. When not combined with other options, this is equivalent to using **I***name.*

-n*number*
: This should be used with the **-m** option to specify a particular module or object. When only **-m** and **-n** are used, this is the same as typing **S***namenumber* as the first command to **cmsconfig**.

-a*attribute*
: This should be used with the **-m** and **-n** options to specify an attribute or constituent of a module to be changed. If just **-m, -n** and **-a** are given, **cmsconfig** will prompt for a new value and attempt to change the given module.

-v*value*
: When all of **-m, -n, -a** and **-v** are given, **cmsconfig** will attempt to set *attribute* to *value*. All rule checks will happen as in the interactive version.

-s Suppress all error and warning messages by redirecting stderr to */dev/null*.

-V*view*
Use the attribute specified by *view* to obtain the text displayed in the rightmost column of the top-level screen. By default, the user_label attribute is used.

-A Display all attributes and their values. By default, attributes that begin with an underscore character are not displayed.

-q This should be used with the **-m, -n** and **-a** options to query the current value of the attribute specified by the **-a** option. The value is printed on the standard output, after which **cmsconfig** exits. This option is primarily intended for use within shell scripts.

FILES

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsconfig
/etc/SUNWcms/.usersocket
Unix domain socket for communicating with **cmsd**(1M).
/usr/platform/SUNW,Ultra-4FT/SUNWcms/etc/cmsconfig.rule
Rules used to generate errors and warnings.

SEE ALSO
cmsd(1M), **cmsfix**(1M)

NAME

cmsfix - user interface to configuration management system

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsfix [-a] [-V *view***]**
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsfix start

DESCRIPTION

cmsfix provides a simple interface to the CMS, to be used when part of the hardware of a Netra ft 1800 computer has failed. It allows the user to see which modules contain faults and to enable and disable modules. **cmsfix** can be used to acknowledge faults.

When **cmsfix** is invoked with no arguments, it presents a visual interface which consists of a list of the modules in the computer which are faulty or disabled. Invoking **cmsfix** with the **-a** option includes all modules in the display. The user can select a particular module to configure, to unconfigure or to acknowledge a fault in by using the cursor keys. The currently selected module is highlighted on the screen.

OPTIONS

-V*view*
Use the specified attribute to obtain the text displayed in the rightmost column of the top-level screen. By default, the user_label attribute is used.

The following commands are available from this visual interface:

D Attempt to disable the currently selected module.

E Attempt to enable the currently selected module.

A Acknowledge the fault of the currently selected module.

Return
Redraw the screen.

N Display the next page of modules.

P Display the previous page of modules.

S Rescan the computer’s hardware for modules whose state is not as it should be.

up,**K** Move the selection up one line.

down,**J**
Move the selection down one line.

? Display a help screen

Q Exit from cmsfix.

cmsfix also provides a non-interactive command-line interface for triggering the initialization of CMS objects at system startup.

FILES

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsfix
/usr/platform/SUNW,Ultra-4FT/SUNWcms/etc/cmsconfig.rule
Rules used to generate errors and warnings.

NAME

cmsfruinfo – displays information in a module's EEPROM

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsfruinfo –l*location* [**–isdgm**] *field*...

DESCRIPTION

Most modules in a Netra ft 1800 system have an EEPROM in which information about the module is stored; the **cmsfruinfo** command is used to display this information. The data in the EEPROM can be divided into two areas: a generic part, which is common to all modules, and a machine specific part. The latter also has a "variant" part, which is module class specific. A list of the fields stored in the EEPROMs can be found in the **cms_eeprom_field**(5) manual page.

OPTIONS

The following options are supported:

–i Selecting this option will cause any EEPROM checksum errors to be non-fatal. By default, checksum errors cause an abortive exit, after printing an error message. Specifying **–i** will still print out the error messages (so that the user will know that the data is unreliable), but processing is allowed to continue.

–l*location*
The location of the module. The format of valid locations is specified in **cms_location**(5).

–s This option prints the value of the specified field only. By default, the field name is printed alongside the value.

–d display the date using words and numbers. By default, the date is displayed as a number of elapsed seconds since 00:00 Universal Cardinal Time, that is, January 1 , 1970.

–g print the EE_GEN_GENVERS version number

–m print the EE_MSP_MSPVERS version number

OPERANDS

The following operand is supported:

field
The field name(s) to print the value of.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

The attribute Availability has the value SUNWcmsf. See **attributes**(5) for a description of this attribute.

SEE ALSO

attributes(5), **cms_eeprom_field**(5), **cms_location**(5)

NAME

cmsintroduce_mbd - introduce a new MBD to the system

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsintroduce_mbd A-MBD
/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsintroduce_mbd B-MBD

DESCRIPTION

On the Netra ft 1800, some of the data that identifies the system is stored in the motherboards (MBDs). When an MBD is replaced, you must therefore use **cmsintroduce_mbd** to copy this data from the running MBD into the new MBD.

cmsintroduce_mbd takes a single argument, the location of the new MBD, and copies the relevant system data into this location. The new MBD must be powered-off (which would normally be the case during MBD replacement). If it is not, an error message is output and no copying takes place.

cmsintroduce_mbd should only be run as part of the MBD replacement process. See the *Netra ft 1800 User Guide* for details.

cmsintroduce_mbd must be run every time a MBD is replaced to copy the system identity to it, then again after the second one is replaced.

EXIT STATUS

0	The system identity was successfully written.
non-zero	An error occurred.

SEE ALSO

cmsfruinfo(1M)

NAME

cmsledctl – fault LED latent fault checker

SYNOPSIS

/opt/SUNWcms/bin/cmsledctl –l*location* [**–f | –n**]
cmsledctl –l*location* [**–d***delay*] [**–o***off_delay*] [**-v**]
cmsledctl –a [**–d***delay*] [**–o***off_delay*] [**-v**]
cmsledctl –r [**–d***delay*] [**–o***off_delay*] [**-v**]

DESCRIPTION

The **cmsledctl** command controls the fault LED of the module at *location*, turning it on or off. It also provides a means to test the fault LEDs for latent faults, by attempting to turn them on and off. At the end of the test, the fault LED is restored to its correct state. If no other arguments except *location* are specified, the status of the selected LED is printed on stdout.

Turning on an LED that is already on has no effect, as does turning off an already extinguished one.

OPTIONS

The following options are supported:

–a Run the test on all fault LEDs simultaneously.

–d*delay*
After turning on a fault LED, wait for *delay* seconds before continuing.

–f Turn the fault LED on the specified module off. This option cannot be used with any other one, with the exception of **–l** or **–v**.

–l*location*
Apply the command only to the fault LED on the module in *location.* The format of valid locations is specified in **cms_location**(5).

–n Turn the fault LED on the specified module on. This option cannot be used with any other one, with the exception of **–l** or **–v**.

–o*off_delay*
After turning off a fault LED, wait for *off_delay* seconds before continuing.

–r Run the test on all LEDs sequentially.

–v Verbose mode. When **–r** is specified, prints out the state(s) applied to each LED. When **–r** is not specified, has no effect.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcmsf

SEE ALSO

attributes(5), **cms_location**(5)

NAME

cmsphonehome - report a failure by phoning home

SYNOPSIS

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/cmsphonehome *module module_no location old_state new_state*

DESCRIPTION

cmsphonehome is a script called by the CMS when any module fails. It is intended that this script can be altered to report device failures to an appropriate place. This could be to send a mail message or to communicate with an alarm system.

The arguments passed to it are:

module
: device type as known to the CMS

module_no
: device number as known to the CMS

location
: location of the FRU in which the device resides

old_state
: previous state of the device before the failure

new_state
: current state of the device

cmsconfig can be used to read further information (attributes) from the CMS if required.

The default script has an example of how to build up a mail message but this is commented out.

SEE ALSO

cmsconfig(1M), *Netra ft 1800 CMS Developer's Guide*

NAME

xcmsfix - X Window System interface to the configuration management system

SYNOPSIS

xcmsfix [-nospawn, -all (and standard X toolkit options) **] [-V** *view***]**

DESCRIPTION

xcmsfix provides a simple graphical interface to the CMS, to be used when part of the hardware of a Netra ft 1800 system has failed. It allows the user to see which modules contain faults and to enable and disable modules. **xcmsfix** can be used to acknowledge faults.

When **xcmsfix** is invoked with no arguments, it presents a visual interface which consists of a list of the modules in the computer which are faulty or disabled. Invoking **xcmsfix** with the **-nospawn** option prevents forking off a child process in which to run. This option stops the program from behaving like a daemon.

Invoking **xcmsfix** with the **-a** option includes all modules in the display. The user can select a particular module to configure, to unconfigure or to acknowledge a fault by using the cursor keys, the mouse or both. The currently-selected module is highlighted on the screen in reverse video.

A scrollbar is displayed when there is insufficient room to display all the faulty modules. This can be adjusted either with the mouse (using the BSelect mouse button, normally button 1) or with the Page Up and Page Down keys.

A module can be selected by moving the mouse pointer to the desired item and when the BSelect mouse button is clicked the item is selected.

Alternatively the key defined as KSelect (normally Space) can be used to select a module using the keyboard.

The reason for the currently selected module's failure is displayed in the lower right hand corner of the **xcmsfix** window. While **xcmsfix** is busy a watch cursor is displayed and the status changes from "READY" to "BUSY".

OPTIONS

-V*view*

Use the specified attribute to obtain the text displayed in the rightmost column of the top-level screen. By default, the user_label attribute is used.

The following commands are available:

The File menu has entries for the following operations and mnemonics:

Operation	Mnemonic	Operation
Unconfigure	U	Attempt to unconfigure the currently selected module
Configure	C	Attempt to configure the currently selected module
Acknowledge	A	Acknowledge the fault of the currently selected module
Rescan	S	Rescan the computer's hardware for faulty modules.
Redraw	R	Redraw the xcmsfix window.
Quit	Q	Quit xcmsfix.

The mnemonics for these functions can be used as standard Motif menu mnemonics but have also been added as accelerators, i.e. they can be typed directly (in upper or lower case).

The Fix and Acknowledge and Rescan functions are available on pushbuttons at the base of the screen.

The Help Menu has entries for the following operations and mnemonics:

Operation	Mnemonic	Operation
About	A	Display xcmsfix revision information.
Use	U	Display xcmsfix help dialog.

The Help key can also be used to display the help dialog.

To set an alternative font in a resource file both the *fontList and *font resources should be set. (The fontList resource is used by all the widgets, but resizing control is determined by the font resource of the automatically created XmScreen widget.)

To use the 12x24 font by default, set these values in the application defaults file:

```
*font: 12x24

*fontList: 12x24
```

FILES

/usr/platform/SUNW,Ultra-4FT/SUNWcms/sbin/xcmsfix
/etc/SUNWcms/.locks/xcmsfix.lock
lock file.
/etc/SUNWcms/.locks/xcmsfix.prep.lock
preparatory lock file.
/etc/SUNWcms/.usersocket
Unix domain socket for communicating with **cmsd**(1M).
/usr/platform/SUNW,Ultra4-FT/SUNWcms/etc/cmsconfig.rule
Rules used to generate errors and warnings.

SEE ALSO

cmsfix(1M), **cmsconfig**(1M)

APPENDIX **B**

Open Boot PROM Information

This document describes commands and features of the Netra ft 1800 Open Boot Prom (OBP) that differ from the corresponding commands or features of other Sun Ultra PCI products. Read this document in conjunction with the *OBP 3.x Command Reference Manual* (Part No. 802-5837-10).

B.1 Overview

The architecture of the Netra ft 1800 is based on that of the Ultra 450. The structure of Netra ft 1800 OBP closely follows that of the Ultra 450 with some differences of detail. The Netra ft 1800 has a number of features that are not present in other Sun Ultra PCI systems.

The Netra ft 1800 OBP contains commands to control and inspect the new features.

Significant differences between the Netra ft 1800 and the Ultra 450 include:

- The Netra ft 1800 has two sides which are independently powered.
- The system can be configured as two independent systems with different identities.
- The maintenance bus architecture is completely different.
- Each module in the system can be independently powered on or off.
- There are four built-in network connections.
- The console is duplicated.

B.2 Features of the Netra ft 1800 System

The following subsection describe the features of the Netra ft 1800.

B.2.1 Device Tree

The devices listed below are specific to the Netra ft 1800 system:

- The default console device and a support package for it. See Section B.3.4 "Console" on page B-11 for a description of the consoles.

```
/u4ftser@0,0
/packages/console-pkg
```

- A device containing NVRAM options which are specific to the Netra ft 1800

```
/u4ft-options
```

- A new flash PROM node to reflect the fact that the system has two flash PROMs

```
/pci@1f,4000/ebus@1/flashprom@10,200000
```

- A new `EEPROM` node to reflect the fact that the system has an extra configuration NVRAM

```
/pci@1f,4000/ebus@1/u4ft-eeprom@14,100000
```

A hierarchy of nodes supports the fault tolerant and hot-plugging aspects of the Netra ft 1800. The actual I/O nodes are children of the `u4ioslot` nodes, described on page B-4.

The table below shows the new nodes and translates them to the corresponding location name. The translation is non-trivial.

TABLE B-1 Node/slot Translations

Name	Location name
`/pci@6,2000/u4ftbus@0`	-
`/pci@6,2000/u4ftbus@0/u4ioslot@7`	-
`/pci@6,2000/u4ftbus@0/u4ioslot@6`	-
`/pci@6,2000/u4ftbus@0/u4ioslot@5`	a-dsk
`/pci@6,2000/u4ftbus@0/u4ioslot@4`	a-caf
`/pci@6,2000/u4ftbus@0/u4ioslot@3`	a-pci4
`/pci@6,2000/u4ftbus@0/u4ioslot@2`	a-pci5
`/pci@6,2000/u4ftbus@0/u4ioslot@1`	a-pci6
`/pci@6,2000/u4ftbus@0/u4ioslot@0`	a-pci7
`/pci@6,4000/u4ftbus@1`	-
`/pci@6,4000/u4ftbus@1/u4ioslot@7`	-
`/pci@6,4000/u4ftbus@1/u4ioslot@6`	-
`/pci@6,4000/u4ftbus@1/u4ioslot@5`	a-rmm
`/pci@6,4000/u4ftbus@1/u4ioslot@4`	a-caf
`/pci@6,4000/u4ftbus@1/u4ioslot@3`	a-pci0
`/pci@6,4000/u4ftbus@1/u4ioslot@2`	a-pci1
`/pci@6,4000/u4ftbus@1/u4ioslot@1`	a-pci2
`/pci@6,4000/u4ftbus@1/u4ioslot@0`	a-pci3
`/pci@4,2000/u4ftbus@0`	-
`/pci@4,2000/u4ftbus@0/u4ioslot@7`	-
`/pci@4,2000/u4ftbus@0/u4ioslot@6`	-
`/pci@4,2000/u4ftbus@0/u4ioslot@5`	b-dsk
`/pci@4,2000/u4ftbus@0/u4ioslot@4`	b-caf
`/pci@4,2000/u4ftbus@0/u4ioslot@3`	b-pci4
`/pci@4,2000/u4ftbus@0/u4ioslot@2`	b-pci5
`/pci@4,2000/u4ftbus@0/u4ioslot@1`	b-pci6
`/pci@4,2000/u4ftbus@0/u4ioslot@0`	b-pci7
`/pci@4,4000/u4ftbus@1`	-

TABLE B-1 Node/slot Translations *(Continued)*

Name	Location name
`/pci@4,4000/u4ftbus@1/u4ioslot@7`	-
`/pci@4,4000/u4ftbus@1/u4ioslot@6`	-
`/pci@4,4000/u4ftbus@1/u4ioslot@5`	b-rmm
`/pci@4,4000/u4ftbus@1/u4ioslot@4`	b-caf
`/pci@4,4000/u4ftbus@1/u4ioslot@3`	b-pci0
`/pci@4,4000/u4ftbus@1/u4ioslot@2`	b-pci1
`/pci@4,4000/u4ftbus@1/u4ioslot@1`	b-pci2
`/pci@4,4000/u4ftbus@1/u4ioslot@0`	b-pci3

The built-in I/O nodes for the Netra ft 1800 are always present. If, however, the corresponding motherboard is not present, the `u4ioslot` nodes for the devices on that motherboard have the `status` property set to `disabled` and the child devices will not be usable. This also applies to devices which are present but have not yet been opened. The OBP initialization sequence attempts to open all of the `ioslot` devices corresponding to the SCSI and net nodes.

CODE EXAMPLE B-1 `ioslot` Devices

```
/pci@6,2000/u4ftbus@0/u4ioslot@7/u4fticc@7
/pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5
/pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/tape
/pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk
/pci@6,2000/u4ftbus@0/u4ioslot@4/network@4,1
/pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4
/pci@6,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000
/pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5
/pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5/tape
/pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk
/pci@6,4000/u4ftbus@1/u4ioslot@4/network@4,1
/pci@6,4000/u4ftbus@1/u4ioslot@4/ebus-test@4
/pci@4,2000/u4ftbus@0/u4ioslot@7/u4fticc@7
/pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5
/pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/tape
/pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk
/pci@4,2000/u4ftbus@0/u4ioslot@4/network@4,1
/pci@4,2000/u4ftbus@0/u4ioslot@4/ebus@4
/pci@4,2000/u4ftbus@0/u4ioslot@4/ebus@4/se@14,400000
/pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5
/pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5/tape
```

CODE EXAMPLE B-1 `ioslot` Devices *(Continued)*

```
/pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk
/pci@4,4000/u4ftbus@1/u4ioslot@4/network@4,1
/pci@4,4000/u4ftbus@1/u4ioslot@4/ebus-test@4
```

The plug-in cards are not probed by OBP during initialization as they would be on other Sun machines. The responsibility for configuring and initializing plug-in components now belongs to the operating environment, specifically to the Configuration Management System (CMS). It follows that fcode which is present on these cards is not run. There is presently no manual way of probing a plug-in card or of running its fcode.

B.2.2 Device Aliases

The aliases for the system can be displayed by entering the `devalias` command.

The following aliases are useful for booting. Aliases are useful mainly for debugging the system.

CODE EXAMPLE B-2 Aliases

```
b-dsk0       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@0
b-dsk1       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@1
b-dsk2       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@2
b-dsk3       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@3
b-dsk4       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@4
b-dsk5       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@5
b-dsk6       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@6
b-dsk7       /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@7
b-rmm0       /pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk@6
b-rmm1       /pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk@0
a-dsk0       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@0
a-dsk1       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@1
a-dsk2       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@2
a-dsk3       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@3
a-dsk4       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@4
a-dsk5       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@5
a-dsk6       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@6
a-dsk7       /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5/disk@7
a-rmm0       /pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk@6
a-rmm1       /pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5/disk@0
b-net0       /pci@4,2000/u4ftbus@0/u4ioslot@4/network@4,1
b-net1       /pci@4,4000/u4ftbus@1/u4ioslot@4/network@4,1
a-net0       /pci@6,2000/u4ftbus@0/u4ioslot@4/network@4,1
a-net1       /pci@6,4000/u4ftbus@1/u4ioslot@4/network@4,1
```

CODE EXAMPLE B-2 Aliases *(Continued)*

```
b-scsi1      /pci@4,4000/u4ftbus@1/u4ioslot@5/scsi@5
b-scsi0      /pci@4,2000/u4ftbus@0/u4ioslot@5/scsi@5
a-scsi1      /pci@6,4000/u4ftbus@1/u4ioslot@5/scsi@5
a-scsi0      /pci@6,2000/u4ftbus@0/u4ioslot@5/scsi@5
```

For the CD-ROM and tape devices there are four ‘magic’ aliases that do not appear as properties. They are ‘magic’ because the act of translating the alias causes the system to read the SCSI id of the CD-ROM or tape from the module configuration EEPROM, and insert the SCSI id into the device path.

These aliases are:

```
a-cdrom
b-cdrom
a-tape
b-tape
```

For example, to install from the CD-ROM in A-RMM, enter the command:

```
boot a-cdrom.
```

The alias `disk` no longer exists. The alias `net` translates to `a-net0`.

B.3 Configuration

B.3.1 Changed Defaults and Behavior

The following configuration variables have defaults different from those described in the *OBP 3.x Command Reference Manual*:

TABLE B-2 Configuration Variables with Changed Defaults

Variable	Old default	New default
input-device	keyboard	`/u4ftser`
output-device	screen	`/u4ftser`
boot-device	disk	-
diag-device	disk net	-
local-mac-address?	false	true

B.3.2 New Configuration Variables

There are no new configuration variables in the `/options` node. There is a new node called `u4ft-options` which represents configuration information stored in the `/u4ft-eeprom` NVRAM. You should not generally change these variables. They are read from the motherboard configuration EEPROMs as part of the global configuration process described on page B-8.

The NVRAM contains the following relevant variables:

`auto-reboot-on-trap?`

Default is `true`. Tells the system to continue the booting sequence with the next potential device if the boot drops back to OBP with a trap.

`no-run-post`

A variable to tell the system not to run POST after a full power-on reset. To avoid running POST enter:

```
h# deaf to no-run-post
```

This setting persists within the CPUset module until cleared. If this variable does not contain the magic value `deaf` then POST will be run. POST can be re-enabled by entering the command: `0 to no-run-post`

`boot-list`

A list of boot devices to be tried. This is read from the motherboard EEPROMs.

`osdog-a` and `osdog-b`

Indicates whether the `osdog` is configured on.

`combined-host`, `system-mode`, `split-master`, `split-host-a`, `split-host-b`

Variables related to split mode. See Section B.3.7 “Split Mode” on page B-12 for details of split mode.

`owner-a`, `owner-b`, `fixed-a`, `fixed-b`

Variables related to split mode ownership.

`hostid1`, `hostid0`, `ether1`, `ether0`

System identity.

B.3.3 Global Configuration.

The global configuration information includes (among other things) the system MAC address, the hostid, the slots which are owned by the system, and whether the system is split.

There are three repositories for configuration information: the two motherboard configuration EEPROMs and the CPUset NVRAM. The values stored in the CPUset NVRAM appear as properties in the `/u4ft-options` node (described in Section B.2.1 “Device Tree” on page B-2). The information in the CPUset NVRAM is read from the motherboard EEPROMs at every CPUset module reset. The configuration algorithm selects what it believes to be the best source of information, then converts the information (if necessary) into the CPUset NVRAM format and stores it in the CPUset NVRAM. After the configuration has taken place, the CPUset NVRAM is the single source for all such information. Thus the configuration algorithm can result in choosing to use the configuration information in motherboard A or in motherboard B, or choosing to ignore both and use the information that is already in the CPUset NVRAM.

The PROM does not contain any safe words to change values in the motherboard EEPROMs. This task can be accomplished in Solaris using the module library, or utilities based on it.

B.3.3.1 Reading the Configuration Information

When the CPUset module is reset, it looks for configuration information in the motherboards and in its own NVRAM, and chooses the best source of information to copy to the CPUset NVRAM. The algorithm does a three-way vote, based on the system identity and the modification time. It gives the motherboards a higher weighting, since CPUset modules are more likely to be moved from one system to another.

The CPUset module looks for configuration information using the following pseudo-C algorithm:

CODE EXAMPLE B-3 Algorithm for Search for Configuration Information

```
attempt to read and verify EEPROM information from each
motherboard;
  if (both mbrds are accessible and valid) {
        if (mbrds from same system) {
           choose most recent motherboard eeprom
        } else if (cpu nvram valid) {
                if (cpu nvram and mbrd a from the same system) {
                      choose more recent of cpu nvram and mbrd a;
                } else
                if (cpu-nvram and mbrd b from same system) {
                      choose more recent of cpu nvram and mbrd b;
                } else {
                       choose mbrd a; /* arbitrary choice here */
                }
        } else {
                choose mbrd a; /* arbitrary choice here */
        }
  } else if (mbrd a valid /*  so mbrd b invalid */) {
        if (cpu nvram valid &&
            cpu nvram and mbrd a from the same system) {
                choose the more recent of cpu nvram and mbrd a;
        } else {
                choose mbrd a; /* the only valid one */
        }
  } else if (mbrd b valid /* so mbrd a invalid */) {
```

CODE EXAMPLE B-3 Algorithm for Search for Configuration Information *(Continued)*

```
         if (cpu nvram valid &&
             cpu nvram and mbrd b from the same system) {
                 choose the more recent of cpu nvram and mbrd b;
         } else {
                 choose mbrd b; /* the only valid one */
         }
   } else {
         /* both mbrds invalid */
         if (cpu nvram invalid)
                 invent a default cpu nvram;
         choose the cpu nvram
   }
 }
```

A motherboard EEPROM is valid if all of the following are true:

- the generic checksum is correct
- the machine specific checksum is correct
- the machine-specific FRU name is A-MBD on side A and B-MBD on side B
- the following fields are sane:
  - `system-mode` is `C` or `S`
  - `combined-host` is 0 or 1
  - the two “split-host” characters are 0 or 1
  - the Ethernet addresses are not 0

A CPUset NVRAM structure is valid if it has the correct magic number: `0xfa115afe`.

The `mtime` field is used to decide which of two pieces of information is more recent: however, if the values in the `mtime` fields are within 30 seconds of each other, they are considered to be the same time.

To decide whether two pieces of information relate to the same system, all twelve bytes of the `ethernet` fields are compared. The `hostid` fields are not used, because they should be generated from the ethernet addresses.

The configuration information is read very early in the reset sequence.

B.3.3.2 Using the Configuration Information.

Once the information has been extracted, it is used to create properties and set variables in the PROM.

The information is used as follows:

- if the system mode is not split, then the `combined_host` value is used to select a system identity from the two possible identities. If `combined_host` is 0, `ether0` becomes the system mac address, and `hostid0` becomes the system hostid. If `combined_host` is 1, `ether1` and `hostid1` are used. If `combined_host` is neither 0 or 1, then 1 is assumed. In a non-split system the primary side owns all of the slots, and the secondary owns none.
- if the system mode is split, then the split-host variables are used to select the right MAC address and hostid. On CPUset A `split-host-a` is used to index the MAC addresses and the hostids; on CPUset B, `split-host-b` is used. If `split-host-a` has the value 0, `split-host-b` should have the value 1, or vice versa, so whatever choice side A makes, side B will make the opposite. In a split system the `owner-a` and `owner-b` variables are used to decide which side owns each slot.

B.3.4 Console

The system has four console ports that are accessible to the user, and two diagnostic ports that are accessible only by removing a cover. The diagnostic ports are intended only for the use of service engineers.

There are two ports on each CAF module, a ‘console’ port, and a ‘modem’ port. These are so called because the former has no modem control lines, and the latter has. By default, the console output of the system comes out on both of the CAF console ports (provided both sides are powered on). The ports used by the system are configured by the `console-flags` configuration variable which is stored in the motherboard EEPROMs. This is a two-byte variable. The top byte contains the magic number 0x43 if the console flags are valid. The bottom byte is a bit mask where each bit indicates whether a given console port is to be included in the list of ports to be used. The bits are:

TABLE B-3 Bit Mask Values

Bit	Port
0	a-caf console
1	a-caf modem
2	b-caf console
3	b-caf modem
4	diagnostic (invisible)

The default setting is `0x4305` which is both CAF module console ports. This can be changed in Solaris in the same way as any other motherboard EEPROM variable.

Note – If one side is powered on much later than the other, then the CAF module on that side will not come into use by the side that is already powered on.

B.3.5 Remote Control Processor

Each side has a remote control processor, which is used to control power and resets for that side. See below for a description of the commands that can be sent to the remote control processor.

B.3.6 Multiple Networks

There are four built-in networks in the Netra ft 1800, with aliases `a-net0`, `a-net1`, `b-net0`, `b-net1`. During OBP initialization they are allocated `local-mac-address` properties from the configuration information as follows:

TABLE B-4 MAC Address Assignment

Network	Mac-address
`a-net0`	ether0
`a-net1`	ether0 + 1
`b-net0`	ether1
`b-net1`	ether1 + 1

This happens whether or not the system is split, and irrespective of the value of `combined-host` or the `split-host` variables, or the system `mac-address`. In addition, the `local-mac-address?` configuration variable is forcibly set to `true`, so that these addresses are used.

B.3.7 Split Mode

A system is split if the `system-mode` in `/u4ft-options` has the value 53 (hex). In a split system, there is no primary, and each side uses the CAF module console (or modem) on its own side as the console. Both sides will complete OBP initialization and attempt to boot independently (conditional on `auto-boot?`). A side of split system will be able to boot only from slots that it owns. The command `srr-show-all` displays slot ownership.

B.4 Booting and Testing the System

B.4.1 Booting

Booting from a specified device works exactly as on an Ultra 450. Booting from a default device is slightly different.

There are two sources for default devices, `boot-device` (or `diag-device`) and the `boot-list` property which has been read from the motherboard EEPROM. Both of these are treated as lists of devices to try, and the devices are tried in order with the `boot-device` devices first. The `boot-list` property should be a list of devices in the Sun StorEdge Volume Manager root disk group. A record is kept of the devices tried, so no device is tried twice. The record is kept in NVRAM, so that the operating environment can decide that the boot is from the wrong device. In this case the operating environment performs a special reboot which indicates to OBP that it should carry on from where it left off in the boot list rather than starting again at the beginning. This also happens if the boot fails because of a trap that goes back to OBP (for example, if a corrupt bootstrap causes a trap).

There is a further subtlety in the ordering of the boot devices. If a device in the `boot-device` list is also in the `boot-list` list, then that device is not tried until it is encountered in the `boot-list`. For example if `boot-device` is `a-dsk0 a-dsk2 b-dsk1` and `boot-list` is `b-dsk2 a-dsk2`, then the order in which devices will be tried is `a-dsk0 b-dsk1 b-dsk2 a-dsk2`. The reasoning behind this is that the users set the devices in `boot-device` by hand, and so are presumably what they want to boot from. Since, however, the order of the devices in `boot-list` is significant, if the users put one of the `boot-list` devices into `boot-device` they could pervert the `boot-list` ordering. The ordering of the `boot-list` devices is significant because the first one contains the dump device, which could contain a crash dump; if another of the root disk group disks were tried first then the booting disk might mirror onto the device with the dump on it, thus destroying the dump.

B.4.2 Powering On

The two sides cannot be powered on simultaneously. They can be powered on using commands to the RCP (see below) or by pressing the switches on the CAF, but in neither case will both sides power on at the same time. The PROM is reasonably tolerant of delays between the two power-on actions, but if the delay is too long then the side which is powered on ignores the side that is not yet powered on, and continues to ignore it, even when it is powered on.

There is no simple method of bringing online a newly powered-on or reset side in OBP. Because of this, it is recommended that the two sides are powered on (or reset) within 10 seconds of each other.

When a system is powered on, the two CPUset modules both attempt to become primary. In a system that is configured as fault tolerant (that is, not split) the side that wins gets both CAF module consoles, and this is the one that boots the operating system. The other side (known as the secondary) goes through the same initialization procedure, but instead of booting, it executes a loop in which it waits for commands from the primary. These commands are used to bring the two CPUset modules into sync. The output of the secondary is visible only at the diagnostic console of the secondary CPUset module and or using the `u4fticc` driver if the primary is running Solaris.

B.4.3 Testing

The descriptions in the *OBP 3.x Command Reference Manual* also apply to the Netra ft 1800 OBP.

The word `probe-scsi` has been given a parameter, so that `probe-scsi b-scsi0` will probe the SCSI bus on side B bridge 0 (the drive chassis). The default is `a-scsi0`.

A 'pre-boot' test is applied before booting. If this test succeeds then the boot is continued. If the test fails, then OBP looks to see if there is another CPUset module waiting to be brought into sync. If there is, then the failing CPUset module power-cycles its own side. The waiting CPUset module sees this and immediately resets and becomes the primary, thus becoming the booting CPUset.

If there is no waiting CPUset module, but the failing CPUset module can see that there is another CPUset module powered on, then the failing module delays for 10 minutes, watching to see if the other CPUset module gets to the point of waiting to be brought into sync. If the other CPUset module is not powered on, or does not get to the waiting for sync stage, then the failing module attempts the boot anyway.

The 'pre-boot' test, is a combination of the results of POST (which tests the CPUset module only) plus a connectivity test of the paths to the motherboards and the boot devices. The devices themselves are not tested as part of this test, because the boot list processing handles faulty boot devices, and the fault tolerant operating system handles faults in other devices, and also because the point of the test is to decide whether or not to pass the responsibility for booting over to another CPUset module. For this purpose a device test is pointless, since the device will be equally faulty when seen from either side.

B.5 Using Forth Tools

B.5.1 Maintenance Bus

There are many commands in the Netra ft 1800 which are used to access the maintenance buses. There are two types of maintenance bus command; those in which the module name is implicit in the command, and those where a module must first be selected before the command can be used. With the latter type, once a maintenance bus has been selected, all subsequent commands of that type refer to the most recently selected maintenance bus. The former type implicitly alters the selected maintenance bus.

B.5.1.1 Selection Words

To select a maintenance bus, type the location name of the module to which you wish to send commands. TABLE B-5 contains a complete list. The selection words also select a slot response RAM (see below). Note that the last four words all select the same maintenance bus, but different slot response RAMs (SRRs).

TABLE B-5 Specifying a Maintenance Bus

Motherboard A	Motherboard B	Description
`a-cpu`	`b-cpu`	CPUset - selects srr for pba on bridge 0
`a-mbd`	`b-mbd`	Motherboard
`a-caf`	`b-caf`	Caf
`a-net0`	`b-net0`	Caf - selects srr for net0
`a-net1`	`b-net1`	Caf - selects srr for net1
`a-rmm`	`b-rmm`	Removable media module
`a-psu0`	`b-psu0`	Power Supply 0 - no srr selected
`a-psu1`	`b-psu1`	Power Supply 1 - no srr selected
`a-psu2`	`b-psu2`	Power Supply 2 - no srr selected
`a-dsk`	`b-dsk`	Drive module
`a-pci0`	`b-pci0`	PCI slot 0
`a-pci1`	`b-pci1`	PCI slot 1

TABLE B-5 Specifying a Maintenance Bus

Motherboard A	Motherboard B	Description
`a-pci2`	`b-pci2`	PCI slot 2
`a-pci3`	`b-pci3`	PCI slot 3
`a-pci4`	`b-pci4`	PCI slot 4
`a-pci5`	`b-pci5`	PCI slot 5
`a-pci6`	`b-pci6`	PCI slot 6
`a-pci7`	`b-pci7`	PCI slot 7
`a-cpu-a-0`	`b-cpu-a-0`	CPUset - selects srr for pba on bridge 0
`a-cpu-a-1`	`b-cpu-a-1`	CPUset - selects srr for pba on bridge 1
`a-cpu-b-0`	`b-cpu-b-0`	CPUset - selects srr for pbb on bridge 0
`a-cpu-b-1`	`b-cpu-b-1`	CPUset - selects srr for pbb on bridge 1

Examples:

- `b-cpu 40 lm78@` selects the CPU maintenance bus on motherboard B and reads register 0x40 from the `lm78` on that bus (see below).
- `a-rmm 30 eeprom@` selects the removable media maintenance bus on motherboard A and reads byte 0x30 from the EEPROM on that bus.

B.5.1.2 Global Display Words

These words apply maintenance bus operations to all maintenance buses in the system. They implicitly select the maintenance buses required.

- `probe-mbus` — probes all of the maintenance buses it can find, and prints a summary. There is also `a-probe-mbus` and `b-probe-mbus` for the respective motherboards.
- `all-leds-on –` tries to turn on all the software controllable LEDs it can find. This does not include power LEDs. Also `a-all-leds-on` and `b-all-leds-on`.
- `all-leds-off` — tries to turn off all the software controllable LEDs it can find. This does not include power LEDs. Also `a-all-leds-off` and `b-all-leds-off`.

B.5.1.3 Maintenance Bus Information Display Words for Specific Modules.

The following words can be used to display the status of the fan/temperature controller (`lm78`) on the corresponding module. They select the module implicitly.

TABLE B-6 Status Commands

Motherboard A	Motherboard B
`.a-caf-lm78`	`.b-caf-lm78`
`.a-rmm-lm78`	`.b-rmm-lm78`
`.a-psu0-lm78`	`.b-psu0-lm78`
`.a-psu1-lm78`	`.b-psu1-lm78`
`.a-psu2-lm78`	`.b-psu2-lm78`
`.a-cpu-lm78`	`.b-cpu-lm78`

The following words display the state of a drive module and the disks in it. They select the module implicitly.

TABLE B-7 DSK Module Status Commands

Motherboard A	Motherboard B
`.a-drive-chassis`	`.b-drive-chassis`

The following words display the status of the PSU. They select the module implicitly:

TABLE B-8 PSU Status Commands

Motherboard A	Motherboard B
`.a-psu0-status`	`.b-psu0-status`
`.a-psu1-status`	`.b-psu1-status`
`.a-psu2-status`	`.b-psu2-status`

The following words display the state of the motherboard I/O ports:

TABLE B-9 MBD I/O Ports Status Commands

Motherboard A	Motherboard B
`.a-mbrd-ports`	`.b-mbrd-ports`

The maintenance bus access words described in the remaining sections return 0 for success and non-zero for failure. In the case of success the stack will have 0 at the top and the return value if any below it. This means that after calling the read access word you must either test or “drop” (if you are certain of success) the error code before getting the value returned.

In general, words intended to display values do not return an error code as above.

B.5.1.4 EEPROM Access Words.

All modules except HDD modules have an EEPROM.

To display a range of a module EEPROM, first select the module, then use:

```
<offset> <size> edump
```

There is no word to decode the EEPROM contents and display it in human-readable form.

The words to access the EEPROM on a bus (provided it exists) are:

- *offset* `eeprom@` -- *value errcode*: returns the byte from the given offset of the EEPROM. For example, `cpu-bb d# 32 eeprom@` returns byte 32 (decimal) of the EEPROM on the CPU in motherboard B.
- *value offset* `eeprom!`-- *errcode*: writes the given byte to the offset in the EEPROM. For example, `a-caf h# 5a h# 20 eeprom!` writes the value 0x5a to address 0x20 in the EEPROM on the CAF in motherboard A.

B.5.1.5 LM75 Access Words

The LM75 is a simple temperature sensor that is used on modules without fans.

The only word that can be used to access the LM75 is:

- `lm75@` -- *value errcode*: this returns the contents of the LM75 temperature which represents the temperature in degrees Celsius. Note that by default the value will be displayed in hex — use `.d` to display it in decimal.

There is no LM75 display word.

B.5.1.6 LM78 Access Words.

The LM78 is a fan and temperature controller. Most modules have only one LM78. The CAF has two, because it has four fans.

To select an LM78 on the CAF module, use `lm78-use-alt` to get the LM78 at 0x1d, and `lm78-use-norm` to get the `lm78` at 0x2d.

To set this up, use `setup-caf-lm78`, which sets the base address of the default LM78, and then enables the second LM78.

Low Level Access Words.

At a lower level, two basic LM78 access words are:

- *address* `lm78@` -- *value errcode*: fetches the value of the register at the specified address in the LM78.
- *value address* `lm78!` -- *errcode*: writes the specified value to the register at the specified address.

The mapping of addresses to registers is described in the `lm78` data sheet. The most commonly used values are shown in TABLE B-10:

TABLE B-10 Register-to-Address Mappings

Register	Address	Forth name
config	0x40	`lm-config`
fans divisor	0x47	`lm-div`
external temperature	0x20	`lm-cpu-temp`
internal temperature	0x27	`lm-temp`
fan0	0x28	`lm-fan0`
fan1	0x29	`lm-fan0 1 +`

Access Words

- *fan#* `lm78-fan@` -- *value errcode*: returns the value of the fan counter register.
- *fan#* `lm78-fan-speed` -- *value errcode*: returns the value of the fan counter translated to RPM, taking account of the value of the divider register.
- *fan#* `lm78-fan-div@` -- *value errcode*: returns the value of the fan divisor being used for the current fan.

Note – This routine decodes the appropriate register, and returns the actual divisor, that is, 1, 2, 4 or 8.

- *divisor fan#* `lm78-fan-div!` -- *errcode*: set the fan divisor (use the actual divisor).
- `lm78-cpu-temp-reg@` -- *value errcode*: returns the value of the adc register controlled by the external temperature sensor.
- `lm78-cpu-temp@` -- *value* /ercode/ returns the value of the CPU temperature translated into centigrade.
- `lm78-start`: -- *errcode*: start the LM78.
- `lm78-stop: --` *errcode*`:` stop the LM78.
- `lm78-reset`: -- *errcode*: reset the LM78 (stops it).

Display

The first three sets of display word select the LM78, start it if necessary and have some knowledge of what ought to be there (that is, how many fans, and whether the CPU temperature is appropriate).

- `.a-caf-lm78` and `.b-caf-lm78`
- `.a-cpu-lm78` and `.b-cpu-lm78`
- `.a-rmm-lm78` and `.b-rmm-lm78`

The remaining words are not as powerful as those above: you have to select the right bus, and start the `lm78`.

- `.lm78`: displays everything. This does not translate any of the values from the ADCs, because this is not generic, but the fan speed translation is done.
- `.lm78-cpu-temp`: displays the temperature (in Celsius) registered by the external temperature sensor. This routine takes the value from the relevant ADC and converts it to Celsius using a table.

Note – This value will only be appropriate if read from the LM78 on the CPUset module.

- `.lm78-temp`: displays the ambient temperature as measured by the LM78 internal thermometer.
- *fan#* `.lm78-fan`: displays the fan speed indicated for the specified fan, taking account of the current value of the divisor for the specified fan.

B.5.2 General Purpose I/O Ports

The 8574s are known as ‘ports’, and are labelled `port0`, `port1` ... `port7`. Most maintenance buses have only one port.

B.5.2.1 Low Level Access

The following low level access routines are available:

- `port0@` -- *value error*: returns the value in port0 on the chosen maintenance bus. Analogously, there are also `port1@`, `port2@`, and so forth.
- *value* `port0!` -- *error*: writes the value to port0. Also `port1!`, `port2!`.

B.5.2.2 Higher Level Access

The following words control the power and LEDs on modules:

- `fault-on` -- *error*: turn on the fault LED of the module.
- `fault-off` -- *error*: turn off the fault LED of the module.
- `power-on-module` -- *error*: turn on the module power.

- `power-off-module` -- *error*: turn off the module power.

Note that `power-on-module` and `power-off-module` refer to the drive chassis when you have selected a drive chassis. To control an individual disk use:

- *disk#* `disk-fault-on`: turn on the fault LED of the disk.
- *disk#* `disk-fault-off`: turn off the fault LED of the disk.
- *disk#* `disk-power-on`: turn on the disk's power.
- *disk#* `disk-power-off`: turn off the disk's power.

B.5.3 Slot Response RAM

The state of the slot response RAM (SRR) for the entire system can be shown by `srr-show-all`. To show the slot response RAM for a specific bridge use `srr-show-a-0`, `srr-show-a-1`, `srr-show-b-0`, or `srr-show-b-1`.

To perform operations on a specific SRR, select the SRR. All subsequent SRR operations are directed to that SRR until another one is selected.

Note – The `show-all` words implicitly select an SRR. If you have one selected, you must select again it after using a `show-all`.

The SRR selection words are the same as the maintenance bus selection words described in Section B.5.1.1 "Selection Words" on page B-15. The four CPU selection words which select the same maintenance bus, but different slot response RAM entries.

The slot response RAM access words are:

- `srr@` - returns the value of the slot RAM.
- *value* `srr!` - writes *value* to the slot RAM.
- *side* `claim-slot` - try and claim the slot for the side - returns 0 for failure, 1 for success.
- `arbiter-on`, `arbiter-off` — attempt to turn on/off the arbiter for the selected slot. Returns 1 for success, 0 for failure.
- `faketa-on`, `faketa-off` — attempt to turn on/off the faketa bit in the SRR. Returns 1 for success, 0 for failure.

Writes to slot response RAM entries will fail if the slot is not owned by the CPUset module doing the write, so care must be taken not to give away ownership of the slot.

B.5.4 Remote Control Processor (RCP)

The two remote control processors are accessible in two ways: via a 9 pin D-type connector on each CAF module, and from the CPUset module, using OBP commands.

B.5.5 Access from the External Connector

Each of the connectors can be used to talk to both RCPs. The commands are:

TABLE B-11 RCP Commands

Command Name	Effect
`AAres`	reset motherboard A
`BBres`	reset motherboard B
`AAoff`	power off motherboard A
`BBoff`	power-off motherboard B
`AAon`	power-on motherboard A
`BBon`	power-on motherboard B
`AAstat`	print RCP A status in hex
`BBstat`	print RCP B status in hex

The RCP has independent power so is still accessible even if it has been used to turn off its side.

B.5.6 Access from OBP

A CPUset module can always access the RCP on its own side (that is, CPUset A can always access the RCP on side A, and similarly for CPUset B). A CPUset module can only access the RCP for the other side if there is no CPUset module inserted in the other side, or if the CPUset module in the other side is not powered on.

The following commands are provided:

TABLE B-12 OBP Access to RCP

RCP A	RCP B	Description
`.a-rcp-status`	`.b-rcp-status`	print the status of the rcp (decoded)
`a-rcp-clr-status`	`b-rcp-clr-status`	clear the rcp status
`a-rcp-reset-side`	`b-rcp-reset-side`	reset the side.
`a-rcp-poff-side`	`b-rcp-poff-side`	power-off the side
`a-rcp-pon-side`	`b-rcp-pon-side`	power-on the side
`a-rcp-cycle-side`	`b-rcp-cycle-side`	power-cycle the side
`a-rcp-force-dload`	`b-rcp-force-dload`	force a download of the side's fpgas (implies a reset)
`a-rcp-acrecover`	`b-rcp-acrecover`	tell RCP to power-down and watch for AC to return

B.5.7 Netra ft 1800 Motherboard Registers.

Only two commands are useful to the average user to display information from the motherboard registers: `.a-fc` and `.b-fc`. These display a large amount of information from the registers, including whether the corresponding motherboard is in EState, whether it is primary, and whether the other motherboard is powered on.

B.6 Loading and Executing Programs

B.6.1 `dload`

The `dload` word in OBP relied on the alias `net` being the only network, so it has been replaced by the new word `netload`. Instead of using, for instance:

```
load-base dload new-file.fth
```

You can use:

```
load-base netload b-net1 new-file.fth
```

The same applies to `watch-net` — so you can now type `watch-net b-net1`.

Glossary

ASIC	Application-Specific Integrated Circuit.
ASR	Automatic System Recovery: reboot on system hang.
BMX+	Crossbar switch *ASIC*.
bridge	The interface between the *CPUsets* and the I/O devices.
CAF	Console, Alarms and Fans *module*.
CMS	Configuration Management System. The software that records and monitors the *modules* in the system. Users access the CMS via a set of utilities which they use to add and remove modules from the system configuration and *enable* and *disable* modules that are in the system configuration.
component	An identifiable part of a *module*.
configure	(*CMS*) Notify the CMS that a *module* is present in a specified location.
constituent	(*CMS*) An object that provides part of the functionality of another object. An object references its constituents.
CPUset	A *module* containing the system processors and associated components.
craft-replaceable	A *module* which clearly indicates when it is faulty and can be *hot-replaced* by a trained craftsperson.
DIMM	Dual Inline Memory Module.
disable	(*CMS*). Bring offline and power down a *module*.
DMA	Direct Memory Access.
DRAM	Dynamic Random Access Memory.
DSK	Disk chassis *module*.
DVMA	Direct Virtual Memory Access. A mechanism to enable a device on the PCI bus to initiate data transfers between it and the CPUsets.

ECC	Error Correcting Code.
EEPROM	Electrically Erasable Programmable Read Only Memory.
EMI	Electro-magnetic Interference.
enable	(*CMS*) Power up and bring online a *module* that is already *configured* into the system.
engineer-replaceable	A *module* which may not indicate that it is faulty and which may require special tools for diagnosis and replacement. The Netra ft 1800 does not have any engineer-replacable modules.
ESD	ElectroStatic Discharge.
EState	Error limitation mode.
fault tolerant	A system in which no single hardware failure can disrupt system operation.
fault-free	No faults are evident in the operating system, or application software, or in external systems, except in the case of certain high demand real-time uses.
faulty module	A *module* one or more of whose devices have gone into the degraded or failed states, as indicated to the *CMS* via the *hot-plug* device driver framework.
FPGA	Field Programmable Gate Array.
front-replaceable	The ability to replace a *module* from the front of the system.
FRU	Field Replacable Unit. Another name for a *module*, used within the *CMS*.
HDD	Hard Disk Drive.
hardened	Specially engineered to be resistant to hardware and some causes of software failure. Applies to device drivers.
health features	Features that can indicate that a fault is about to occur.
hot plug	The ability to insert or remove a *module* without causing an interruption of service to the operating platform.
hotPCI	An implementation of the PCI bus designed to minimize the probability that a fault on a *module* will corrupt the bus, and so to ensure that the system control mechanism runs without interruption
hot-replaceable	A *module* that can be replaced without stopping the system.
I^2C	Inter Integrated Circuit
IOMMU	Input/Output Memory Management Unit
LED	Light-Emitting Diode.

location	A slot where a *module* can be inserted. Each location has a unique name and is clearly marked on the chassis.
lockstep	The process by which two *CPUsets* work in synchronization.
losing side	The side of a *split* system which has a new identity when rebooted.
MBD	Motherboard.
Mbus	Maintenance bus.
module	An assembly that can be replaced without requiring the base machine to be returned to the factory. A module is a physical assembly that has a module number which is stored in the software on the machine, generally in the *EEPROM* of the physical assembly
PCI	Peripheral Component Interconnect.
PCIO	PCI-to-Ebus2/Ethernet controller *ASIC*.
PRI	Processor re-integration. The process by which the two CPUsets come into *lockstep* to function as a fault tolerant system. *Re-integration* is preferred.
PROM	Programmable Read Only Memory.
PSU	Power Supply Unit.
RAS	Reliability, Availability and Serviceability.
RCP	Remote Control Processor.
RMM	Removable Media Module.
RS232	An EIA specification that defines the interface between DTE and DCE using asynchronous binary data interchange.
SC_UP+	System controller *ASIC*.
side	One *CPUset* and its associated *modules*, capable of running as a standalone system. A side is one half of a *fault tolerant* system or one of two systems in a *split* system.
SPF	Single Point of Failure.
split system	A system whose two *sides* run as separate systems.
stealthy PRI	Stealthy processor re-integration. Processor re-integration (*PRI*) which is completed without user intervention.
subsystem	(*CMS*) A fault tolerant configuration of *modules* defined in the CMS.
system attribute	(*CMS*) An attribute of a CMS object that is written only by the CMS.
TLB	Translation Lookaside Buffer. The hardware which handles the mapping of virtual addresses to real addresses.

surviving side	The side of a *split* system which retains the identity of the previous *fault tolerant* system.
U2P	UPA-to-PCI bridge (U2P) *ASIC*.
UPA	UltraSPARC Port Architecture.

Index